Implementation-Level Corruptions in Distance Bounding – Exhibiting Faults and Provably-Secure Fixes in the Electronic Payment Protocol `PayPass` –

Ioana Boureanu[1], David Gerault[2], and Pascal Lafourcade[3]

[1] University of Surrey SCCS `i.boureanu@surrey.ac.uk`
[2] Temasek Laboratories, Nanyang Technological University, Singapore
[3] University Clermont Auvergne LIMOS `pascal.lafourcade@uca.fr`

Abstract. In relay attacks, a man-in-the-middle attacker gains access to a service by relaying the messages between two legitimate parties. Distance-bounding protocols are a countermeasure to relay attacks, whereby a verifier measures the round-trip time in exchanges with a prover. Inspired by application-security definitions, we propose a new security model, *OracleDB*, distinguishing two prover-corruption types: black-box and white-box. We use this distinction to settle the long-lasting arguments about terrorist-fraud resistance, by showing that it is irrelevant in both the black-box and white-box corruption models. We then exhibit a security flaw in the PayPass protocol with relay protection, used in EMV contactless payments. We propose an extension to this industry-standard protocol, with only small modifications, and prove its security in our strongest adversary model. Finally, we exhibit a new generalised distance-fraud attack strategy that defeats the security claims of at least 12 existing distance-bounding protocols.

1 Introduction

In 2015, the most widely used contactless electronic-payment protocol, EMV (Europay, Mastercard and Visa), was shown susceptible to relay attacks [19]. In 2016, the EMV standard included relay-protection for the contactless protocol PayPass [35]. The relay-countermeasure imposes an upper-bound on the round-trip times (RTTs) of message-exchanges. This mechanism is often referred to as *proximity-checking*: a prover (e.g., contactless card) proves that it is within close distance to a given verifier (e.g., EMV reader). Often, within the proximity-proof, the former also authenticates itself to the latter. This primitive where *proximity-checking is composed with a unilateral authentication or identification mechanism* bares the name of *distance-bounding* (DB) [16].

A Recap of the DB Threats. In DB, the tag and the reader are often referred to as the *prover* and the *verifier*. In the vast literature covering DB protocols [2], three "classical" types of attacks have been distinguished. In a *distance-fraud* (DF), a dishonest prover P^* tries to prove that he is within the distance bound when in fact he is not. A *mafia-fraud* (MF) attack involves three entities: a honest prover P, far-away from an honest verifier V, and an adversary $\mathcal{A}$; $\mathcal{A}$ tries to authenticate as P. Finally, in a *terrorist-fraud* (TF), a dishonest prover P^* colludes with an accomplice $\mathcal{A}$ so that this accomplice authenticates as P, while P is outside of the distance bound. The fraud is considered successful only if $\mathcal{A}$ cannot impersonate P in a latter session. This excludes the trivial and difficult to counter attack in which P gives his credentials to $\mathcal{A}$. In recent years, the DB threat model has been widened: [23] coined *impersonation attacks*, [20] presented distance hijacking — which extends distance fraud by letting P^* exploit honest provers located near V, [14] gives further generalisations of these: e.g., mafia-fraud captured via more powerful *men-in-the-middle (MiM)* and terrorist-fraud up-cast to *collusion-fraud*, which considers repeated collusions between P^* and $\mathcal{A}$. The notion of provable terrorist fraud resistance is yet notoriously hard to capture in a sensible way [27,49].

The Rise of Application-Level Distance-Bounding. Contactless EMV with relay protection is being studied as a DB protocol in [22,36]. At Usenix 2018, Chothia et al. [18] presented a hierarchy of properties against which PayPass with relay protection was analysed. These recent security analyses also look at other application-level DB, such as the Mifare protocol from NXP. In this trend, we argue that a finer threat model is necessary to analysis application-level DB.

Contributions. 1. We propose *OracleDB*, a new security model for distance bounding. Inspired by real-life implementations, this model explicitly distinguishes *specification* and *implementation* of DB protocols, and black- and white-box provers. Our model covers all DB threats with 3 attacks: a black-box threat called *secret-extraction*, a white-box *generalised distance fraud*, and a *generalised mafia fraud*.
2. We show that terrorist-fraud resistance is irrelevant: we exhibit a generic TF in the white-box model, and show that TF-resistance is meaningless in the black-box model.
3. We use *OracleDB* to show that the current version of the EMV contactless protocol PayPass with relay protection is insecure. We propose a minor modification, and prove the security of the resulting protocol secure in our strongest adversarial setting.
4. We use *OracleDB* to exhibit a generalised distance-fraud on 13 existing protocols.

2 OracleDB: A Refined Provable-Security Model for Distance Bounding

2.1 Protocols & Distance-related Aspects

Security Parameter. Security measures are given depending on a *security parameter s*. The associated notions, such as polynomial probabilistic time (ppt), negligible, overwhelming, and others (such as Interactive Turing Machines (ITMs) [48]) are considered commonplace; we do not remind their definitions.

Definition 1 (Specification of a DB Protocol.). *A* specification Π of a DB protocol *is a tuple* $\Pi = (\mathbb{P}, \mathbb{V}, idents, s, \texttt{dist-bound})$*, where:*
– s *is a variable to hold the values of the security parameter;*
– `dist-bound` *is a variable to hold the values of the* distance bound *allowed by* Π *(which may vary with s);*
– *The* prover algorithm $\mathbb{P}$ *and* verifier algorithm $\mathbb{V}$ *are ITMs running probabilistic polynomial time algorithms in s;*
– $\mathbb{P}$ *and* $\mathbb{V}$ *run an identification/authentication scheme which uses the identities idents;*
– $\mathbb{V}$ *contains measurements of the round-trip communication times which it compares against* `dist-bound`*, and outputs* 0 *or* 1*, denoting respectively failure or success.*

Definition 1 corresponds to how formal models introduce a DB protocol: it encodes its algorithmic specification.
<u>Parties or Devices.</u> We take the view that for a more faithful security-analysis we should move beyond Definition 1 into *explicitly* capturing the fact that a DB protocol is realised via *actual implementations* of the $\mathbb{P}$ and $\mathbb{V}$. In this sense, we use the notion of a *"party"* or *"device"* to denote as the implementation of an algorithm (in hardware or software). This algorithm can be that of a prover, verifier, or adversary. Def. 2 below will leave the implementation of $\mathbb{V}$ open (since herein we will not focus on corruptions coming from this ITM). Instead, Def. 2 nominates two classes of implementations for $\mathbb{P}$.

Definition 2 (Protocol Parties and DB Realisation.). *A* realisation $\Pi^{\texttt{real}}$ of a specification Π of a DB protocol *is a tuple* Π^{real}=$(Setup, P_{XB}, V, \mathbb{B}, s)$*, where:*
– *s is the variable to hold the values of the security parameter as per Π and it is left free (i.e., uninstantiated);*
– $\mathbb{B}$ *is an actual value of the* distance bound *allowed by Π (which may be given as function of s);*
– *Setup is an algorithm that instantiates Π's idents by generating or pulling from a database* long-term identifiable information *(typically cryptographic keys and/or PUF challenge/response pairs), where the size/security of this identifiable information depends on s. This procedure is an abstraction for the procedures used to make a new prover and/or verifier join the system, obtain long-term identifiable information, and be recognised by the previously instantiated parties. It can be called several times.*
– *a* verifier-party/verifier-device V *is an implementation identical to the Π-specified $\mathbb{V}$ together with Setup-instantiated long-term identifiable information;*
– *a* prover-party/prover-device P_{XB} *is an implementation of $\mathbb{P}$ with its own Setup-instantiated long-term identifiable information, accordingly to one of the* two possible types of implementation $\{P_{XB} \mid XB \in \{\mathsf{WB}, \mathsf{BB}\}\}$ for $\mathbb{P}$*:* white-box (WB) *and* black-box (BB)*, respectively.*

 • *If the prover-party is realised in the white-box manner, i.e., P_{XB}=P_{WB}, then there exists an algorithm ppt. in s that can retrieve the long-term identifiable information (created by Setup), found on the device realising $\mathbb{P}$.*

 • *If the prover-party is realised in the black-box manner, i.e., P_{XB}=P_{BB}, then any ppt. algorithm in s will interact with P_{BB} only in the same way any ppt. ITM interacts with $\mathbb{P}$.*

We stress that Def. 2 per se does not give one specific implementation of a specification Π of a DB protocol. Instead, Def. 2 stipulates that there is an algorithm called *Setup* which can be used to produce correct implementations of the verifier algorithm, as well as implementations of the prover algorithm that can be either white-box (WB) or black-box (BB). On the one hand, if an implementation produced is of the BB-type, then it is one of the many possible correct implementation which cannot be altered or tampered with. In fact, the last bullet point in Def. 2 denotes that we assume, that the black-box implementations simply ignore all messages that do not conform with the protocol specification. On the other hand, if an implementation is of WB-type, then it is one of the many possible implementations that can be fully read and altered by an third party.

The *Setup* Procedure. One important point of Def. 2 is its inner *Setup* procedure. We mean for this procedure to encode the creation of real-life prover-devices/verifier-devices and managing their identities/enrolment into a back-end system. This typically consists in generating appropriate cryptographic keys such that each prover can communicate with each verifier, be it symmetric keys or public/private key pairs. We do not expand further on the details of *Setup*, as we believe that –for the purpose of this material– modelling the "backend" of registering real-life provers and verifiers is not necessary.

Uniquely Identifiable Devices. As per Definition 2, each party carries uniquely identifiable and secret, long-term information on it, such as private keys or PUFs (Physically Unclonable Functions). We assume that each such secret information is mapped by a non-invertible function to a *publicly disclosable, unique identifier* denoted *id*. For a party X, we write $X.id$ to directly refer to its publicly disclosable identifier *id*. Alternatively, we sometimes write party P_i or V_j to mean that it is a prover-device with publicly disclosable identifier i or, respectively, a verifier-party with publicly disclosable identifier j. From now on, we use simply "identifier" instead of "publicly disclosable identifier".

Device Holders. Prover-devices are necessarily held by a *holder*. A holder h holding a prover P_{XB} is denoted as $h^{P_{XB}}$. The holder is an entity, which can be assumed to be a human, a robot, a car, etc. We also allow holders to hold several devices $\{P_1, P_2, \ldots\}$, up to a polynomial number in the security parameter s. In this case, we write $h^{\{P_1,P_2,\ldots\}}$ or simply $h^{\{P\}}$. We only deal with protocols that cannot identify holders[4].

Locations. Each party B, and therefore the holder h_B of any prover-device, occupy one *position* pos_B within an Euclidean space, *i.e.*, a point in the space. In a protocol realisation, we say that two parties are *close* if the Euclidean distance between their positions is at most $\mathbb{B}$, and *far* otherwise. More specifically, we operate a notion of *location*, which allows us to define a party being close or far to an area of interest.

Definition 3. ***Location.*** *Let Π^{real} be a realisation of a DB protocol. Let $\mathcal{P}$ be a set parties and let them be respectively fixed at positions in the Euclidean space. By abuse of notation, let the resulting set of positions also be called $\mathcal{P}$. A $\mathcal{P}$-*location*, or simply* location*, is a $\mathbb{B}$-neighbourhood of $\mathcal{P}$, which is the set of all Euclidean points (i.e., positions) that are at distance at most $\mathbb{B}$ from $\mathcal{P}$ (or, the union of all the open balls of radius $\mathbb{B}$ that are centred at a point in $\mathcal{P}$.*

If a party E found in a location loc, we often write E_{loc}.

Definition 4. ***Distance between locations.*** *The distance between a $\mathcal{P}_1$-location loc_1 and a $\mathcal{P}_2$-location loc_2 is the shortest distance between a party in $\mathcal{P}_1$ and a party in $\mathcal{P}_2$.*

Let us explain Def. 3. We fix a set $\mathcal{P}$ of parties, and we are interested in singling out all the possible positions of other parties which are no further than $\mathbb{B}$ from devices in $\mathcal{P}$. All these close-by positions give us the notion of $\mathcal{P}$-location, formalised in Def. 3. We simply write *location*, when the set $\mathcal{P}$ is implicit or un-important. Parties can be found only inside locations, *i.e.*, we always reason over one or several locations.

An illustration is presented in Figure 1, where $\mathbb{B}$=1 and the parties are A, B, C. The $\{A, B, C\}$-location therefore is the union of the three circles of radius 1. One set $\mathcal{P}$ may be formed, for instance, of just one verifier-party, and therefore the location may be just one circle.

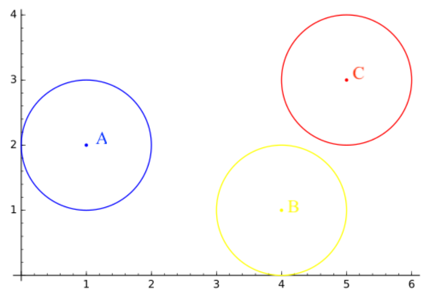

Fig. 1. An $\{A, B, C\}$-Location (Def. 3): Portion of the space no further than $\mathbb{B}$ from A, portion of the space no further than $\mathbb{B}$ from B, portion of the space no further than $\mathbb{B}$ from C.

[4] If verifier-devices can reliably ascertain the exact holder of the device in contactless settings, then relay attacks would hardly be a problem.

The notion of location is quite general. The specific ways in which we use it are restricted via the definition below.

Definition 5. ***Appropriate Universe of Locations.*** *Let Π^{real} be a realisation of a DB protocol, with two aprioriliy fixed sets $\mathcal{P}_1$ and $\mathcal{P}_2$ of parties. An* appropriate universe of locations *$\mathcal{U} = (loc_1, loc_2)$ is formed of a $\mathcal{P}_1$-location loc_1 and a $\mathcal{P}_2$-location loc_2 such that:*
1. any two parties in each locations are close together, i.e., within the bound from one another;
2. these two locations are the closest two locations such that the distance between loc_1 and loc_2 is larger than $\mathbb{B}$.

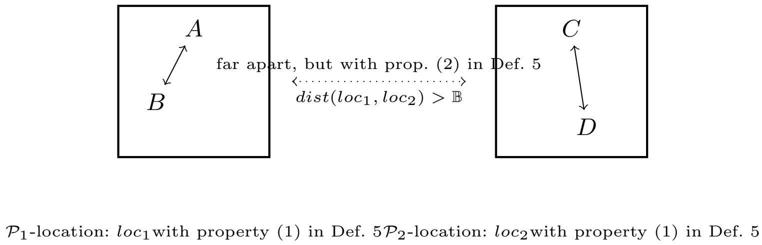

Fig. 2. An appropriate universe of locations. A full arrow denotes a distance lower than $\mathbb{B}$, and a dotted arrow denotes a distance larger than $\mathbb{B}$.

Figure 2 illustrates Def. 5: parties A and B are in one location and are close to each other, and parties C and D are close to each other in another location (condition one in Def. 5). Party A is further than $\mathbb{B}$ from party C and from party D, and party B is further than $\mathbb{B}$ from party C and party D (condition two in Def. 5). Note that condition two in Def. 5 is an over-approximation of the concept of parties being far apart, as it demands that *any* party in loc_1 is far away from *any* party in loc_2.

There may multiple valid appropriate universes of locations, depending on which parties we chose to include. For instance, in Fig. 2, the set $\mathcal{P}_1$ could as well contain only A or B. The set of parties that we consider depends on the security property we study: we restrict ourselves to the set of parties that are relevant. This typically includes a honest holder, a set of verifiers and an adversary, because the presence of other honest holders nearby is generally irrelevant.

2.2 Communication & Threat Models

Communication Model. Let $\mathcal{U} = (loc_1, loc_2)$ be an appropriate universe of locations. As ITMs, the realisation of $\mathbb{P}$ and $\mathbb{V}$ can exchange messages. These messages are subject to a time of flight. There exists a *time-bound* $t^{\mathbb{B}}$, such that a message from a party C will reach a party D within the time $t^{\mathbb{B}}$ if and only if party C is *close* to party D (i.e., both in the same *loc*). Hence, if party A is in loc_1 and C is in loc_2, then a message from party A takes longer than $t^{\mathbb{B}}$ to reach party C, and vice versa.

All messages sent by honest parties are broadcast.

We assume that, at both on the prover and the verifier's side of a timed round trip, the computation of message is instantaneous.

Threat Model. Def. 2 introduced prover- and verifier-parties. We now introduce a new type of party: the adversary.

Definition 6. ***Adversary.*** *Let Π^{real} be the realisation of a DB protocol Π, and let $\mathcal{U} = (loc_1, loc_2)$ be an appropriate universe of locations.*

The adversary $\mathcal{A} = (\mathcal{A}_{loc_1}, \mathcal{A}_{loc_2})$ *is a party formed of two sub-parties $\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$, such that:*

(1). *Each device, $\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$, implements an arbitrary ppt. algorithm in the security parameter of Π^{real}.*

(2). *Devices $\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$ may be respectively present or absent from the two locations loc_1 and loc_2.*

(3). *If one such adversarial device is not present at its associated location loc, we consider the algorithm $\mathcal{A}_{loc}$ to be void.*

(4). *$\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$ operate as ITMs too (*i.e., *they collaborate and communicate).*

(5). *The communication between $\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$ follow the communication model: they cannot send/receive messages faster than $t^{\mathbb{B}}$.*

(6). *$\mathcal{A}$ cannot change the corruption level of a realised party, i.e., he cannot change a P_{BB} into P_{WB} or vice versa.*

(7). *$\mathcal{A}$ interacts with the verifier parties by sending them messages, and they reply honestly as per the specification Π.*

(8). *In addition, when a prover party is white-box,* i.e. *$P_{XB} = P_{WB}$, $\mathcal{A}$ can read its memory (but not modify it).*

(9). *$\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$ can act as holders*[5] *for prover-devices. In this case, we use a notation similar to that of honest holders: $\mathcal{A}_{loc}^{P}$ denotes that $\mathcal{A}_{loc}$ holds the prover-party P.*

(10). *$\mathcal{A}$ can* move *prover-parties,* i.e., *change their holder.*

(11). *$\mathcal{A}$ can* send unicast messages*, which can only be read by their intended target, e.g., using directional antennas [1].*

(12). *$\mathcal{A}$ can* block *any message from being received by a party of his choice, irrespectively of their position.*[6]*.*

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

(14). *Messages sent by $\mathcal{A}$ have priority,* i.e., *if a bit b sent by $\mathcal{A}$ arrives to a honest party B at the same time as another bit b' sent by a honest party C, then B ignores b' and reads b* [7]*.*

2.3 Concurrent-Execution Model

Now, we define what we consider in terms of possible executions of a realised DB protocol.

Definition 7. ***Execution Environments.*** *Let Π^{real} be a realisation of a DB protocol and let $\mathcal{U} = (loc_1, loc_2)$ be an appropriate universe of locations. An* execution environment for $\Pi^{\mathtt{real}}_{\mathcal{U}}$ at the universe of locations $\mathcal{U}$ *denotes a polynomial number of (possibly concurrent) executions by a polynomial number of prover-parties and verifier-parties, positioned in the universe of locations $\mathcal{U}$.*

If neither $\mathcal{A}_{loc_1}$ nor $\mathcal{A}_{loc_2}$ is present in $\mathcal{U}$, then the environment is said to be basic.

Otherwise, the environment is extended.

[5] We are aware that devices (i.e., actual implementation of algorithmic specifications) may be able to distinguish a holder (e.g., via a fingerprint reader). However, we deliberately choose to have a strong adversary model, in which an adversary is able to operate such a device without restrictions.

[6] In line with the threats defined in [41] and the adversary model of [22].

[7] This is known as overshadowing, see [41] for more details

Figure 3 illustrates an example of extended environment. As mentioned in the previous sections, we restrict ourselves to the parties that are relevant to a given security property (defined in Sec. 4.2) when building the corresponding extended environment. For instance, in the definition of the secret extraction resistance property, one of the two locations is empty. And, for our generalised distance fraud, during the attack phase, loc_1 contains no adversary.

Also note that the execution environment is modulo an apriorily fixed universe of locations: if we change the universe of locations, we produce a different environment.

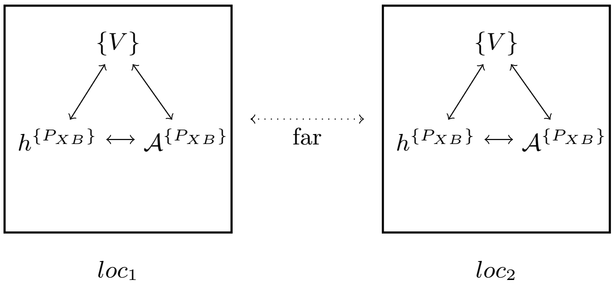

Fig. 3. An example of extended execution environment. h denotes a honest holder, $\mathcal{A}$ is the adversary, $\{V\}$ is a set of verifiers, and $\{P_{XB}\}$ is a set of provers which are either black-box or white-box.

Concurrency. Def. 7 allows for concurrent executions of the protocol. More specifically, as in other formal security models such as Bellare-Rogaway formalisms [9], we allow devices to run their algorithm several times, in interleaved fashion: one execution may start while another one is still in progress. We additionally allow several executions of one prover-device to happen concurrently, with one or several verifier-devices. Hence, in this concurrency model, we can have several prover-sessions from multiple prover-devices and several verifier sessions from multiple verifier-devices, interleaved in any possible way.

Sessions. As usual, we call each such execution a *session*: if one execution is run on a prover-device then it is a *prover session*; if it is run on a verifier-device then it is *verifier session*. A session is *full* if its transcript contains the last message of the ITM specification; otherwise, it is *partial*.

The chronologically-ordered list of the messages sent and received in a session is called the *transcript* of the session.
A session can be identified pseudo-uniquely (e.g., via the application of the pseudorandom function to the transcript). As such, we write X^i for the *i-th session of a party* X.

In an legitimate execution, each prover session has a corresponding verifier session, and vice versa. To denote this, we write verifier-sessions as (V^i, P), and prover-sessions as (V, P^j). As per Bellare-Rogaway models [9], we call two corresponding sessions (V^i, P) and (V, P^j) *partnered sessions*. If the transcript of a verifier-session V^i and a prover-session P^j are the same, then V^i and P^j have *matching conversations*.

Master Sessions. Adversaries may interfere in a session, such that the conversations are not matching anymore.

Definition 8. ***Master Sessions.*** *We say that the adversary $\mathcal{A}$* disrupts partnering *(over a session (V^i, P)) if $\mathcal{A}$ is involved in an extended execution environment such that there is a verifier-session (V^i, P) but this session V^i has no partner (V, P^j) with which it is engaged in a matching conversation.*

A master session *is a set of sessions needed for the adversary to disrupt partnering. We consider that master sessions are also identifiable uniquely and we use mid to denote master-session identifiers. Also, the chronologically-ordered list of all messages in the master session gives the* transcript of the master session*.*

Several master sessions may lead to the same disruption of partnering. In this case, if we need to nominate one such master session, we randomly choose one of several possible for the same disruption.

Note that, unlike in classical Bellare-Rogaway models, Def. 8 does not require that (V^i, P) output 1 for a disrupted partnering. By Def. 8, we only mean to gather in a master-session all the sessions that the attacker interleaves in order to mount his attack.

Notation. If we mean to be specific that it is a verifier-party with id m and a prover-party with id k that are communicating, then instead of (V^i, P) and (V, P^j) we write: $({V_m}^i, P_k)$ and $(V_m, {P_k}^j)$ to clearly mean that the i-th session of the verifier-party V_m is partnered with the j-th session of the prover-party P_k. We try to omit the party identifiers for readability purposes.

2.4 OracleDB: Game-based Model for DB

We gave the high level description of the adversary capabilities. We now provide a game-based security model to define these capacities formally.

The Challenger. The challenger generates an execution environment, the properties of which depend on the security game. For instance, in a strong generalised mafia fraud game, the challenger does not need to add any white-box prover to the environment. After generating an execution environment, he provides oracles to the adversary, as an interface to interact with the environment. Whenever an adversary wants to interact with a party, the challenger simulates it. The challenger keeps track of all sessions and master sessions.

Definition 9. ***The Challenger.*** *Let Π be a specification of a DB protocol. A challenger* Ch *is a ppt. algorithm, which does the following. The challenger Ch picks an arbitrarily fixed number of appropriate universes of locations $\mathcal{U}$=(loc_1, loc_2). The challenger Ch runs the Setup algorithm multiple times to realise Π. Then, for each appropriate universe $\mathcal{U}$, the challenger Ch creates an execution environment for $\Pi_{\mathcal{U}}^{real}$ at the universe of locations $\mathcal{U}$.*

Each party involved in each $\Pi_{\mathcal{U}}^{real}$ has a status*, which denotes if it is active or not. When a prover or verifier is inactive, it ignores all incoming messages. Initially, all are inactive. If one is active, then it means it executes at least one session.*

For each $\Pi_{\mathcal{U}}^{real}$, the following holds:

A. *the challenger Ch maintains a* prover-list PL *of (the polynomial number of) prover-parties present in $\Pi_{\mathcal{U}}^{real}$.*

1. The prover-list PL contains tuples of the form $(\mathtt{P_k.id}, \mathtt{xb}, \mathtt{status}, \{\mathtt{P_k^j.sid}\}_{\mathtt{j \in sessions}}, \mathtt{secret})$ *where: $P_k.id$ is the identifier of a prover-party P_k; $xb \in \{WB, BB\}$ denoting if the prover-party is white-box or black-box; status is either inactive or active; $\{P_k^j.sid\}_{j \in sessions}$ is the set of session identifiers for all the sessions j that the prover-party P_k are running by/at this time*[8]*; if xb=WB, then secret is formed of all the identifiable, inner material of the prover; if xb=BB, then secret is null.*

2. The prover-list PL is indexed over $P.id$.

[8] This set is the empty set is the prover-party is inactive.

B. *the challenger Ch maintains a* verifier-list VL *of (the polynomial number of) verifiers present in $\Pi_{\mathcal{U}}^{real}$. The verifier-list VL contains tuples of the form* $(\mathtt{V_m.id}, \mathtt{status}, \{\mathtt{V^i_m.sid}\}_{\mathtt{i\in sessions}})$, *where: $V_m.id$ is the identifier of a verifier-party V_m; status is either inactive or active; $\{V^i_m.sid\}_{i\in sessions}$ is the set of session identifiers for all the sessions i that the verifier-party V_m is running by/at this time*[9].

We recall that $X.id$ and X both denote the uniquely identifiable party X, with the first just being more specific than the second. The same is the case for $X^i.sid$ vs. X^i, w.r.t. to the ith session of the party X. To simplify notations, from here on, we do not always index parties: i.e., instead of writing V_m and P_k, we just write a generic V and P meaning any arbitrary verifier party V or prover party P.

C. *the challenger Ch stores a* list *$masterL$* of master-sessions *present in the $\Pi_{\mathcal{U}}^{real}$ environment, with $masterL$ containing tuples* $(\mathtt{E}, \mathtt{mid}, \mathtt{t}, \mathtt{out}, \mathtt{E'})$, *where: E, E' are prover-sessions or verifier-sessions such that $E, E' \in \cup_{i,j}\{(V,P^j),(V^i,P) \,|\, (V,P^j)$ partnered with $(V^i,P)\}$ and $E \neq E'$; mid is the identifier of the master-session in which E and E' are involved and the whole master session is kept in the list; t is the corresponding transcript of the master-session; out is the result of the protocol as seen by E: -1 for unfinished protocol, 0 for failure, 1 for success. This list is in sync with the PL and VL lists.*

D. *for the universe $\mathcal{U}$=(loc_1, loc_2) determining the execution environment $\Pi_{\mathcal{U}}^{real}$, the challenger Ch creates a* locations' list *$LOCL$, which indicates the position of each prover, verifier and adversary entity. It is defined as* $(\mathtt{h}^{\{\mathtt{P}\}}_{\mathtt{loc_1}}, \mathcal{A}^{\{\mathtt{P}\}}_{\mathtt{loc_1}}, \{\mathtt{V_{loc_1}}\}, \mathtt{h}^{\{\mathtt{P}\}}_{\mathtt{loc_2}}, \mathcal{A}^{\{\mathtt{P}\}}_{\mathtt{loc_2}}, \{\mathtt{V_{loc_2}}\})$, *where $h^{\{P\}}_{loc_1}, \mathcal{A}^{\{P\}}_{loc_1}, \{V_{loc_1}\}$ respectively denote the set of provers held by honest holders at location loc_1, the set of prover-parties held by the adversary at location loc_1, the set of verifiers found at location loc_1, and the same respectively for location loc_2. These sets might be empty.*

Adversary vs. Challenger: The Oracles Accessible to $\mathcal{A}$. The adversary has access to the prover-list PL, the verifier-list VL and the sessions' list $masterL$ in that he can read them, but $\mathcal{A}$ cannot modify them directly. He can however inflict modifications on them via oracles. These oracles provided to $\mathcal{A}$ are defined below.

init(P, V): If both P and V are inactive, this sets the status of P and V to active. It then opens a new session running as (V, P^j) and (V^i, P). Either P or V can be omitted, in which case only one record is added to masterL. Otherwise, 2 records are added to masterL: one with $E = V^i$, and one with $E = P^j$. Once the session is created, it is recorded in the relevant lists PL, VL, masterL. The session identifier is given to $\mathcal{A}$ slightly before the session actually starts, so that $\mathcal{A}$ can use it directly with the oracles that take a session identifier as input.

term(P_k^i): If the prover-party P_k is inactive, then this oracle has no effect. If both P_k is active, then this oracle terminates the i-th sessions of this prover-party. The call deletes the corresponding entry from PL, for the i-th session $P^i_k.sid$ of the prover-party P_k. If needed, then it adjusts the $masterL$ list to state, e.g., the outcome of the protocol as viewed by the i-th session P^i_k of the prover-party P_k within the master-session that P^i_k may be involved in, etc.

term(V_m^j): If the verifier-party V_m are inactive, then this oracle has no effect. If V_m is active, then this oracle terminates the j-th sessions of this verifier-party. The call deletes the corresponding entry from VL , for the j-th session $V^j_m.sid$ of the verifier-party V_m. If the case, then it adjusts the $masterL$ list, e.g., to set the outcome of the protocol as viewed by the j-th session V^j_m, etc.

term(P_k^i, V_m^j): This oracle combines the two oracles above, i.e., both the i-th session $P^i_k.sid$

[9] This set is the empty set is the verifier-party is inactive.

of the prover-party P_k and the j-th session V_m^j of the verifier-party V_m are terminated at once.

attach(P, h): If the status of P is inactive, then it attaches the prover-party P to a holder h, where h is either a honest holder or an adversary, in loc_1or loc_2 of some universe U used in the lists $LOCL$. The list $LOCL$ is updated as such. This encodes moving P to a specific location.

send($\mathcal{A}, E^i, m$): Via this oracle, the adversary $\mathcal{A} \in \{\mathcal{A}_{loc_1}, \mathcal{A}_{loc_2}\}$ sends the message m to the session i of party E. If the party E is not specified, the message is broadcast *i.e.*, sent to all parties and all their sessions.

block($E^j, \mathcal{B}, \{S^k \mid S$ party $\}$): Let $M = M_0 \ldots M_k$ denote all the bits sent by the party E during a honest protocol execution. The oracle checks if E is active by looking up in the PL, VL lists and if it has one running session E^j. If it does not, the oracle aborts. If it does, then the oracle blocks the transmission of bits $\{M_i | i \in \mathcal{B}\}$, in such a way that only the parties S (and their sessions $\{S^k\}$) receive it.

result(sid): If there exists session sid included in a master session mid inside a tuple $(V, mid, t, out, P) \in masterL$ with V being a verifier-party, then this oracle returns out.

ident(sid): If there exists session sid included in a master session mid inside a tuple $(V, mid, t, out, P) \in masterL$ with V being a verifier-party, then this oracle returns P. Otherwise, it returns error.

Note that if these oracles are called on prover (resp. verifier) identities that do not belong to PL (resp. VL), then the calls do nothing. The last two oracles are not useful to the adversary in practice, but allow to define the security goals in a more convenient way. In some of the oracles, some parameters are optional; this is implicit from their description.

3 Real-World Terrorist Frauds via OracleDB

In Sec. 2, we put forward a much closer to real-world corruption model for prover-devices whereby they can be white-box or black-box. We now show this solves the controversial problems of formal analysis of terrorist-frauds in DB. The take-away message of the section is that **TF-analysis needs not be considered in the OracleDB model, or in any formal DB-security model**. Sec. 3.1 discusses our corruption model w.r.t. a specific class of DB protocols. Sec. 3.2 gives the main result: in real-life cases, TF attacks are unavoidable.

3.1 White/Black-box Identification

In our model (Def. 1), the algorithms of the provers *P-alg* make use cryptographic identification mechanisms (called *idents*) which are left under-specified: *idents* can be for instance cryptographic keys, physically unclonable functions[10] (PUF). The question naturally arising is how to treat PUFs with regards to the white-box/black-box corruption model.

PUFs in the White-box and Black-box Corruption Models. In the DB literature, protocols that use PUFs [33,30] use their uncloneability, and implicitly consider the provers using PUFs as black-box. This treatment is however unfair to DB protocols where cryptographic keys are used, instead of PUFs, for prover-authentication purposes; in this case, the default option is to consider these provers as white-box. So, the current state of affairs it to compare the security of protocols using prover-devices considered fully-corruptible (for which the key is known to the adversary) with protocols where prover-devices that are considered to behave honestly. However, security models in which the PUF are corrupted exist. In particular, in the *"simulatable PUF"* [43] model, the PUF can be maliciously replaced by a PRF, for instance by

[10] Recall that there are two DB protocols [33,30] that use PUFs as the prover's identification mechanisms.

the manufacturer. It remains indistinguishable from an actual PUF for everyone who does not know the key to the PRF. However, the user who corrupted the device has access to the key, and can therefore simulate, or even clone, the resulting function. The notion of simulatable PUF fits the white-box corruption model, and we believe that protocols using PUFs should be analysed in this model, rather than in a black-box manner.

Therefore, in the rest of this section, in the white-box case, we operate in the simulatable PUF model where corrupted, white-box provers can simulate the PUF's behaviour via a PRF. In the black-box case, a PUF is an non-simulatable, honest PUF.

3.2 Terrorist Frauds with Real-World Provers

In DB, a man-in-the-middle attacker $\mathcal{A}_{MiM}$ has the goal to authenticate as a legitimate, far-away prover-device P in a session *sid*, without knowing a-priori the authentication details (i.e., *idents*) of P. A terrorist-fraud (TF) attacker $\mathcal{A}_{TF}$ has the same goal as a MiM attacker $\mathcal{A}_{MiM}$, but $\mathcal{A}_{TF}$ is allowed to be helped by P in the session *sid*, as long as this help does not permit $\mathcal{A}_{TF}$ to authenticate in future sessions unaided. Section 4 will further formalise the MiM notion in our model. However, Section 4 will not formalise further the TF notion, as we argue in this section that the TF notion is unnecessary.

To be able to show that a formal TF notion and analysis is not necessary, we now give a high-level formulation of terrorist-fraud resistance, which is in line with traditional definitions of this type. Again, this is not a formalisation of TF in our formal model (even though, for continuity, it uses terminology we introduced in previous section). It is a standard expression of TF taken which can be cast in any commonplace model for DB, whereby one has encodings of time, distance, and accomplices to mount such collusion-based attacks over proximity measurements.

General Acceptation in DB. ***TF-resistance.*** *Let Π be a specification of a DB protocol and Π^{real} be its realisation.*

We say that a prover-device P in Π^{real} helps *an adversary if any ppt. algorithm is allowed to interact with P and give its outputs to $\mathcal{A}$. Any prover-device P who helps is called a* TF-prover*.*

We say that Π^{real} is TF-resistant *if the following holds: for any white-box or black-box prover-device P found far-away*[11] *from a given verifier-party V, for any ppt. adversary $\mathcal{A}$, if P helps $\mathcal{A}$, and makes the verifier-party V output* 1 *in a master-session (in which V believes to have a partnered session with a session of P's), with overwhelming probability, then there exist an adversary (who can use $\mathcal{A}$'s knowledge) and make the verifier V output* 1 *without P's help or the inadvertent help of any other prover, in another master-session mid' appearing after the master-session mid in the masterL list (in which V believes to have a partnered session is an execution of P), with overwhelming probability.*

The tuple $(P, \mathcal{A})$ is called a TF-attacker*. We say that a TF-attacker* succeeds *if Π^{real} is not TF-resistant. Otherwise, we say that all TF-attackers* fail*.*

Note that in the above formulation, we stipulated "without P's help or the inadvertent help of any other prover". This means that in the second run of the TF-attack, when the attacker $\mathcal{A}$ is trying to impersonate the prover a second time, no prover can take part unwillingly in some form of MiM mounted by $\mathcal{A}$. This is restriction compared to existing work, where in the second attempt of impersonation by $\mathcal{A}$, provers can be unknowingly/unwillingly take part. However, this restriction is only in place for the sake of an easy-to-understand proof

[11] I.e., P and V are at different locations in a universe of location used in the execution environment.

for Proposition 10. In fact, as the rest of the section will show, our results do hold for the extended case where honest provers can be present and unknowingly/unwillingly take part in subsequent attempts by $\mathcal{A}$ to impersonate the initially collusive prover P.

No Terrorist Frauds with Black-box Provers We prove that if we consider a DB threat-model where prover-devices are black-box, then TF-resistance is an unnecessary security property.

Indeed, if provers P are black-box, then all the help they may give to adversaries is nothing beyond a normal protocol execution and so it is futile, i.e., if P is afar, the help cannot authenticate. We state this formally in Proposition 20 which we also prove in Appendix B.

Another way of stating the above result is this. An attacker can hold a far-away black-box prover and communicate with a second adversarial device found near the verifier. This MiM attacker is clearly equivalent with a terrorist-fraud where the TF-prover is black-box. We state this formally in Lemma 21 which we also prove in Appendix B.

So, from the above, if the provers are black-box, then the following is the case:

— terrorist-fraud resistance is conceptually irrelevant (as a malicious prover cannot help an accomplice);

— the security notion of terrorist-fraud resistance is shown to be redundant, as it is a subset of MiM attacks (which constitute a more generic notion).

Always Terrorist Frauds with White-box Provers If white-box provers can be copied into black-box provers, then a terrorist-fraud attack is always possible, with the exception of some uninteresting cases. These uninteresting cases are:

(a) if protocols identify the holders of devices (which we already argued as a measure that counteracts relaying per se and as such has no place in DB);

(b) if protocols are not MiM-resistant (which is un-interesting as that would make the protocols already widely insecure, so the point of being TF-resistant or not becomes mute). So, **if white-box provers can be copied into black-box provers, then a protocol which is widely accepted as secure w.r.t. authentication (against MiM) will never be TF-resistant**. This is formally stated in Lemma 10.

Proposition 10. ***[TF-attacks with White-box Provers.]*** *Let Π be a specification of a DB protocol and Π^{real} be its realisation in an execution environment. Assume Π^{real} is resistant to MiM attacks.*

If white-box provers can be copied into black-box provers, then Π^{real} is not TF-resistant.

Synonymously, if WB provers can be copied into BB provers, then there exists a TF-attacker $(P, \mathcal{A})$ which succeeds against Π^{real} with P being a white-box TF-prover.

The proof of Proposition 10 is non-technical and exhibits such a terrorist-fraud attack which is always possible against a MiM-secure protocol if the TF-prover is white-box.

Proof. The TF-attacker $(P, \mathcal{A})$ is as follows: a device terrorist–device is produced (e.g., by P's holder or by $\mathcal{A}$) as a black-box copy of P. The terrorist–device is programmed to self-destruct (*i.e.*, wipe its memory) after a successful authentication. The help $\mathcal{B}$ by P to $\mathcal{A}$ is the terrorist–device itself (which is passed to $\mathcal{A}$ e.g., by P's holder, at any point before the attack-session mid). Then, $\mathcal{A}$ (or the part of $\mathcal{A}$ that is close to V) executes $P\text{-}alg$ via terrorist–device to authenticate in session mid. Since terrorist–device is black-box, $\mathcal{A}$ learns nothing other than in a honest session of the protocol. Since terrorist–device wipes its memory after one use, $\mathcal{A}$ cannot use it to authenticate in a post-help session mid'. $\square$

We call the attack in the proof of Lemma 10 *white-box terrorist-fraud (WB-TF)* . **The WB-TF attack works on all protocols of the literature**. WB-TF circumvents the formal models, such as [13,23], as those did not consider a black-box cloned terrorist–device as a means towards a TF-attack.

White-box Terrorist-frauds. WB-TF differs from traditional terrorist frauds, in the sense that the accomplice can authenticate at any time whilst he holds the terrorist–device and the terrorist–device has not wiped itself. Instead, traditionally TF attacks are performed online: the accomplice needs to be able to query the far away prover during the authentication. However, this behaviour can be emulated with a WB-TF by integrating a remote activation/deactivation mechanism to the terrorist–device, so that it only works when the holder intends it to. Adding a remote activation mechanism to the terrorist–device even permits to perform more advanced types of terrorist frauds. For instance, the holder can permit his accomplice to impersonate him at will, but only during certain time periods, by removing the self-destruction mechanism, and activating the terrorist–device only during the desired time-slots. Such control over the actions of the accomplice could be particularly relevant, for instance, if the holder of the prover-device P wants to delegate his access to a facility to his accomplice $\mathcal{A}$ only when he is not present. Note that since the accomplice $\mathcal{A}$ only observes a protocol execution every time he uses the terrorist–device, as long as the protocol is resistant to MiM attacks, $\mathcal{A}$ never becomes able to impersonate the prover-device P when the terrorist–device is not active.

WB-TF was presented w.r.t. our definition of terrorist-fraud resistance, which requires just one successful authentication outside of the the helped session. However, WB-TF can be made more generic as follows. Instead of self-destructing after one authentication, the terrorist–device can be programmed to self-destruct after the k^{th} successful authentication. This modification accommodates the definition for terrorist fraud given in [13], whereby the adversary is helped k times instead of just one.

Finally, WB-TF can be adapted, as the terrorist–device can embed any algorithm. The terrorist–device does not necessarily need to be a copy of the terrorist-device P as per the original description of WB-TF. Actually, whenever there exists any terrorist fraud against a protocol in previous security models, it can be emulated with such a slightly-adapted WB-TF:

Proposition 11. *Let Π be a specification of a DB protocol and Π^{real} be its realisation in an execution environment. Assume Π^{real} is resistant to MiM attacks, and that white-box provers can be copied into black-box provers. If Π is vulnerable to a terrorist-fraud attack in any existing formal model, then there is a white-box terrorist-fraud against. Π^{real}.*

Proof. Let $\mathcal{B}$ be the ppt. algorithm terrorist-fraud attack in another formal model.

Let terrorist–device be a black-box device which is produced as in Proposition 10 but now contains this new ppt. $\mathcal{B}$. By the fact that $\mathcal{B}$ amounts in a valid TF-attack, when $\mathcal{A}$ uses terrorist–device to authenticate, $\mathcal{A}$ does not gain a significant advantage for latter authentications. The rest stays the same as in Proposition 10. □

So, this section formally proves that:

– **in the black-box setting, there is no TF attack that is relevant;**

– **in the white-box setting, if the protocol is MiM-resistant, then there is no way to protect against TF-attacks.**

As such, we argue that the TF analyses matter is closed and formal models, including ours, should discard the analysis of this threat.

Kilinc and Vaudenay [31] also looked at the equivalence between terrorist fraud and MiM attacks in the black-box corruption-model. However, they interpreted their conclusion differently to us. They do not deem TF irrelevant, instead they consider it as the strongest possible threat (in the black-box corruption-model). Due to the fact that MiM-security and TF-security are equivalent in the black-box corruption-model, our viewpoint and their are synonymous, as far as far models are concerned. However, our additional observations about the irrelevance of TF-resistance in the white-box model leads us to militate that TF-resistance ought to be scrapped as a DB security requirement.

4 DB Security Requirements in OracleDB

4.1 Threat Model: High-level Descriptions

We now study which security goals should be achieved when the provers are black-box vs. white-box.

Corruption Models. We distinguish two corruption modes: **strong corruption**, and **weak corruption**. In the strong corruption case, all provers (except, depending on the attack type, for the attacked prover) are white-box. This corresponds, for instance, to a situation where an adversary registers corrupted devices into the system.

In the weak corruption case, only the attacked prover may (or may not, depending on the attack type) be white-box, whereas the other provers are all black-box. This corresponds, for instance, to an adversary performing an expensive side-channel attack on one prover to recover its secret material, but not willing to do the same for other devices.

Our formalisation only considers 3 types of attacks: **secret-extraction**, **generalised mafia fraud** (GMF), and **generalised distance fraud** (GDF).

Secret extraction. Secret extraction (Definition 12), is a new property for distance bounding, that concerns black-box provers. Black-box provers are meant to be temper-proof, and a protocol that leaks secret material violates this temper-proofness. The protocols in the literature that are vulnerable to secret extraction attacks, such as the ones descibed in [7], are typically also vulnerable to a MiM attack as a conseauence. On the other hand, secret extraction does not necessarily imply that the adversary can authenticate. For instance, if we add an additional long-term dummy key that is sent in clear at the beginning of a MiM-secure protocol, then it becomes vulnerable to a secret-extraction attack, as the protocol reveals a long-term secret. However, it is still MiM-resistant. We formalise this notion in Definition 12.

Generalised mafia-fraud (GMF). GMF (Definition 14), covers both MiM and distance-fraud/hijacking for black-box provers, and corresponds to the traditional MiM attack for white-box provers. Indeed, if the provers are black-box, then the adversary gains no additional knowledge by holding the prover. Hence, a distance fraud/hijacking adversary can be modeled as a far-away adversary holding a prover. In a mafia-fraud context, a second adversary is located near the verifier. Hence, a mafia-fraud adversary has more resources than a distance fraud/hijacking adversary, and is more general.

Since we exclude terrorist fraud resistance from our model, GMF generalises all classical attacks for black-box provers. However, for white box provers, we need an additional property, defined below.

Generalised distance-fraud. Adding white-box capabilities to the adversary allows him to perform more advanced attacks. In particular, in the white-box context, distance-fraud and distance-hijacking differ from mafia fraud, since the adversary can, for instance maliciously pick nonces, possibly depending on the secret key (for instance, to mount PRF programming attacks [12]). Hence, in the white-box case, we need to consider another property: **generalised distance-fraud** (Definition 13).

For white-box provers, generalised mafia-fraud and generalised distance-fraud are enough to cover all the usual security properties, except for terrorist fraud, which we need not to consider (as per Section 3).

4.2 Threat Model: Formal Definitions

By using security games $\mathcal{G}$, we now formally define the DB-security properties informally introduced above.

Definition 12. Secret-extraction. *Let Γ^{real} be the realisation of a DB protocol Γ in an appropriate universe of locations, such that in one location there is the an adversary holding provers and a set of verifiers, and there is no one in the second location. Let $\mathcal{P}$ be a set of prover-devices in Γ^{real} which have idents being cryptographic keys. Let $\mathcal{G}$ be a security game against a protocol Γ in which the challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ access to all the oracles. The game $\mathcal{G}$ is a* secret-extraction game*, if the play is as follows: 1. $\mathcal{A}$ choses a prover $P \in \mathcal{P}$; 2. $\mathcal{A}$ outputs a key-name k and a bitstring x.*

The winning condition *in $\mathcal{G}$ is that x is the correct value for the key k of the prover P. The* advantage of the adversary *$\mathcal{A}$ in winning $\mathcal{G}$ with probability α in the secret-extraction game is defined as $|\alpha - \frac{1}{2^{|x|}}|$. The protocol Γ is* secure against secret-extraction *if the advantage of an adversary $\mathcal{A}$ in the secret-extraction game is negligible in the security parameter defining Γ. If all provers are black-box, then the game is a* weak secret-extraction game*. Otherwise, it is a* strong secret-extraction game*.*

Note that this notion can be easily extended to *idents* which are PUFs or other types as well. However, for this manuscript, due to the protocols we analyse herein, we focus on *idents* which are keys.

Definition 13. Generalised Distance-fraud. *Let Γ^{real} be the realisation of a DB protocol Γ in an appropriate universe of locations $\mathcal{U} = (loc_1, loc_2)$ such that:*
– initially, loc_1 contains an adversary $\mathcal{A}_{loc_1}$, a designated verifier $d\mathbb{V}$, as well as other verifiers and a honest holder, and loc_2 contains a honest holder, an adversary $\mathcal{A}_{loc_2}$ and a set of verifiers.
– later, in what is called below the attack phase, $\mathcal{A}_l a$ is removed from loc_1.

Let $\mathcal{G}$ be a security game against a protocol Γ in which the challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ access to all the oracles. The game $\mathcal{G}$ is a generalised distance-fraud (GDF) game*, if the play is in the two phases below:*
***1. Learning phase.** 1. $\mathcal{A}$ outputs a prover identifier P, and a designated verifier identifier $d\mathbb{V}$, such that P is in loc_2 and $d\mathbb{V}$ is in loc_1;*
***2. Attack phase.** 1. The adversary $\mathcal{A}_{loc_1}$ is removed from loc_1: only $\mathcal{A}_{loc_2}$ is left in loc_2 2. $\mathcal{C}$ checks the location of P in $LOCL$. If it is close to $d\mathbb{V}$(i.e., the holder is h_{loc_1}), or if P is not in PL, then the game $\mathcal{G}$ is aborted. Similarly, if $d\mathbb{V}$ is not in V_{loc_1}, then the game is aborted.*

The winning condition *in an un-aborted generalised distance-fraud game $\mathcal{G}$ is as follows: in the attack phase, there must exist a session identified as sid such that $ident(sid) = P$ and $result(sid) = 1$ (as per $\mathcal{C}$ checking the records in $masterL$). Equivalently, $\mathcal{A}$ wins if he disrupts partnering over a verifier-session $(d\mathbb{V}^{sid}, P)$ and $result(sid) = 1$. The* advantage of an adversary $\mathcal{A}$ in the GDF game *is his success probability α. The protocol Γ is* secure against GDF *if the advantage of an adversary $\mathcal{A}$ in the generalised distance-fraud game is negligible* *in the security parameter defining Γ. If all provers but P are black-box, then the game is a* weak distance-fraud game. *Otherwise, it is a* strong distance-fraud game.

For this generalised distance-fraud property in the WB model, as per Definition 13 above, no adversarially-controlled entity are allowed near to the prover. Yet, honest holders are allowed near the verifier. Hence, our Definition 13 also covers the threat of distance-hijacking (DH) [20].

Definition 14. Generalised Mafia-fraud. *Let Γ^{real} be the realisation of a DB protocol Γ in an appropriate universe of locations $\mathcal{U} = (loc_1, loc_2)$ such that:*
– *loc_1 contains an adversary $\mathcal{A}_{loc_1}$, a designated verifier $d\mathbb{V}$, as well as other verifiers and a honest holder, and loc_2 contains a honest holder, an adversary $\mathcal{A}_{loc_2}$ and a set of verifiers.*

Let $\mathcal{G}$ be a DB security game against a protocol Γ in which the challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ access to all the oracles. The game $\mathcal{G}$ is a generalised mafia-fraud (GMF) game*, if the play is in two phases, as follows.*
1. Learning phase. *1. $\mathcal{A}$ outputs a prover identifier P, and a designated verifier $d\mathbb{V} \in V_{loc_1}$;*
2. Attack phase. *1. $\mathcal{A}$ loses access to the oracle attach, 2. $\mathcal{C}$ checks the characteristics of P in PL: if it is close to $d\mathbb{V}$ (*i.e.*, its holder is h_{loc_1} or $\mathcal{A}_{loc_1}$), or if it is white-box, then the game $\mathcal{G}$ is aborted.*

The winning condition *on an un-aborted generalised mafia-fraud game $\mathcal{G}$ is as follows: in the attack phase, there must exist a session identified as sid such that $ident(sid) = P$ and $result(sid) = 1$ (as per $\mathcal{C}$ checking the records in $masterL$). Equivalently, $\mathcal{A}$ wins if he disrupts partnering over a verifier-session $(d\mathbb{V}^{sid}, P)$ and $result(sid) = 1$. The* advantage of an adversary $\mathcal{A}$ in the gmf game *is his success probability α. The protocol Γ is* secure against generalised mafia-fraud *if the advantage of an adversary $\mathcal{A}$ in the generalised mafia-fraud game is negligible in the security parameter defining Γ. If all provers are black-box, then the game is a weak generalised mafia fraud game. Otherwise, it is a strong generalised mafia fraud game.*

In this game's formalisation in Definition 14, like in [14], prior to the execution of the attack phase whereby the fraudulent authentication is attempted, the adversary is given access to a *learning phase*. During this phase, he can place the target-prover close to the verifier. We mean that this learning phase is typically used to recover some secret material, by modifying messages during the challenge response part of the protocol, which is possible if both the prover and the adversary are close to the verifier.

5 Protocol Analyses in OracleDB

We now show first that the ubiquitous electronic payment protocol EMV (Europay, MasterCard and Visa) in this variant PayPass protocol with relay protection (which we simply call ***`PayPass`***) is vulnerable to proximity-based attacks. Then, we propose a simple modification of `PayPass` and prove it fully secure in our strongest corruption model.

The `PayPass` Protocol. The `PayPass` protocol with relay protection is an industry standard for relay-attack protection in contactless payments. It is used by Mastercard, and described in the EMV book C-2, Kernel 2 specification 2.5. It is depicted in figure 4, and a more complete high-level description can be found in [47].

Note that the correctness of the MAC is irrelevant to (the distance-bounding component of) the protocol, as the reader does not have the key to verify it, and this verification is performed by the bank.

<u>Note:</u> From here on, we use interchangeably the words: a). *card* and *prover*; b). *reader* and *verifier*.

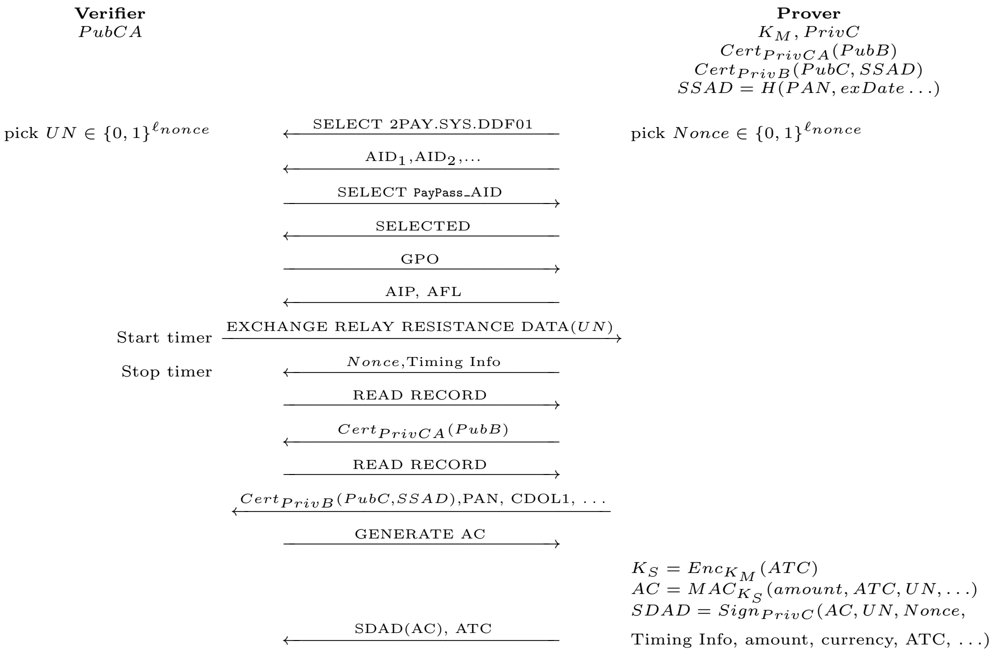

Fig. 4. Contactless PayPass with Relay Protection.

5.1 Insecurities in the Contactless Payments

While `PayPass` is secure against generalised mafia fraud, it fails to protect users against generalised distance fraud. However, distance frauds can be costly for the card issuer. Imagine a malicious user $\mathcal{A}$, mounting a distance fraud to pay in store A, and paying in a far away store B at the same time. $\mathcal{A}$ could claim that his card was hacked, because it appears to be in two locations at the same time, and ask for reimbursement of both his purchases to the bank. Since the contactless payment limit can sometimes be bypassed [24], the attacker could claim reimbursement for a very expensive goods. Such fraudulent operations can actually be prevented with minor modifications to the `PayPass` protocol.

The Attack on `PayPass` In the `PayPass` protocol with relay attack protection, the response of the prover in the timed phase is independent of the challenge. Hence, a malicious prover can send this response in advance, to meet the time bound [11]. To counter such attacks, a first step is to include UN in the response due by the prover, so that the malicious prover cannot send the response before receiving UN. However, this is not sufficient to prevent distance-hijacking style attacks. Indeed, a distant GDF adversary $\mathcal{A}$ can do the following:

– replace the messages that a card C found close to the reader R sends to the reader R with his own during the non-timed parts of the protocol;
– overwrite the nonce from C with his own, while not interfering with the UN part of the message;
– send the correct MAC/signature to the reader R (i.e., the one containing $\mathcal{A}$'s injected nonce, and $\mathcal{A}$'s SSAD).

In this way, $\mathcal{A}$ will be accepted by the reader. To prevent this type of attacks, we propose to tie UN with something identifying P, so that $\mathcal{A}$ cannot anymore claim the message as his own. To this end, we consider a (unique) identifier ID, with same length as UN.

5.2 Securing Contactless Payments

We now present our protocol PayPass+ which –with the modification briefly described above– becomes provably secure against generalised distance frauds.

The PayPass+ Protocol Our protocol is part-depicted in Fig. 5

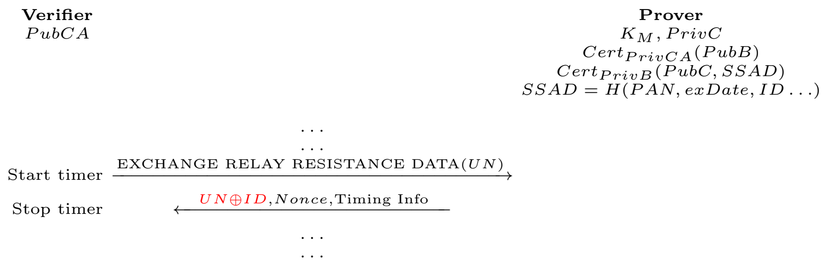

Fig. 5. The PayPass+ Distance-Bounding Protocol.

It differs from PayPass only in that, during the timed phase, the card responds with $(UN \oplus ID, Nonce, \text{Timing Info})$ instead of just $(Nonce, \text{Timing Info})$, i.e., $UN \oplus ID$ is added; this is marked with red on Figure 5. The ID bitstring is a public, pseudo-unique identifier written on the card and in its $SSAD$. Therefore, the card's certificate certifies this ID too. Moreover, the lengths of nonces be becomes a variable ℓ_{nonce}, with values depending on the security parameter; this is to be able to formally prove the security of our protocol.

Security Analysis of PayPass+ We now analyse the security of our protocol w.r.t. our security properties, given in Sec. 4.

In PayPass+, the prover has 2 cryptographic keys, K_M and $PrivC$, that are of interest for our secret-extraction definition introduced in Sec. 4. Each of them is used only once.

Encryption-driven security. We first define a *secret recovery (SR)* experiment for symmetric encryption. Informally, an encryption scheme is SR-secure if no polynomially bounded adversary can recover the secret key used for the encryption of messages of his choice. Any reasonable encryption scheme, with a reasonable key size, should be SR secure. Similarly to other security definitions for block ciphers, *e.g.* [8], security is not considered in an asymptotic fashion, since practical symmetric key schemes do not have a security parameter. However, we assume that the key size that is used in the protocol is chosen to be in line with the security parameter.

Definition 15. ***Secret recovery security.*** *Let E be a symmetric encryption scheme, for which we write $E_k(m)$ to denote that message m is encrypted with the key k. The* SR security experiment *is defined as follows. A challenger picks a random key K, and gives $\mathcal{A}$ an encryption oracle $EO(\cdots)$, such that $EO(m) = E_K(m)$ for any message m. The adversary outputs a bitstring K', and wins if $K' = K$. E is* SR-secure *if no practical adversary wins this game.*

Signature-driven security. In our security proofs, we need the signature scheme to be secure in a multi-user setting, since we allow for several provers with different keys. Hence, we use the multi-user security definition for digital signatures given in [38]. In particular, we use the

GMR-SKS security notion. In this security model, the adversary is given access to a signature oracle S corresponding to a public key y. This oracle takes as input a message m, and returns a signature s, such that s is valid with regards to m and y. A signature scheme is *GMR-SKS secure* if no polynomially bounded adversary can output a triple $t = (y', m', s')$ such that the signature s' is valid for the message m' and the public key y', and either of the two following conditions hold: (1) m was not sent to S, $y' = y$ and s is correct for m' and y', or (2) m' has been sent to S, $s' = s$, $y' \neq y$, and s' is correct for y' and m'.

Theorem 16. ***Secret-Extraction Security.*** *Let S and E respectively denote the signature and encryption scheme used in `PayPass`. If the E is SR secure, and S is GMR-SKS secure, then `PayPass+` is strong secret-extraction secure.*

Proof. We start with the key K_M. Assume the adversary $\mathcal{A}$ wins the secret extraction game for K_M. Then, we can use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the SR experiment. The construction is as follows: for each prover, $\mathcal{B}$ starts a new SR experiment, uses the corresponding encryption oracle EO to produce $K_S = E_.(ATC)$. When the adversary $\mathcal{A}$, placed in the environment simulated by $\mathcal{B}$, outputs a value for K_M, $\mathcal{B}$ returns this value to the corresponding SR challenger. Hence, its success probability in the SR experiment $\frac{p^{\mathcal{A}}}{q_p}$, where $p^{\mathcal{A}}$ is the success probability of $\mathcal{A}$, and q_p is the (polynomial) number of provers. Therefore, if $p^{\mathcal{A}}$ is non negligible, then $\mathcal{B}$ breaks the SR security of E.

We now prove the security for the security for $PrivC$. Assume $\mathcal{A}$ wins the secret extraction game for $PrivC$. Then we can use $\mathcal{A}$ to to build an adversary $\mathcal{B}$ that wins the forgery game against the signature scheme. First, the $\mathcal{B}$ creates a new forgery experiment for each prover, and uses the corresponding signing oracles to generate the messages $SDAD$. When $\mathcal{A}$ outputs a value $PrivC'$, $\mathcal{B}$ picks a random message m (which was not previously queried to the oracle), computes $\sigma = S_{PrivC'}(m)$, and returns (m, σ) to the forgery challenger. If $\mathcal{A}$ recovered the correct signature key, then (y, m, σ) (where y is the public key of P) is a valid forgery. Hence, the success probability in the GMR-SKS experiment is $\frac{p^{\mathcal{A}}}{q_p}$. □

Theorem 17. ***Generalised Mafia-Fraud Security.*** *Let S be the signature scheme used in `PayPass`. If S is GMR-SKS secure, then `PayPass+` is strong generalised mafia-fraud secure.*

Proof. This is a game-based proof [44]

G_1: This game is the initial game G_0, where no $Nonce$ value is indeed used more than once by any prover.

Let qn be the number of $Nonce$ values issued by provers, during the experiment encapsulating this game. The probability that one $Nonce$ repeats is upper bounded by $\frac{qn^2}{2^{\ell_{nonce}}}$. G_0 and G_1 are identical except for the failure event that two identical $Nonce$ values are used, so we have $Pr[G_1] - Pr[G_0] \leq \frac{qn^2}{2^{\ell_{nonce}}}$, which is negligible.

G_2: This game is the game G_1, where no UN value is indeed used more than once by any verifier.

Let qun be the number of UN values issued by provers, during the experiment encapsulating this game. The probability that one UN repeats is upper bounded by $\frac{qun^2}{2^{\ell_{nonce}}}$. G_1 and G_2 are identical except for the failure event that two identical UN values are used, so we have $Pr[G_2] - Pr[G_1] \leq \frac{qun^2}{2^{\ell_{nonce}}}$, which is negligible.

G_3: This game is the same as G_2, except that the verfier never sends a value UN that has previously been sent by an adversary through the *send* oracle. Additionally, the experiment is aborted if a prover located in a different location than the verifier who sent a given value UN receives it before a time corresponding to $\frac{\mathbb{B}}{2}$.

The aim of this transition is to eliminate the failure event E_{guess} that $\mathcal{A}$ randomly guesses a value UN in advance. Let qs denote the (polynomial) number of calls to the *send* oracle, and qv the (polynomial) number of executions of the verifier algorithm: we have $Pr[E_{guess}] \leq \frac{qs \cdot qn}{2^{\ell_{nonce}}}$. Hence, $Pr[G_3] - Pr[G_2] \leq \frac{qs \cdot qn}{2^{\ell_{nonce}}}$, which is negligible.

We now prove that the success probability of $\mathcal{A}$ in G_3 is negligible. Remark that in G_3, $\mathcal{A}$ cannot relay UN to a distant prover and receive its response and the corresponding $Nonce$ in time to be accepted by $d\mathbb{V}$. Hence, either $\mathcal{A}$ sends a random UN' in advance to obtain $Nonce$ in time, or it receives UN and replies with a random $Nonce$. In the first case, we have $UN \neq UN'$, due to the transition of G_2. In the second case, the probability that $\mathcal{A}$'s guess is correct is negligible ($\frac{qv}{2^{\ell_{nonce}}}$). Hence, except with negligible probability, the signature $SDAD$ provided by the attacked prover is not related to both UN and $Nonce$. However, to be accepted by $d\mathbb{V}$, the signature needs to be correct. Hence, the authentication is only accepted if $\mathcal{A}$ forges a correct signature on $(Nonce, UN, AC, \ldots)$, corresponding to the public key of the attacked prover, which contradicts the hypothesis on the signature scheme. $\square$

For distance-fraud resistance, the ID values need to be *pseudo-unique*, *i.e.*, generated in such a way that any two IDs have a high hamming distance.

Definition 18. ***Pseudo-unique identifiers.*** *Let s be a security parameter. A set of identifiers $\mathcal{I}$ is set to have the pseudo-unique property if, for any pair of identifiers $(a, b) \in \mathcal{I}^2$, such that $d = HD(a, b)$ (where HD is the hamming distance), it holds that 2^{-d} is negligible in s.*

Pseudo-unique identifiers can be instantiated with Reed-Solomon codes [42]: any two codewords have a minimum hamming distance $d = n - k - 1$, where n is the length of the codeword, and k is the length of the message (identifiers). If we fix k to an arbitrarily large constant, and n varying with the security parameter, then 2^{-d} is negligible, which satisfies the definition.

Theorem 19. ***Generalised Distance-Fraud Security.*** *If the UN values sent by the reader are random and uniformly distributed, and the identifiers ID are pseudo unique, then `PayPass+` has strong generalised distance-fraud security.*

Proof. The value $UN \oplus ID$ uniquely identifies the prover running the protocol for a given UN. In a GDF, $\mathcal{A}$ is at a distance greater than $\mathbb{B}$, so that if he waits until he receives UN to send a response, then the elapsed time will be larger than $2 \cdot \mathbb{B}$, and $d\mathbb{V}$ will reject $\mathcal{A}$. Hence, to be accepted, the response needs to either (1) be sent in advance by $\mathcal{A}$ or (2) be sent by a closeby prover P, or (3) be a composition of a message from P and a message from $\mathcal{A}$. Let qv denote the number of verifier sessions started during the attack phase. In case (1), $\mathcal{A}$ needs to guess UN for his response to be correct, which succeeds with a probability upper bounded by $\frac{qv}{2^{\ell_{nonce}}}$. In case (2), the response of P is $UN' \oplus ID_P$, where UN' is sent by either a verifier or $\mathcal{A}$. Let qp denote the number of prover sessions started by $\mathcal{A}$ during the attack phase. The probability for a random UN' to satisfy $UN' \oplus ID_P = UN \oplus ID_{\mathcal{A}}$, where UN is sent by $d\mathbb{V}$, is upper bounded by $\frac{nb_P \cdot nv}{2^{\ell_{nonce}}}$, where nb_P is the total number of prover IDs. Finally, for case 3, $\mathcal{A}$ could overwrite parts of the response from a closeby prover P: since $\mathcal{A}$ knows the ID values, he knows which bits of the $ID_P \oplus UN$ differ from the corresponding ones in $ID_{\mathcal{A}} \oplus UN$. Hence, $\mathcal{A}$ only needs to guess the send these bits and can let P send the other ones. He therefore has x bits to guess, where $x = HD(ID_{\mathcal{A}}, ID_P)$. Due to the pseudo unique property of the identifiers, 2^{-x} is negligible, so the probability for $\mathcal{A}$ to properly guess these x bits is negligible. $\square$

6 Generic Distance-Fraud Attacks

In this section, we introduce a generalised distance fraud (similar to a distance-hijacking) that works on most symmetric-key distance bounding protocols using a PRF. Our attack is similar in nature to attacks presented in symbolic-verification formalisms [22,36], but uses the notion of "programmable PRF" [12], and is therefore applicable to more protocols.

6.1 PRF Programming

"Programmable PRFs" [12], which are at the basis of our attack, underline a loophole in the security claims of many distance bounding protocols: the security property for a PRF is that it behaves randomly to someone who does not know the key, but not to someone who knows the key. In other words, there exist functions that are secure PRFs, but that contain trapdoors, *i.e.*, input values derived from the key, and for which the output is not random. Dishonest provers do know their keys, so they can exploit programmable PRFs. Suchs trapdoors can be implemented, for instance by the manufacturer of the device implementing it. We define the programmed PRF PPRF that we use for our attack. Let f_x be a PRF keyed with a key x, and R be a constant.

$$\mathsf{PPRF}_x(NP, NV) = \begin{cases} R \text{ if } NP = g(x) \\ R \text{ if } NV = h(x) \\ f_x(NP, NV) \text{ otherwise,} \end{cases}$$

where g and h are arbitrary functions functions from $\{0,1\}^{|x|}$ to $\{0,1\}^{|nonce|}$. For clarity, we use $g(x) = h(x) = x$ but other functions could be used, for instance hash functions. The function PPRF is a PRF: for a PRF adversary, guessing a value that triggers a non-random behaviour accounts to guessing the key. Using this PRF, we can mount a distance-hijacking attack against a wide range of distance bounding protocols.

Description of Our Generic Attack We illustrate the attack on the DB3 protocol [15]. The DB3 (q=2) protocol works as follows. The verifier sends a nonce N_V, the prover replies with a nonce N_V. Both compute $a = f_x(N_P, N_V)$, where f_x is a PRF keyed with the shared key x. Then, in n timed rounds, the verifier sends a random bit c_i, expects a response $r_i = a_i \oplus c_i$. Finally, the prover sends $tag = f_x(M_P, N_V, c)$ (where c is the concatenation of all c_i values). The verifier accepts if the times, r_i and tag are correct.

We consider a generalised distance-fraud scenario, with an adversary $\mathcal{A}_{loc_2}$ in location loc_2, far from the location of the designated verifier $d\mathbb{V}$. Let P_x and P_y be WB provers, using respectively secret keys x and y. The attack goes as follows, and is illustrated on Figure 6:
1. $\mathcal{A}$ reads the secrets x, y from P_x, P_y; **2.** $\mathcal{A}$ uses $Move$ to move P_x far from $d\mathbb{V}$, and P_y close to $d\mathbb{V}$; **3.** $\mathcal{A}_{loc_2}$ sends $NP_x = x$ to $d\mathbb{V}$; **4.** $\mathcal{A}$ uses his $Launch$ oracle to make $\P_y$ start a session with $d\mathbb{V}$ by sending a message NP_y, which $\mathcal{A}$ redirects to $\mathcal{A}_{loc_2}$ with the $ChangeDestination$ oracle; **5.** $\mathcal{A}_{loc_2}$ receives a message NV_x from $d\mathbb{V}$, and sends a message $NV_y = y$ to P_y; **6.** P_y computes $a_y = PRF_y(NP_y, y) = R$, and $d\mathbb{V}$ computes $a_x = PRF_x(x, NV_x) = R$; **7.** $d\mathbb{V}$ sends n challenges, to which P_y replies using $a_y = a_x = R$; **8.** $\mathcal{A}_{loc_2}$ sends the final message tag, computed with the key x; **9.** $d\mathbb{V}$ accepts the auhentication of P_x, and $\mathcal{A}$ wins.

While we used the DB3 protocol to illustrate our attack, it applies to several protocols of the literature. For some protocols, NV is sent before NP, but a similar attack applies. Vulnerable protocols include: Kim and Avoine [32], Benfarah *et al.* [10] (both versions), TMA [46], Hancke and Kuhn [29], Munilla et Peinado [40], Avoine et Tchamkerten [5], Poulidor [45], NUS [28], Lee *et al.* [34], LPDB [39], EBT [25], Baghernejad *et al.* [26]. This list is not exhaustive.

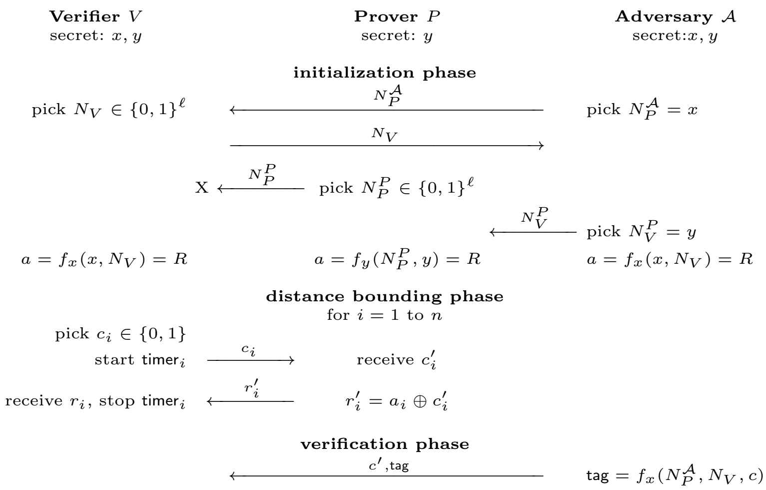

Fig. 6. Generalised distance-fraud on the DB3 (q=2) protocol [15]. The cross indicates a message blocked by $\mathcal{A}$.

7 Conclusions

We proposed a new application-oriented security model for distance-bounding. Using our adversary model, we exhibit flaws in 13 protocols of the literature, previously believed to be secure. One of them is the EMV protocol called PayPass with relay protection. We also propose a backwards-compatible version of PayPass, which we show fully secure in our strongest collusion model. This is the first practical DB protocol shown secure in a provable-security model. Moreover, our model completely eliminates terrorist-fraud, as irrelevant to any concrete implementation of a DB protocol. This underlines the need for fully secure protocols for real-life applications. We aimed to end the debate about how terrorist-fraud resistance should be formalised, and bring hope for a unified model in which real-life protocols can actually be proven secure. Our results pave the way for exciting research directions, such as the design of optimal application-oriented DB protocols, filling the long-lasting gap between academic protocols and practical applications/implementations. Finally, we built this model closes to the formalisms used in automatic tools to soon yield mechanised cryptographic proofs for DB. As future work, we aim to extend this with dishonest verifiers and with privacy/anonymity requirements, both of which are needed in some application-level DB.

References

1. A. Ahmadi and R. Safavi-Naini. Directional distance-bounding identification. In P. Mori, S. Furnell, and O. Camp, editors, *Information Systems Security and Privacy*, pages 197–221, Cham, 2018. Springer International Publishing.
2. G. Avoine, M. Bingol, I. Boureanu, S. Capkun, G. Hancke, S. Kardas, C. Kim, C. Lauradoux, B. Martin, J. Munilla, et al. Security of distance-bounding: A survey. *ACM Computing Surveys*, 2017.
3. G. Avoine, M. A. Bingol, S. Karda, C. Lauradoux, and B. Martin. A formal framework for analyzing RFID distance bounding protocols. In *Journal of Computer Security - Special Issue on RFID System Security, 2010*, 2010.
4. G. Avoine, X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J. Robert. A terrorist-fraud resistant and extractor-free anonymous distance-bounding protocol. In *Proc. of ASIA CCS '17*, pages 800–814. ACM, 2017.
5. G. Avoine and A. Tchamkerten. An efficient distance bounding RFID authentication protocol: Balancing false-acceptance rate and memory requirement. In *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*, pages 250–261, 2009.
6. D. Basin, S. Capkun, P. Schaller, and B. Schmidt. Lets get physical: Models and methods for real-world security protocols. In *International Conference on Theorem Proving in Higher Order Logics*, pages 1–22. Springer, 2009.
7. A. Bay, I. Boureanu, A. Mitrokotsa, I. Spulber, and S. Vaudenay. The Bussard-Bagga and Other Distance-Bounding Protocols under Attacks. In *Information Security and Cryptology - 8th International Conference, Inscrypt 2012*, Lecture Notes in Computer Science 7763, pages 371–391, Beijing, China, November 2012. Springer.
8. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, Berlin, Heidelberg, 2000. Springer.
9. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proc. of Advances in Cryptology – CRYPTO '93*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, 1994.
10. A. Benfarah, B. Miscopein, J. Gorce, C. Lauradoux, and B. Roux. Distance bounding protocols on TH-UWB radios. In *Proceedings of the Global Communications Conference, 2010. GLOBECOM 2010, 6-10 December 2010, Miami, Florida, USA*, pages 1–6, 2010.
11. I. Boureanu and A. Anda. Another look at relay and distance-based attacks in contactless payments. Cryptology ePrint Archive, Report 2018/402, 2018. `https://eprint.iacr.org/2018/402`.
12. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. On the pseudorandom function assumption in (Secure) distance-bounding protocols. In *Progress in Cryptology – LATINCRYPT 2012*, volume 7533 of *LNCS*, pages 100–120. Springer Verlag, 2012.
13. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Secure and lightweight distance-bounding. In *Proceedings of LightSec 2013*, volume 8162 of *LNCS*, pages 97–113. Springer-Verlag, 2013.
14. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical and Provably Secure Distance-Bounding. *Journal of Computer Security*, 23(2):229–257, 2015.
15. I. Boureanu and S. Vaudenay. Optimal proximity proofs. In *Proc. of Inscrypt*, pages 170–190. Springer, 2015.
16. S. Brands and D. Chaum. Distance-bounding protocols. In *Proc. of Advances in Cryptology – EUROCRYPT'93*, volume 765 of *LNCS*, pages 344–359. Springer-Verlag, 1993.
17. X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *WISEC 2016*, New York, NY, USA, 2016. ACM.
18. T. Chothia, J. de Ruiter, and B. Smyth. Modelling and analysis of a hierarchy of distance bounding attacks. In W. Enck and A. P. Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 1563–1580. USENIX Association, 2018.
19. 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. Springer, 2015.
20. C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *IEEE Symposium on Security and Privacy*, 2012.
21. C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *S&P 2012*, Washington, DC, USA, 2012. IEEE.

22. A. Debant, S. Delaune, and C. Wiedling. Proving physical proximity using symbolic models. Research report, Univ Rennes, CNRS, IRISA, France, Feb. 2018.
23. U. Dürholz, M. Fischlin, M. Kasper, and C. Onete. A formal approach to distance-bounding rfid protocols. In X. Lai, J. Zhou, and H. Li, editors, *Information Security*, volume 7001 of *Lecture Notes in Computer Science*, pages 47–62. Springer Berlin Heidelberg, 2011.
24. M. Emms, B. Arief, L. Freitas, J. Hannon, and A. van Moorsel. Harvesting high value foreign currency transactions from emv contactless credit cards without the pin. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 716–726, New York, NY, USA, 2014. ACM.
25. R. Entezari, H. Bahramgiri, and M. Tajamolian. A mafia and distance fraud high-resistance rfid distance bounding protocol. In *ISCISC*, pages 67–72, 2014.
26. M. S. Fatemeh Baghernejad, Nasour Bagheri. Security analysis of the distance bounding protocol proposed by jannati and falahati. *Electrical and Computer Engineering Innovations*, 2(2):85–92, 2014.
27. M. Fischlin and C. Onete. Terrorism in distance bounding: Modeling terrorist-fraud resistance. In M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *Applied Cryptography and Network Security*, pages 414–431, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
28. A. O. Gürel, A. Arslan, and M. Akgün. Non-uniform stepping approach to rfid distance bounding problem. In *Proceedings of the 5th International Workshop on Data Privacy Management, and 3rd International Conference on Autonomous Spontaneous Security*, DPM'10/SETOP'10, pages 64–78, Berlin, Heidelberg, 2011. Springer-Verlag.
29. G. P. Hancke and M. G. Kuhn. An RFID distance bounding protocol. In *Proceedings of SecureComm 2005*, pages 67–73. IEEE, 2005.
30. M. Igier and S. Vaudenay. Distance Bounding Based on PUF. In S. Foresti and G. Persiano, editors, *Cryptology and Network Security*, pages 701–710, Cham, 2016. Springer International Publishing.
31. H. Kilinç and S. Vaudenay. Formal analysis of distance bounding with secure hardware. In B. Preneel and F. Vercauteren, editors, *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, volume 10892 of *Lecture Notes in Computer Science*, pages 579–597. Springer, 2018.
32. C. H. Kim and G. Avoine. Rfid distance bounding protocol with mixed challenges to prevent relay attacks. In *Proceedings of the 8th International Conference on Cryptology and Network Security*, CANS '09, pages 119–133, Berlin, Heidelberg, 2009. Springer-Verlag.
33. S. Kleber, R. W. van der Heijden, H. Kopp, and F. Kargl. Terrorist fraud resistance of distance bounding protocols employing physical unclonable functions. In *2015 International Conference and Workshops on Networked Systems (NetSys)*, pages 1–8, March 2015.
34. S. Lee, J. S. Kim, S. J. Hong, and J. Kim. Distance bounding with delayed responses. *IEEE Communications Letters*, 16(9):1478–1481, 2012.
35. MasterCard. Contactless paypass reader specifications v3.1,. not publically available, 2017.
36. S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua. Distance-bounding protocols: Verification without time and location. In *S&P 2018*. Springer, 2018.
37. C. Meadows, R. Poovendran, D. Pavlovic, L. Chang, and P. Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. In R. Poovendran, S. Roy, and C. Wang, editors, *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, Boston, MA, 2007. Springer.
38. A. Menezes and N. P. Smart. Security of signature schemes in a multi-user setting. *Des. Codes Cryptography*, 33(3):261–274, 2004.
39. A. Mitrokotsa, C. Onete, and S. Vaudenay. Mafia fraud attack against the rč distance-bounding protocol. In *2012 IEEE International Conference on RFID-Technologies and Applications, RFID-TA 2012, Nice, France, November 5-7, 2012*, pages 74–79, 2012.
40. J. Munilla and A. Peinado. Distance bounding protocols for rfid enhanced by using void-challenges and analysis in noisy channels. *Wirel. Commun. Mob. Comput.*, 8(9):1227–1232, Nov. 2008.
41. C. Pöpper, N. O. Tippenhauer, B. Danev, and S. Capkun. Investigation of signal and message manipulations on the wireless channel. In *European Symposium on Research in Computer Security*, pages 40–59. Springer, 2011.
42. I. S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.
43. U. Ruhrmair and M. van Dijk. Pufs in security protocols: Attack models and security evaluations. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 286–300, Washington, DC, USA, 2013. IEEE Computer Society.

44. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. `http://eprint.iacr.org/2004/332`, 2004.
45. R. Trujillo-Rasua, B. Martin, and G. Avoine. The poulidor distance-bounding protocol. In *Proceedings of the 6th International Conference on Radio Frequency Identification: Security and Privacy Issues*, RFIDSec'10, pages 239–257, Berlin, Heidelberg, 2010. Springer-Verlag.
46. R. Trujillo-Rasua, B. Martin, and G. Avoine. Distance-bounding facing both mafia and distance frauds: Technical report. *CoRR*, abs/1405.5704, 2014.
47. J. van den Breekel, D. A. Ortiz-Yepes, E. Poll, and J. de Ruiter. Emv in a nutshell. Technical report, Technical Report, 2016.
48. J. van Leeuwen and J. Wiedermann. The turing machine paradigm in contemporary computing. In B. Engquist and W. Schmid, editors, *Mathematics Unlimited — 2001 and Beyond*, pages 1139–1155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
49. S. Vaudenay. On modeling terrorist frauds. In *International Conference on Provable Security*, pages 1–20. Springer, 2013.

A Related Work

Provable-Security Models for DB. The formalisation effort for distance bounding began with a first classification of attacks by Avoine et al. [3]. The authors distinguished black- and white-box provers, but did not provide a formal security model. Later, two main computational models were proposed: the DFKO model [23], and the BMV model [13]. They were further refined with different flavour of terrorist-fraud resistance: three different definitions for terrorist fraud in the DFKO model are given in [27], and even more in the successors of the BMV model. Terrorist-fraud resistance is generally achieved through artificial mechanisms (extractors, leakage schemes, backdoors ...), around which the definition is specifically designed. More recently, Kilinç and Vaudenay proposed a new provable-security model, which considers black-box provers [31]. In particular, they state that in a black-box context, terrorist-fraud resistance is the strongest and most generic property. Traditionally, these provable-security models have been used to study protocols from the academia, rather than practical ones. Our model is the first provable-security formalism to study practical, real-life protocols. Our model is most inspired by the initial framework of Avoine et al. [3], which it extends substantially.

Symbolic-Verification Models for DB. On the symbolic-verification side, where the cryptographic aspects are idealised, a first model was proposed in 2007 by Meadows et al. [37]. Later, Basin et al. proposed an Isabelle/HOL formalisation in [6]. This formalisation was extended by Cremers et al. [21] to include advanced message-manipulation, such as overshodowing, which permitted to discover distance -ijacking attacks. Last year, at Usenix, Chothia et al. [18] proposed a full DB formalisation, with a hierarchy of attacks, and used it to analyse real-world protocols: Mastercard's `PayPass` with relay protection, and NXP's relay-resistant protocol. This formalisation does however not consider dishonest provers. The same year, Mauw et al. [36] and Debant et al. [22] proposed other symbolic models for analysing distance-bounding protocols. In the last two lines, adversary model differs more widely from other works in formal DB: in particular, they allow for dishonest verifiers, and permit the adversary to block messages from afar. With this modified adversary model, they exhibit new attacks on recent protocols, such as [17,4]. In our new model, we also allow for such blocking of messages by the attacker, but we do not allow for dishonest verifiers. We add more finesse to the corruption of the provers and, as such, we exhibit new attacks, that no other model has found thus-far. Some of these are on practical protocols such as `PayPass`.

B Proofs for Section 3

Lemma 20. *Let Π be a specification of a DB protocol and Π^{real} be its realisation in an execution environment. If TF-provers are black-box, then Π^{real} is TF-resistant.*

Proof. Let P be a black-box TF-prover, far-away from a given verifier-party V. From Def 2, P being black-box means that any algorithm $\mathcal{B}$ that interacts with it follows the ITM specification in Π. So, P's help is the adversary $\mathcal{A}$ having $\mathcal{B}$, which equates to $\mathcal{A}$ interacting in the session *mid* with a far-away prover P. According to the communication model and adversarial model, V cannot accept P in *mid* (as P is far-away). So, Π is TF-resistant by trivial implications: i.e., " if P helps $\mathcal{A}$ make the verifier-party V output 1..." is always false. $\square$

Lemma 21. *Let Π be a specification of a DB protocol and Π^{real} be its realisation in an execution environment. If TF-provers are black-box, then a TF-attacker $(P, \mathcal{A}^1)$ succeeds against Π^{real} if and only if there is a successful MiM attacker $\mathcal{A}$ against Π^{real}.*

Proof. Assume a TF-attacker $(P, \mathcal{A}^1)$ where P is black-box. Let us consider an arbitrary universe of locations (loc_1, loc_2) in the execution environment such that, w.l.o.g.,that the verifier-party V in the TF attack is found in location loc_2 and P is therefore found in loc_1.

Let $\mathcal{A}$=$(\mathcal{A}_{loc_1}, \mathcal{A}_{loc_2})$ be an arbitrary adversary in our model. Let the adversarial party $\mathcal{A}_{loc_1}$ hold the black-box prover P (which is allowed in our model). And let $\mathcal{A}_{loc_1}$ communicate with $\mathcal{A}_{loc_2}$ found at the same location as V, as per our communication model. So, if TF-attacker $(P, \mathcal{A}^1)$ succeeds, since P is black-box, so does $(\mathcal{A}_{loc_1}, \mathcal{A}_{loc_2})$.

If $(\mathcal{A}_{loc_1}, \mathcal{A}_{loc_2})$ authenticates as P, then let $(\mathcal{A}_{loc_1}, adv_{loc_2})$ be the help that P gives $\mathcal{A}^1$. And so, $(P, \mathcal{A}^1)$ succeeds. $\square$