BoxDB: Realistic Adversary Model for Distance Bounding

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. Recently, the worldwide-used EMVCo standard for electronic payments included the "EMV RRP (Europay Mastercard Visa Relay-Resistant Protocol)" protocol. This uses distance bounding to counteract relay attacks in contactless payments. Last year, EMV RRP was widely analysed by symbolic verification methods, with several distance-bounding attacks and fixes proposed. Yet, one version of EMV RRP was found secure by all such formal analyses. Contrary to this, we exhibit an attack on this version of EMV RRP. Moreover, we exhibit similar vulnerabilities on another EMV RRP variant and on 13 distance-bounding protocols. We then propose a secure version of the EMV RRP protocol, called *PayPass+*, and prove its security, in a strong adversary model.
Our attacks stem from a new, fine-grained corruption model. We formalise it in a provable-security model, called *BoxDB*. In BoxDB, we express traditional as well as *new* DB security-properties. All our positive and negative security results are given in this formal model. Also, to fill a gap in computer-aided verification of security, BoxDB is designed specifically to lay the foundations for machine-checked cryptographic proofs for protocols based on distance bounding.
The threat model in BoxDB is modular and can be tailored to different applications. Importantly, the corruption model in BoxDB also leads us to show that the strongest threat against DB protocols, namely terrorist frauds, need not be considered in formal DB-security models.

1 Introduction

In relay attacks, an adversary forwards communications between two parties, without their knowledge, with the aim to an illicit gain, such as authenticating as one of the two. In 2015, the most widely used contactless electronic-payment protocol, EMV (Europay, Mastercard and Visa), was shown susceptible to relay attacks [19] leading to fraudulent payments. In 2016, Mastercard and then the EMV standard included relay-protection onto contactless EMV, yielding a variant of PayPass often referred to as the "contactless EMV with Relay Resistant Protocol (RRP)" [40] or *EMV-RRP* – for short; see Figure 1. The relay countermeasure in EMV-RRP imposes an upper bound on the round-trip times (RTTs) of message-exchanges. This widely studied mechanism is known as *proximity-checking*: a prover (e.g., contactless card) proves that it is within close distance to a given verifier (e.g., EMV reader) if the RTTs between the two are within the said upper bound. In EMV-RRP the prover also authenticates itself to the verifier. The primitive where *proximity-checking is composed with a unilateral authentication or identification mechanism* is known as *distance-bounding* (DB) [16]. The threat model for DB includes identify-frauds and impersonation attacks, but also vulnerabilities stemming from forged proximity proofs.

In the last year, EMV-RRP started being studied widely in DB-centred security models [22,41,18,21]. In particular, [41,11] showed that the EMV-RRP protocol has a flaw, whereby dishonest cards found far away from the reader can still authenticate. They proposed a fix to this attack, in a new version of the protocol[4] which we call *EMV-RRPv2*, described in Figure 1. EMV-RRPv2 has been analysed and proven secure [41,18,21].

[4] This is called sometimes PaySafe-v2 [18,21].

On a different line, very recently, [50] looks at alleviating other problems in EMV-RRP which stem from the reader being dishonest and not carrying out the proximity check. To this end, a new variant of EMV-RRP was proposed, called *PayBCR* [50], whereby the card-issuing bank receives and can verify a proof of the proximity-check executed in the protocol.

In this paper, we follow the lines of analysing variants of EMV-RRP in formal distance-bounding models. We propose a fine-grained security model for application-level distance-bounding (and EMV-RRP is an application-level DB protocol). Unlike in previous models, we distinguish different types of prover-corruptions, in a precise manner. This leads us to finding new attacks in EMV-RRPv2, PayBCR and other DB protocols.

More concretely, our contributions are as follows.

Contributions.

1. New Attacks on Contactless EMV. We show that the proven-secure protocol EMV-RRPv2 is in fact insecure: a far-away card-holder can fool the reader into believing that he is near-by. The strategy behind our attack is new. The flaw we exhibit is all the more worrying as there are new proposals [50] where the RTT-checks done by the reader are sent to the card-issuing bank as proximity-proofs. That is, our attack can lead to a bank accepting a forged proximity proof, which in turn can then be used for fraudulent claims of reimbursements. In other words, our attack also applies to PayBCR, i.e., the newly-enhanced EMV-RRP designs in [50]. We show that, in fact, even a weaker version of the attack on EMV-RRPv2 affects PayBCR.

2. New Attacks on Existing DB Protocols. We show that the EMV-RRPv2 vulnerability also affects 13 existing DB protocols.

3. New Formal Model & Security Definitions. We propose a new security model for distance bounding called *BoxDB*. Inspired by real-life implementation, this model explicitly distinguishes provers being corrupted in a black-box or white-box manner. First, in BoxDB, we formalise a general vulnerability that captures traditional DB frauds, our aforementioned attacks and more: *"generalised distance fraud" (GDF)*. Second, we formalise other DB-specific threats via a generalised notion akin to man-in-the-middle attacks, called *"generalised mafia fraud" (GMF)*. Our security definitions are **not** synonymous to previous formalisations of advanced DB frauds (e.g., in [14,20], etc.). They are new, stronger security definitions, cast in a strong and fine-grained adversarial model. I.e., a protocol can be secure w.r.t. the generalised distance-fraud in [14] but insecure w.r.t. our GDF property[5]. Third, we introduce a new security definition of resistance to *secret extraction*. This notion is not explicitly present in previous DB models, yet we argue that it is crucial that it becomes a standalone requirement in DB – at least, when considering a fine-grained threat model like ours. Forth, BoxDB is designed to lay the foundations for machine-checked proofs for application-level DB: game-based security definitions accompanied by an oracle-based adversary model, a simple mechanism based on "locations" to handle distance, a reduction in the number of properties to check, and more.

4. New Terrorist-Fraud Treatment. Leveraging the fine-grained corruptions in BoxDB, we show that the hard-to-achieve resistance against a DB-specific identity fraud, called *terrorist fraud* [23], is in fact irrelevant. This settles decade-old controversies around attaining security against this threat. Concretely, we exhibit a generic terrorist fraud in the case where we operate with provers corrupted in a white-box manner, and show that resistance to terrorist fraud is meaningless if the provers are corrupted in a black-box manner.

[5] In Section 3, when we present the attack on EMV-RRPv2, as well as in Section 8, we argue that our strong threat model is the sensible one to use for DB and, in general, for wireless communications.

5. Provable-Security of an Enhanced `EMV-RRPv2`. We propose a minor modification to `EMV-RRPv2`, and prove its full security in BoxDB.

Outline. We start by recalling some preliminaries in the next section. In Section 3, we describe our attacks. In Section 4, we introduce our model *BoxDB*. In Section 5, we define the security properties in our model, and –importantly– show how they can be cast, in a simple manner, to stronger or weaker versions, depending on the level of corruption (in a DB system). We also discuss which properties would be needed in different modular, adversarial settings (which may fit respectively different applications). In Section 6, we prove the security of an improved version of `EMV-RRPv2` in our model. In Section 7, we show that terrorist-fraud resistance is a dispensable property in provable-security DB models. In Section 8, we compare our results with related work and conclude.

2 Preliminaries

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 [3], three "classical" types of attacks are considered. 1) In a *distance-fraud* (DF), a dishonest prover P^* tries to prove that he is within the distance bound when he is not. 2) 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 to V as P. 3) In a *terrorist-Fraud* (TF) [23], a dishonest and remote prover P^* colludes with an accomplice $\mathcal{A}$ so that $\mathcal{A}$ authenticates as the out-of-bound P^*. The fraud is considered successful only if $\mathcal{A}$ cannot impersonate P^* in a latter session. The TF adversary-model excludes the trivial attack in which P^* gives his credentials to $\mathcal{A}$. In recent years, the DB threat-model has widened: [25] coined *impersonation attacks*; [20] presented *distance hijacking (DH)* — which extends distance fraud by allowing the dishonest P^* to exploit honest provers located near V; [14] gives further generalisations of these, e.g., mafia-fraud captured via more powerful *man-in-the-middle (MiM)* and terrorist-fraud up-cast to *collusion-fraud*, which considers repeated collusions between P^* and $\mathcal{A}$.

In the rest of the paper, we use interchangeably the words: *card* and *prover*; *reader* and *verifier*.

Contactless EMV with Relay Protection. The `EMV-RRP` protocol with relay protection is an industry standard for relay-attack protection in contactless payments. It is depicted in Figure 1. It was adopted first by Mastercard, and it has been present in the EMV specification from version 2.5 onwards [26].

The card includes: a private key $Priv_C$; a symmetric key K_M that it shares with the bank; a certificate chain for its public key Pub_C. The reader generates a 32-bit nonce UN. The card generates a 32-bit nonce $Nonce$. The reader initiates the communication, and, in the 2nd message, the card sends some payment-application data. The reader then sends the "relay-resistance" command, which equates to "here is my nonce UN". Then, the card replies with $Nonce$ and some timing info (i.e., how long the card takes for certain computations). This exchange is timed by the reader, to potentially detect if the messages are relayed. The reader then retrieves Pub_C from the relevant certificates. Finally, the reader asks the card to generate a "cryptogram" AC to make the payment. To this end, the card generates a session key K_S as the encryption of its application transaction counter (ATC) under K_M. The cryptogram AC is a MAC keyed under K_S of a series of data including the ATC, the nonce UN. The card also produces the "Signed Dynamic Application Data"(SDAD): it signs a message including UN, amount, currency, ATC, $Nonce$, etc. In the last message in Figure 1, the card sends

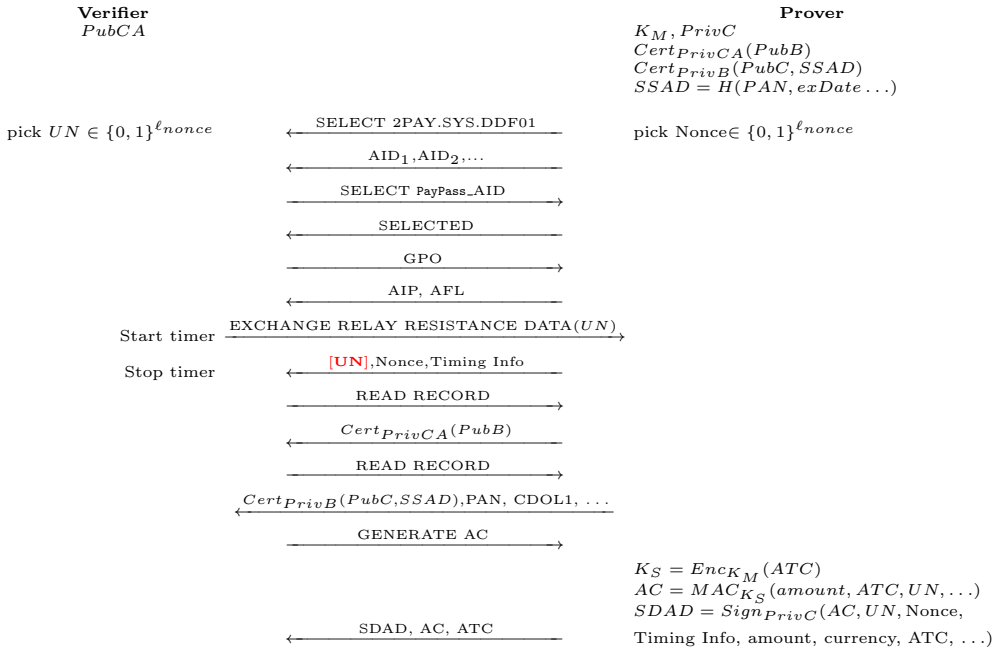

Fig. 1. `EMV-RRP` & `EMV-RRPv2`. The nonce UN, in red and in brackets, in the 8th message, is only sent in `EMV-RRPv2`

the cryptogram AC, the SDAD, as well as the ATC. The reader uses Pub_C to the check the signature SDAD and the consistency of the sub-messages inside (e.g., UN, amount, etc.); it forwards AC to the card-issuing bank. The reader accepts if the measured time is within a set bound and the $SDAD$ is correct. Note that the correctness of the MAC is irrelevant to the reader-side of the protocol, as it can only be verified by the bank.

The version `EMV-RRPv2` of `EMV-RRP` is also shown in Fig. 1. It differs from `EMV-RRP` only in that in the timed phase, the card adds UN to its response. This protects against certain distance frauds in `EMV-RRP`, as [41,11] showed.

3 Attacks on Contactless Payments and Authentication

3.1 Insecurities in Contactless Payments

A.1 Attacks on `EMV-RRPv2`. `EMV-RRPv2` was symbolically verified in [41,18,22,21] and found secure. We now show that `EMV-RRPv2` is in fact vulnerable to a form of advanced distance hijacking. In Section 5, in our model BoxDB, we formalise this *new type of advanced fraud*, which we call *generalised distance fraud*.

Our Attack on `EMV-RRPv2`. Consider a malicious prover $\mathcal{A}$ found far-away from the verifier V, and a honest prover P within the distance bound of V. The attack begins when P starts a session with V, and runs is as follows:

1. during the non-timed parts, $\mathcal{A}$ overwrites the messages sent by P with his own;
2. in the timed phase, $\mathcal{A}$ overwrites the nonce from P with his own, without affecting the rest of the message (UN);
3. $\mathcal{A}$ sends the correct MAC/signature, containing $\mathcal{A}$'s injected nonce and SSAD.

In this way, $\mathcal{A}$ is accepted by V, even though he is far away. This attack is shown in Figure 2, where **X** denotes the fact that a message is (sometimes partly) overwritten by $\mathcal{A}$.

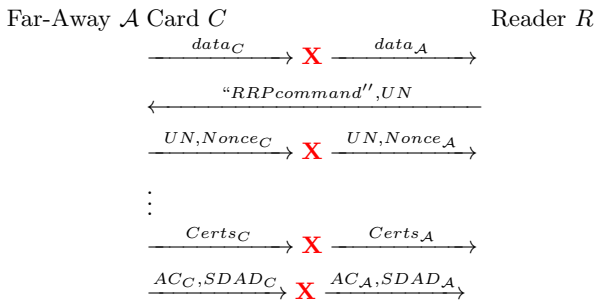

Fig. 2. Generalised Distance Fraud Attack on `EMV-RRPv2`.

In Section 6, we prove that a minor modification to `EMV-RRPv2` provably protects it against this attack, in BoxDB.

Threat Model for Our Attack on `EMV-RRPv2`. The adversary in our attack is strong. In particular, it can block or overwrite selected parts of a message from afar. Yet, these assumptions are in line with the real-life threats in wireless communications. For instance, jamming attacks include signal annihilation, modification (bit-flipping, overshadowing) as well as the insertion of forged or replayed signals [1,45]. Moreover, other formalisms for distance bounding (used in analysing `EMV-RRP` and `EMV-RRPv2`) also consider such strong adversaries who can, e.g., block messages from afar [22,21,41].

There is one difference between our adversary and that in [22,21,41]. In the latter, the attacker is a proximity-driven, Dolev-Yao attacker [24]. He can block $(UN, Nonce_C)$, but –as he is far-away– he cannot decompose the term $(UN, Nonce_C)$, hence he cannot learn UN, and cannot compose $(UN, Nonce_{\mathcal{A}})$ in time to inject it. Overshadowing attacks, and ours is one of these, are possible in practice[6], and are in fact used in symbolic verification of DB [22,21,41]. But, the symbolic-verification DB-attacker theory in [22,21,41] cannot fully capture this particular overshadowing attack (i.e., on a pair of two messages) and therefore fails to find it.

A.2 Attacks on PayBCR. In [50], a new version of `EMV-RRP`, called PayBCR is proposed. An attestation of the proximity-checking performed by the reader is sent to the card-issuing bank, who can further check it. In this case, the transaction constitutes a strong proof that the card was within the range of the verifier when the purchase was made.

Yet, PayBCR is based directly on `EMV-RRP`. So, PayBCR is already vulnerable[7] to the standard distance fraud show in [41,11] on `EMV-RRP`, whereby the card sends $Nonce_C$ before receiving UN. The fix [41,11] that transformed `EMV-RRP` in `EMV-RRPv2` (i.e., adding UN to the card's timed response), would also sanitise PayBCR of this DF attack.

[6] As [45] explains, this is certainly possible even if C and R are close together and $\mathcal{A}$ is far from both, as long as there are no major physical obstacles (e.g, large buildings) between $\mathcal{A}$ and the two honest parties C and R.

[7] The threat model in [50] does not include distance fraud, yet if `EMV-RRP` is analysed against DF and then so should PayBCR. This is even more so since the card-issuing bank receives and verifies an attested "copy" of the proximity proof.

However, our "strong DF" attack above, onto EMV-RRPv2, also applies to PayBCR and its potential fixes a la EMV-RRPv2. Fortunately, the remedy we propose in Section 6 to provably repair EMV-RRPv2 also applies to PayBCR.

A.3 Are these attacks important? Firstly, distance frauds would also translate into financial loss for the banks. Assume a malicious card paying legitimately in store A. If this card can mount a distance fraud to pay in a far-away store B at the same time, then the card owner can then claim that their card was hacked/cloned, as it appears to be paying in two locations at the same time. This would most likely entail the bank having to reimburse both purchases.

Secondly, the above is even more problematic in the case of the DF we showed on top of PayBCR. Therein, any forgery of proximity-proof by a dishonest card is a forgery of a (hardware-attested) proof accepted by the bank. So, this can not only lead to reimbursement of fraudulent payments, but it can be used as a strong alibi by the card owner to show that he/she was by the payment terminal when they were not.

3.2 Insecurities in Contactless Authentication

The attack we showed above on EMV-RRPv2 exploits an honest prover P found within the distance-bound from the verifier: the adversary $\mathcal{A}$ overwrites some of the honest prover's messages with his own. In more advanced attacks, $\mathcal{A}$ could combine such message-replacing with the exploitation of cryptographic primitives in the protocol. We show how such techniques lead to new attacks on the proven-secure DB3 distance-bounding protocol [15], as well as at least 13 other distance-bounding protocols.

B.1 Attacks on DB3. The DB3 protocol (with its main parameter q equal to 2) is a proven-secure DB protocol. It is depicted in Figure 3. A complete description can be found in [15]. In DB3, as in many distance bounding protocols, the response to the challenge c_i is computed as a function of a and c_i, where a is the output of a PRF f for some initially-exchanged nonces. The key idea of our attack is to force a honest prover located near the verifier to use the same a as an adversary located far from the verifier.

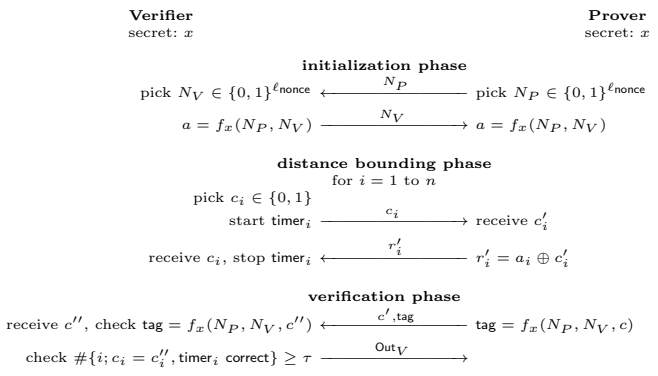

Fig. 3. The DB3 Distance-Bounding Protocol [15] (parameter $q = 2$)

Consider an adversary $\mathcal{A}_{far}$ located out of the distance bound of the verifier V. Consider that $\mathcal{A}_{far}$ knows the respective secret keys x and y of two provers: P_x and P_y. P_y is within

the distance bound of a verifier V, whereas P_x is far away. Finally, assume that the PRF used in the protocol is programmed, as defined in [12]. Specifically, it returns a constant value R when one of its inputs is its secret key. It remains a secure PRF, as another adversary would have to guess the secret key to see the non-random behaviour, but this PRF can be exploited by our (key-knowing) $\mathcal{A}$ to mount his attack. Let f be the PRF specified in DB3, f_z denote an instance of f keyed with a key z, and R be a constant. Let pf be the programmed version of f, used in the actual realisation of DB3. Namely, pf is programmed as follows:

$$\mathsf{pf}_z(N_P, N_V) = \begin{cases} R \text{ if } N_P = g(z) \\ R \text{ if } N_V = h(z) \\ f_z(N_P, N_V) \text{ otherwise,} \end{cases}$$

where g and h are functions from $\{0,1\}^{|z|}$ to $\{0,1\}^{|nonce|}$. For clarity, we use $g(z) = h(z) = z$. So, due to this programmed PRF, a timed phase run by the close-by P_y will appear the same as if this was run by the distant prover P_x. In the verification phase after the timed phase, the remote attacker can send the P_x-produced message tag from afar.

The full attack, illustrated on Fig. 4, is summarised step-by-step thereafter.

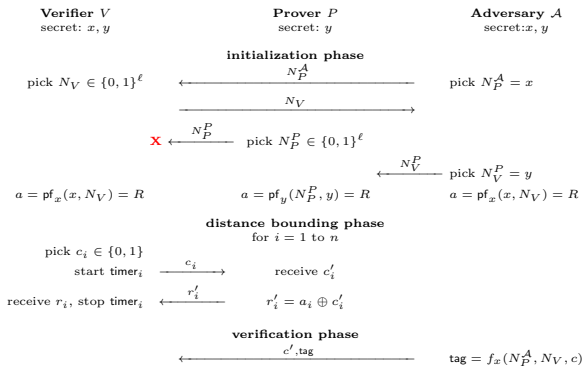

Fig. 4. Generalised Distance-fraud on the DB3 (q=2) Protocol [15]. The cross **X** indicates a message blocked by $\mathcal{A}$.

1. $\mathcal{A}_{far}$ requires V to start a session for P_x;
2. $\mathcal{A}_{far}$ sends $N_P^{P_x} = x$ to V;
3. $\mathcal{A}$ makes P_y start a session with V, and blocks P_y's nonce $N_P^{P_y}$;
4. $\mathcal{A}_{far}$ receives the verifier's nonce $N_V^{P_x}$, and sends $N_V^{P_y} = y$ to P_y (while preventing P_y to receive $N_V^{P_x}$);
5. P_y computes $a_y = \mathsf{pf}_y(N_P^{P_y}, y) = R$, and V computes $a_x = \mathsf{pf}_x(x, N_V^{P_x}) = R$;
6. V sends n challenges, to which P_y replies using R;
7. $\mathcal{A}_{\mathsf{far}}$ blocks all messages from P_y after the end of the timed phase, and sends his own messages instead, computed with $x, N_V^{P_x}, N_P^{P_x}$;
8. V wrongly accepts the authentication on x of P_x who is afar as if P_x was close by.

Discussion of the Attack on DB3. Our attack on DB3 uses an even stronger adversary model than our attack against `EMV-RRPv2`. Not only can the adversary perform advanced

message-manipulation, such as preventing parties from receiving messages from afar, but he also knows the secret key of more than one prover. In contexts where secret keys can be known, PRF programming attacks [12] are generally accepted: rogue manufacturers (who do know the keys) can easily use such PRFs inside devices they produce.

B.2 Furthering the Attack on DB3. The same attack applies to other distance bounding protocols. A non-exhaustive list of which is given in Table 1. While the details of the response function may vary with each protocols in Table 1, this function always uses a bitstring a as the output a PRF on two nonces used in the initialisation phase. However, compared with DB3, some other details may differ. Columns 2 and 3 of Table 1 capture such differences: column 2 indicates whether N_P is sent before N_V; column 3 indicates whether messages are sent after the timed phase.

For the protocols where V sends his nonce before the prover's, step 4 in our DB3 attack would be moved between steps 1 and 2.

For the protocols where no messages are sent during after the end of the timed phase; in this case, step 7 becomes void.

Finally, in the protocol in [6], an additional value v_0, derived from a, is sent by the prover before the timed phase: $\mathcal{A}$ can either send it, or let the close-by prover send it.

Protocol	N_P first	Final message
Kim and Avoine [37]	✗	✗
Benfarah *et al.* [10] (both versions)	✗	✗
TMA [52]	✗	✗
Hancke and Kuhn [34]	✓	✗
Munilla et Peinado [44]	✓	✓
Avoine et Tchamkerten [6]	✓	✗
Poulidor [51]	✓	✗
NUS [32]	✓	✓
Lee *et al.* [39]	✓	✗
LPDB [43]	✓	✗
EBT [27]	✓	✗
Baghernejad *et al.* [28]	✓	✗

Table 1. Certain DB Protocols Vulnerable to Generalised Distance Fraud via Programmable PRF, in *BoxDB*.

The attacks in this section are not captured[8] by previous formal models for DB. So, next, we introduce a new model BoxDB. Our BoxDB formalism does not only include a security property that captures (security against) these attacks on DB3 and the protocols in 1, but it also takes into account fine-grained corruptions and a strong adversary model (in line with these attacks).

4 BoxDB: A Fine-grained Provable-Security Model for Distance Bounding

DB protocols are unilateral authentication/identification schemes, where (a) the verifier authenticates/identifies the prover; (b) the verifier measures the RTT of exchange(s) with the prover to be convinced that he is close to the prover. We consider that a ***DB protocol-specification***

[8] Boureanu et al. in [12] showed different DFs via PRFs on the 4th and the 6th protocols in the Table.

contains a tuple $(b, s, \mathbb{P}, \mathbb{V}, auths)$: a security parameter s[9]; b denotes a upper-bound for certain round-trip times (RTTs) in the protocol exchanges; a *system of unique long-term material* (e.g., keys), denoted *auths*, on which the authentication can take place; the *publicly-known algorithm of the prover* $\mathbb{P}$; *publicly-known algorithm of the verifier* $\mathbb{V}$, which outputs a boolean $out_{\mathbb{V}}$ denoting whether the authentication and RTT-checks were both successful or not.

Parties or Devices. We use the notion of a *"party"* or *"device"* to denote the hardware/software implementation of an algorithm. For now, we distinguish prover vs. verifier devices.

Definition 1 (Prover party). *Let Π be a protocol specification. A* prover-party/prover-device *P is an implementation of the algorithm $\mathbb{P}$ together with instantiated authentication information – auths*[10] *in Π. We consider two classes of implementation:* white-box (WB)*, denoted P_{WB}, and* black-box (BB)*, denoted P_{BB}.*

• P_{WB}: there exists a ppt. algorithm that can retrieve the long-term authentication information from the device.

• P_{BB}: any ppt. algorithm interacts with P_{BB} only in the same way any ppt. algorithm interacts with the algorithm $\mathbb{P}$.

So, by Def. 1, (secret) data found inside a black-box prover implementation P_{BB} cannot be read. Meanwhile, a white-box implementation prover P_{WB} can be fully read by a third party. Whilst a WB implementation can be read, memories or algorithms found inside WB or BB parties can not be altered. This means that prover-parties (as per Def. 1) behave honestly. Dishonest provers will be modelled as adversaries interacting with provers, which will be defined in Def. 6.

Definition 2 (Verifier party). *Let Π be a protocol specification. A* verifier-party/verifier-device *V is an implementation identical to the algorithm $\mathbb{V}$ together with instantiated authentication information – part of auths in Π.*

Any ppt. algorithm interacts with V only in the same way any ppt. algorithm interacts with the algorithm $\mathbb{V}$.

So, by Def. 2, we consider only one class of implementations for the verifier-algorithm $\mathbb{V}$. And, each implementation in this class is correct, as per the specification and cannot be read or altered by a third party.

Registering Parties. Parties are used in the actual running of DB protocols. For parties to be functional within a protocol execution, we consider that a *Setup procedure* exists. Via this *Setup* procedure, verifiers, white-box and black-box provers can be added to a distance-bounding backend system, and their instantiated *auths* linked up as necessary, e.g., concrete keys be shared by prover-parties and verifier-parties, etc.

Public Identifiers. According to DB literature [3], *auths* inside a protocol specification is formed of long-term shared keys, pairs of secret/public keys, physically unclonable functions (PUF) [31]. We consider that each given party is uniquely identified by part of such *auths*, once instantiated inside implementations as per Definitions 1 and 2. Part of this uniquely identifiable, long-term information carried by a party is secret.

[9] Computational measures such as polynomial probabilistic time (ppt), negligible, etc., vary with the security parameter. We consider these and associated notions, e.g., Interactive Turing Machines (ITMs) [53], commonplace.

[10] The bitlength/security of this authentication information is a function of s.

We assume that each such information is mapped by a non-invertible function to a *publicly disclosable, unique identifier* or simply *identifier*. We 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.

Definition 3 (Realised Distance-Bounding Protocols). *A* realisation $\Pi^{\texttt{real}}$ of a DB protocol Π *is a tuple* $\Pi^{\texttt{real}}=(P_1,\ldots,P_r,V_1,\ldots,V_m,\mathbb{B})$*, where:*
– *r is at most a polynomial in s and $P_1,P_2,\ldots,P_r$ are WB and BB prover-devices implementing $\mathbb{P}$ in Π and registered via Setup;*
– *m is at most a polynomial in s and $V_1,V_2,\ldots,V_m$ are verifier-devices implementing $\mathbb{V}$ in Π and registered via Setup;*
– *$\mathbb{B}$ is a value of the* distance bound *(which may be fine-tuned compared to the value b given in the specification Π).*

On White-box vs. Black-box Provers. In a realised DB protocol $\Pi^{\texttt{real}}$, we consider that we may have WB or BB implementations of the algorithm specification $\mathbb{P}$ in Π. As such, Def. 3 goes into actual realisation of algorithms, which has not been explicitly done before. Yet, in classical provable-security DB models [14,29], provers are considered white-box by default. Whilst Avoine et al [4] do consider the WB-BB distinction, this is not formalised in the way or at the level (of adversarial, fine-tuned manipulation) pursued further herein.

In our case, white-box provers (the memory of which can be read by third parties) can be explicitly registered into a "DB system". This denotes a possible corruption or vulnerability of those provers that pre-dates the attacks we are concerned with in the "DB system". As such, registered WB provers can denote: an implementation onto a mobile phone (even in the case of EMV) which can be read by a mobile-app developer, or a prover that has been already attacked (e.g., via a side-channel attack) and as such his memory was already read by the adversary, etc In other words, we can assume that white-box provers registered onto the system are provers who's secrets already leaked to the adversary; clearly, we will not be interested in any security properties concerning directly these provers but we do target analysing if their presence can affect the security of the BB provers in the "DB system". But, we will also look at the case where there are no such WB provers registered in $\Pi^{\texttt{real}}$ and the system runs only with BB provers.

Lastly, note that the adversary controlling a series of secret/private keys (i.e., which is equivalent to registering certain WB provers onto the system) is a setting used in provable-security DB models [14,29] and in symbolic verification of DB [41,22,18].

4.1 Physical & Communication Model

Like other DB models where distances are explicitly treated, e.g., [14], we consider that parties are positioned in an Euclidean space. But unlike other provable-security models, we also operate over *locations* (in the Euclidean space), which are formalised below.

Definition 4. ***Locations & Distances between Locations.*** *Let $\mathcal{P}$ be a set of Euclidean points. A* $\mathcal{P}$-location *is a $\mathbb{B}$-neighbourhood of $\mathcal{P}$*[11]*.*

The distance between a $\mathcal{P}_1$-location loc_1 and a $\mathcal{P}_2$-location loc_2 *is the shortest distance between a point in $\mathcal{P}_1$ and a point in $\mathcal{P}_2$.*

[11] This is the set of all Euclidean points 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 $\mathcal{P}$).

To formalise DB communication and security, we operate only with a convenient set of locations.

Definition 5. ***Universe of locations.*** *A* 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 points in each locations are within the distance $\mathbb{B}$ *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. 5 is a simple illustration of Def. 5: (a) points A and B are in one location and within distance $\mathbb{B}$ from one another; (b) points C and D are within distance $\mathbb{B}$ from one another, in another location. Point A is further than $\mathbb{B}$ from point C and from point D, and point B is further than $\mathbb{B}$ from point C and point D. However, Fig. 5 is simply an example: e.g., in a (universe of) locations, the set $\mathcal{P}_1$ could contain only A, or only B, or other points.

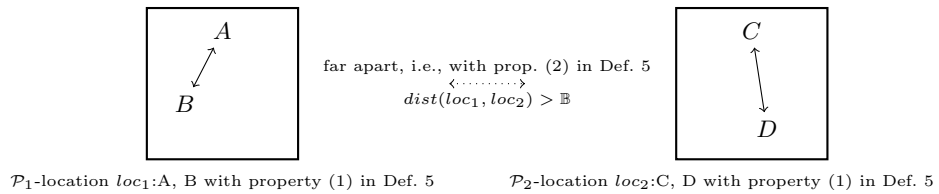

Fig. 5. An illustrative universe of locations.

Physical Positioning. We consider that parties associated with a realised DB protocol Π^{real} are always positioned within a universe of location $\mathcal{U} = (loc_1, loc_2)$, i.e., either some parties will be loc_1 and others in loc_2, or all with be in one of the two locations loc_1 or loc_2. We say that two parties are *close* if their are in the same location, and *far* otherwise. If a party E is found in a location loc, we write E_{loc}.

Communication. Parties exchange messages, since they are implementations of the algorithms $\mathbb{P}$ and $\mathbb{V}$ in a DB specification Π. 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 reaches a party D within the time $t^{\mathbb{B}}$ if and only if party C is *close*, i.e., within bound $\mathbb{B}$, to party D.

All messages are by default broadcast. Also, like all models for DB where there are distant and MiM attackers, we rightly abstract away from restrictions on the maximum communication distance between two parties[12]. We assume that response-computation is instantaneous during the timed phases of the protocol.

4.2 Threat Model

We now introduce a new type of party: the adversary.

Definition 6. ***Adversary.*** *Let* $\mathcal{U} = (loc_1, loc_2)$ *be a universe of locations in which parties of the realisation* Π^{real} *of a DB protocol are present. 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* $\mathcal{A}_{loc_2}$*, respectively located in* loc_1 *and* loc_2*, such that:*
(1). *Both* $\mathcal{A}_{loc_1}$ *and* $\mathcal{A}_{loc_2}$ *implement an arbitrary ppt. algorithm.*

[12] In attacks in DB or contactless communications, signals are adversarially amplified.

(2). *Devices $\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$ may be absent from their respective location*[13]*. If one such adversarial device is not present at its associated location loc, we consider the algorithm $\mathcal{A}_{loc}$ to be void.*
(3). *$\mathcal{A}$ cannot change the corruption level of a party, i.e., he cannot change a P_{BB} into P_{WB} or vice versa.*
(4). *$\mathcal{A}$ interacts with non-adversary parties by sending them messages. These parties reply honestly as per Π.*
(5). *When a prover party is white-box, $\mathcal{A}$ can read its memory, but not modify it.*
(6). *$\mathcal{A}$ has physical access to prover-devices: he can, for instance, read any identifier printed on it.*
(7). *When interacting with a white-box prover-device, $\mathcal{A}$ has access to all relevant authentication accessible to the legitimate owner of this prover-devices, including, for instance, passwords or second factor authentication.*
(8). *$\mathcal{A}$ can* move *prover-parties from one location to another.*
(9). *$\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$ operate as ITMs: they collaborate and communicate.*
(10). *The messages between $\mathcal{A}_{loc_1}$ and $\mathcal{A}_{loc_2}$ are subject to a time-of-flight lower bounded by $t^{\mathbb{B}}$.*
(11). *$\mathcal{A}$ can* send unicast messages*, which can only be read by their intended target, e.g., using directional antennas [2].*
(12). *$\mathcal{A}$ can* block *any message from being received by a party of his choice, irrespectively of their position.*[14]
(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* [15]*.*

4.3 Concurrent-Execution Model

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

Definition 7. ***Execution Environments.*** *Let $\Pi^{\mathtt{real}}$ be a realisation of a DB protocol and let $\mathcal{U} = (loc_1, loc_2)$ be an arbitrarily fixed universe of locations. An* execution environment for $\Pi^{\mathtt{real}}_{\mathcal{U}}$ at the universe of locations $\mathcal{U}$ *denotes a polynomial number of 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*.*

Different extended environments are used for respectively different security properties as defined in Sec. 5. For instance, for our secret-extraction property, one of the two locations is empty. Or, in the attack phase of our generalised distance-fraud, $\mathcal{A}_{loc_1}$ is absent.

Sessions. Def. 7 allows for several executions from multiple prover-devices and several verifier executions from multiple verifier-devices interleaved concurrently in any possible way. As usual, we call each such execution a *session*. If one execution is run on a prover-device or

[13] The presence or absence of $\mathcal{A}_{loc}$ in loc, with $loc \in \{loc_1, loc_2\}$, is determined by the security property considered. See Section 5.
[14] In line with the threats defined in [46] and the adversary model of [22].
[15] This is known as overshadowing, see [46] for more details

verifier-device, then it is a *prover session* or a *verifier session*, respectively. We write X^i for the *i-th session of a party* X.

Each party involved in each execution environment $\Pi_{\mathcal{U}}^{\texttt{real}}$ has a *status*: active or inactive. When a prover or verifier is inactive, it ignores all incoming messages. Initially, all are inactive. A party is only active when it is involved in one or more session(s), and becomes inactive again when this/these finish.

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

4.4 BoxDB: Game-based Model for DB

We now provide our security model *BoxDB*. To formalise the interactions of the adversary with an execution environment, we introduce a *challenger*.

The transcripts of a prover-session and a corresponding verifier-session may differ, due to adversarial manipulation of messages. In the analysis of distance-bounding protocols, the transcript as seen by the verifier is vital (unlike the prover's), as verifier-transcripts that determine whether the authentication is accepted or not. Moreover, we consider that from a successful, full verifier-transcript one can extract the public identifier of the prover-party that was authenticated[16].

The Challenger (*Ch*). It generates executions environments. For a given security property, *Ch* creates a specific extended environment[17] (see Sec. 5). The adversary cannot directly interact with the environment: it can only use oracles provided by the challenger, which simulate the corresponding interactions. The challenger keeps track of all oracle calls and sessions.

Definition 8. ***The Challenger.*** *Let Π be a DB protocol. A challenger* Ch *is a ppt. algorithm who does the following.*

The challenger Ch picks an arbitrarily fixed universes of locations $\mathcal{U}$=(loc_1, loc_2), and runs the Setup algorithm multiple times to realise Π^{real}. Then, for the universe $\mathcal{U}$, the challenger Ch creates an extended execution environment for $\Pi_{\mathcal{U}}^{real}$. For each $\Pi_{\mathcal{U}}^{real}$, the challenger Ch maintains the following lists:

A. a* session-list** *Sessions* ***of (the polynomial number of) sessions present in the $\Pi_{\mathcal{U}}^{real}$ environment*, with Sessions containing tuples* $(\texttt{sid}, \texttt{E}, \texttt{t})$*, where: E is the identifier of a honest party (i.e., a prover or a verifier id), sid is a unique identifier of a session of E type (i.e., prover session or verifier session), and t is the corresponding transcript as seen by party E. The list is uniquely indexed by sid.*

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

The prover-list PL contains tuples of the form $(\texttt{P}_\texttt{k}, \texttt{xb}, \texttt{status}, \{\texttt{P}_\texttt{k}^\texttt{j}\}_{\texttt{j}\in\texttt{Sessions.index}}, \texttt{secret})$ *where: P_k is a prover-party identifier, $xb \in \{WB, BB\}$ denotes whether the prover-party is white-box or black-box; status is either inactive or active; $\{P_k^j\}_{j\in Sessions.index}$ is the set of*

[16] This is realistic (as such public identifiers are often sent in clear) and poses no problem herein, as we do not treat provers' anonymity or privacy.

[17] E.g., in a weak generalised mafia-fraud game, the challenger does not include white-box provers to the environment.

session identifiers in which P_k is involved at a given time[18] *and a corresponding entry (P_k^j, P_k, t) is found in Sessions; if xb=WB, then secret is formed of all the identifiable, inner material of the prover; if xb=BB, then secret is null. The prover-list PL is indexed over prover ids.*

C. a* verifier-list VL *of (the polynomial number of) verifiers present in $\Pi_{\mathcal{U}}^{\mathtt{real}}$.
The verifier-list VL contains tuples of the form $(\mathtt{V_m}, \mathtt{status}, \{\mathtt{V_m^i}\}_{\mathtt{i \in Sessions.index}})$, where: V_m is a verifier-party identifier, status is either inactive or active, $\{V_m^i\}_{i \in Sessions.index}$ is the set of session identifiers in which V_m is involved in at a given time[19] *and a corresponding entry (V_m^j, V_m, t) is found in Sessions. The verifier-list VL is indexed over verifier ids.*

D.* Locations' list *$LOCL$, which has the position of each prover, verifier and adversary entity.
It is $(\{\mathtt{P_{loc_1}}\}, \mathcal{A}_{\mathtt{loc_1}}, \{\mathtt{V_{loc_1}}\}, \{\mathtt{P_{loc_2}}\}, \mathcal{A}_{\mathtt{loc_2}}, \{\mathtt{V_{loc_2}}\})$, where $\{P_{loc_1}\}, \mathcal{A}_{loc_1}, \{V_{loc_1}\}$ respectively denote the set of provers at location loc_1, the adversary at location loc_1, the set of verifiers found at location loc_1, and the same respectively for location loc_2. Any of these values can be $\emptyset$.

Oracles Accessible to the Adversary $\mathcal{A}$. The adversary has read-only access to the prover-list PL, the verifier-list VL and the session-list *Sessions*.

The adversary can however inflict modifications on these lists via oracles; this emulates real-life adversarial activity.

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 oracles provided to $\mathcal{A}$ are defined below.

init($[P, V]$): This oracle simulates the start of new sessions for a prover-party with id P and/or for a verifier-party with the id V (i.e., both start executing the protocol). Either P or V can be omitted, in which case only one corresponding session is started. If P and/or V are inactive, this sets the status of P and/or V to active. Ch generates the new session identifier(s) V^i and/or P^j, and adds $(V^i, V, empty)$ and/or $(P^j, P, empty)$ to the *Sessions* list.

All the messages sent/received by P (resp. V) during its execution are added to the transcript of the corresponding prover session $(P^j, P, \cdot)$ (resp. verifier session $(V^i, V, \cdot)$) in the *Sessions* list. Once the session(s) is/are created, their identifiers are added in the relevant lists PL, VL, as well.

The session identifiers are given to $\mathcal{A}$ before the session starts, so that $\mathcal{A}$ can use them directly with the oracles described next, which take a session identifier as input.

term(E^j): This oracle permits to prematurely terminate a session E^j ran by a honest party (prover or verifier) with the identifier E. If the party E is inactive, then this oracle has no effect. If E is active, this oracle terminates the session E^j: E stops receiving/sending message in session E^j, and E^j is removed from PL if E is a prover, or from VL if E is a verifier.

move(P, l): This oracle is used to move a prover-party with the identifier P from one location to the other. If the status of P is inactive, Ch moves P to the location l, by updating $LOCL$.

send($\mathcal{A}, [E], [sid], m$): In this oracle, the following is the case: 1). $\mathcal{A} \in \{\mathcal{A}_{loc_1}, \mathcal{A}_{loc_2}\}$, E is a prover party, or verifier party, or $\mathcal{A}_{loc_1}$ or $\mathcal{A}_{loc_2}$; 2). *sid* is a session identifier and it used only when E is a prover party or verifier party; 3). m is a message. Every message sent by $\mathcal{A}$ is sent through this oracle. If E is $\mathcal{A}_{loc_1}$ or $\mathcal{A}_{loc_2}$, then the message is sent there. Otherwise, m is

[18] This set is the empty set is the prover-party is inactive.
[19] This set is the empty set is the verifier-party is inactive.

added to the transcript of either all sessions in which E is active (if sid is omitted) or just E's sid session (if sid is provided). These additions are applied to the corresponding transcript in the $Sessions$ list. Parameters E and sid are optional. If omitted, then the message m is broadcast.

We recall that messages sent by this oracle overwrite honest messages, as per the threat-model in Def. 6. For instance, $\mathcal{A}$ can send a single bit through this oracle at a given time such that it overwrites a given bit of a message sent by a honest party. Additionally, we recall that messages between adversaries (*i.e.*, from $\mathcal{A}_{loc_1}$ to $\mathcal{A}_{loc_2}$) are subject to time of flight.

block$(E, sid, \mathcal{B}, \{S^k \mid S$ **party** $\})$**:** In this oracle, E is a prover or a verifier identifier. Let $M = M_0 \dots M_k$ denote all the bits that the party E can send during a honest protocol execution. The oracle checks if E is active by seeing if session E^{sid} is found in PL or VL (depending if E is a prover or verifier identifier). If it is not, the oracle aborts. If E is active, then the oracle blocks the transmission of bits $\{M_i | i \in \mathcal{B}\}$, in such a way that only the parties S receive them, in their sessions $\{S^k\}$ – which the oracle will check for existence. The relevant transcripts in $Sessions$ are modified accordingly.

result(sid)**:** Ch checks if there is a terminated verifier-session $(sid, V, t) \in Sessions$ (i.e., if t contains out). If the session does not exist, Ch returns null. If the session exists, Ch returns (out, P), where out indicates whether V accepted the authentication and proximity-checking in session V^{sid} (i.e., out is as per $out_{\mathbb{V}}$ in Π), and P is either the identifier of the authenticated prover if out=1 or $null$ if out=0.

The last oracle is not useful to the adversary in practice, but allows to define the security goals in a more convenient way.

5 DB Security Properties in BoxDB

We formalise the security goals to be achieved in BoxDB.

5.1 Overview of Security Definitions

Our model considers three types of attacks. We define: ***secret extraction (SE)***, ***generalised mafia-fraud (GMF)***, and ***generalised distance-fraud (GDF)***. Secret extraction is a completely new property w.r.t. DB-security, whereas the others strengthen existing DB-security definitions. The adversarial settings for these are summarised in Table 2.

	loc_2	loc_1	Goal of $\mathcal{A}$
GMF	$\mathcal{A}_{loc_2}, P_{BB}(x), \{\mathcal{P}\}, V$	$\mathcal{A}_{loc_1}, d\mathbb{V}, \{\mathcal{P}\}, V$	$d\mathbb{V}$ accepts $P(x)$
GDF	$\mathcal{A}_{loc_2}, P_{WB}(x), \{\mathcal{P}\}, V$	$d\mathbb{V}, \{\mathcal{P}\}, V$	$d\mathbb{V}$ accepts $P(x)$
SE	$\emptyset$	$\mathcal{A}_{loc_1}, P_{BB(x)}, \{\mathcal{P}\}, V$	$\mathcal{A}$ outputs x

Table 2. General Security Properties in BoxDB, Their Universes of Location, Attack- ed/-ing Parties (in red) & Adversarial Goals.

First, in Table 2, for each property, we give the universe of locations and execution environment considered (i.e., the parties necessarily present in each location and their implementation type – BB or WB). Second, Table 2 gives the goal of the adversary in each attack. The notation is as follows:

– $P(x)$ denotes a prover-party being attacked or attacking the protocol, who has secret auths

x (which, in BoxDB, also identify it uniquely);
– $\{\mathcal{P}\}$ denotes a set of prover-parties that does not contain $P(x)$; by abuse of notation, we use the set $\{\mathcal{P}\}$ at both locations, yet the set $\{\mathcal{P}\}$ in location loc_1 is not the same as in location loc_2;
– if BB or WB does appear alongside a prover-party P, then this denotes their respective type of implementation;
– if BB, WB does not appear alongside a prover-party P, then this means that their implementation-type could be of either BB or WB; notably, in $\{\mathcal{P}\}$, one can have both BB and WW provers;
– $\mathcal{A}$ is the adversary;
– $d\mathbb{V}$ denotes the verifier to which $\mathcal{A}$ try to authenticate;
– V denotes any verifier other than $d\mathbb{V}$;
– in red, we have the attacking and the attacked parties.

5.2 Corruption Modes

For each of our security properties in Table 2 and formalised below, we consider *two corruption modes:* ***weak,*** and ***strong***. This are detailed below.

Weak corruption mode. In a *weak generalised distance-fraud* (***weak GDF***), the adversary learns only the long-term secrets of the far-away prover trying to authenticate, i.e., the attacking prover $P(x)$ is implemented in WB mode and he is the only prover implemented in this way; the other provers (in $\{\mathcal{P}\}$ in Table 2) are all implemented in BB manner, i.e., the attacker did not learn their respective keys.

In a *weak generalised mafia-fraud* (***weak GMF***), all provers are corrupted in BB mode, i.e., the man-in-the middle attacker is given no long-term secrets at all.

In a *weak secret extraction* (***weak SE***), all provers are implemented in BB mode, i.e., the secret-extraction attacker is given no secrets at all.

Strong corruption mode. In strong generalised mafia-fraud (***strong GMF***), the attacker is given access to the secrets of potentially all provers, except for the one being attacked: in Table 2, $\{\mathcal{P}\}$ can be some/all WB, and $P(x)$ is BB.

For *strong secret extraction* (***strong SE***), the same setting as in strong GMF is the case.

In a *strong generalised distance-fraud* (***strong GDF***), the adversary potentially knows the secrets of all provers, i.e., in Table 2, $\{\mathcal{P}\}$ can be some/all WB, and $P(x)$ is WB.

So, the two modes, weak and strong, cover classes of attack-scenarios determined by whether the attacker controls the secrets of provers others that the one being attacked, i.e., the provers in the set $\{\mathcal{P}\}$ in Table 2. Below, in Table 3, we show this in one snapshot.

	loc_2	loc_1	$\{\mathcal{P}\}$s can contain WB provers
strong GMF	$\mathcal{A}_{loc_2}, P_{BB}(x), \{\mathcal{P}\}, V$	$\mathcal{A}_{loc_1}, d\mathbb{V}, \{\mathcal{P}\}, V$	✓
weak GMF	$\mathcal{A}_{loc_2}, P_{BB}(x), \{\mathcal{P}\}, V$	$\mathcal{A}_{loc_1}, d\mathbb{V}, \{\mathcal{P}\}, V$	✗
strong GDF	$\mathcal{A}_{loc_2}, P_{WB}(x), \{\mathcal{P}\}, V$	$d\mathbb{V}, \{\mathcal{P}\}, V$	✓
weak GDF	$\mathcal{A}_{loc_2}, P_{WB}(x), \{\mathcal{P}\}, V$	$d\mathbb{V}, \{\mathcal{P}\}, V$	✗
strong SE	$\emptyset$	$\mathcal{A}_{loc_1}, P_{BB(x)}, \{\mathcal{P}\}, V$	✓
weak SE	$\emptyset$	$\mathcal{A}_{loc_1}, P_{BB(x)}, \{\mathcal{P}\}, V$	✗

Table 3. Weak/Strong Security Properties in BoxDB, Their Universes of Location, Attacked/Attacking Parties (in red).

Weak vs. Strong Corruptions. The weak corruption mode is reflects the implicit assumptions taken in traditional threat-models for distance fraud and mafia fraud (though our

fine-grained adversarial scenarios did not exist before). The strong corruption mode, however, allows to assess the robustness of a protocol, for instance if the adversary is allowed to register provers with a secret key of his choice. In such a context, consider –for instance– secret extraction: then, it is desirable to have strong secret-extraction rather than weak. Indeed, strong secret extraction denotes that knowing the secret keys of some provers should not make it easier to obtain the secret key of another one.

It is the strong corruption-modes that allow us to show the attack on against DB3 and the other 13 DB protocols in Section 3. Notably, the attacks we show are strong GDFs. And it is the fact that, in *strong* GDF, unlike in weak GDF or traditional distance-fraud settings, the attacker $\mathcal{A}$ has previously learnt the secret keys $x, y, \ldots$ of some provers $P(x)$, $P(y)$, etc., and this allows him to mount a DF-like attack and make prover $P(x)$ be authenticated from afar (by using his knowledge of y in the process).

5.3 Security Definition.

Secret extraction (for black-box provers). Black-box provers are meant to be tamper-proof, and a protocol that leaks their secret material (even if the cryptographic primitives themselves are secure) would violate this. DB protocols vulnerable to this type of leakage attacks, as shown in [8], are typically also consequently insecure in front of MiM. However, secret extraction does not necessarily imply that a MiM adversary can authenticate. For instance, one can transform a MiM-secure protocol into a protocol where a long-term key, not used for authentication, is sent in clear at the beginning. This protocol becomes vulnerable to a secret-extraction attack, as it reveals a long-term secret. However, since the key leaked is authentication-agnostic, the protocol is still MiM-resistant. As such, we formalise secret extraction as a standalone DB-security property.

Definition 9. Secret-extraction. *Let Π^{real} be the realisation of a DB protocol Π using cryptographic keys as idents, Π^{real} cast in a universe of locations s. that in one location there is an adversary, prover-devices and a set of verifiers, and there is no one in the second location. Let $\mathcal{P}'$ be a set of prover-devices in Π^{real}. A* secret-extraction game *$\mathcal{G}$ against the DB protocol Π is as follows:*

1. the challenger $\mathcal{C}h$ gives the adversary $\mathcal{A}$ access to all the oracles; 2. at some point, after no more than a polynomial number of oracle-calls, $\mathcal{A}$ chooses a prover $P \in \mathcal{P}'$; 3. $\mathcal{A}$ 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 attacked prover P is black-box. If all the other provers are black-box, then the above game is a weak secret-extraction game. *Otherwise, the above game 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 due to the protocols we analyse herein, we only focus on *idents* which are keys.

We now formalise generalised mafia-fraud in Definition 10.

Definition 10. Generalised Mafia-fraud. *Let Π^{real} be the realisation of a DB protocol Π in a universe of locations $\mathcal{U} = (loc_1, loc_2)$ such that:*

– loc_1 contains an adversary $\mathcal{A}_{loc_1}$, a designated verifier, denoted by $d\mathbb{V}$, as well as other verifiers

and a set of provers,
– loc_2 contains a set of provers, an adversary $\mathcal{A}_{loc_2}$ and a set of verifiers.

A generalised mafia-fraud (GMF) game $\mathcal{G}$ *against the DB protocol Π is as follows:*
– the challenger $\mathcal{C}$ gives $\mathcal{A}$ access to all the oracles;
– the game $\mathcal{G}$ is then two phases:
1. Learning phase. *at some point, after no more than a polynomial number of oracle-calls, $\mathcal{A}$ outputs a prover identifier P and a designated verifier $d\mathbb{V} \in V_{loc_1}$ and then the next phase begins*
2. Attack phase. *1. $\mathcal{C}$ terminates all sessions, and adds a marker to the session list Sessions to denote the start of the attack phase; 2. $\mathcal{A}$ loses access to the oracle* **move**, *but can call all the other oracles; 3. $\mathcal{C}$ checks the characteristics of P in PL. If P is close to $d\mathbb{V}$ (*i.e., *its location is loc_1), or if P 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: there must exist a verifier-session $(d\mathbb{V}, sid, t)$ started after the beginning of the attack phase such that $result(sid) = (true, P)$, i.e., there is a verifier-session in which $d\mathbb{V}$ accepted the far-away prover P during the attack phase. The* advantage of an adversary $\mathcal{A}$ in the GMF game *is his success probability α.*

If all provers are black-box, then the above game is a weak GMF game.

If at least one prover –which cannot be the attacked one P– is white-box, then the above game is a strong GMF game.

In this game's formalisation in Definition 10, 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, the adversary can place the target-prover close to the verifier, to potentially gain advantage in learning protocol/parties' secrets.

Our security property GMF generalises MiM attacks, where two adversaries collaborate to authenticate as a prover located outside of the distance bound. Also, specifically, a strong-GMF adversary is strictly stronger than previous MiM definitions.

We now formalise generalised distance-fraud in Definition 11.

Definition 11. Generalised Distance-fraud. *Let Π^{real} be the realisation of a DB protocol Π in a 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 set of provers,
– loc_2 contains a set of provers, an adversary $\mathcal{A}_{loc_2}$ and a set of verifiers.
Later, in the attack phase, $\mathcal{A}_{la}$ is removed from loc_1.

A generalised distance-fraud (GDF) game $\mathcal{G}$ *against the DB protocol Π is as follows:*
– the challenger $\mathcal{C}$ gives $\mathcal{A}$ access to all the oracles;
– the game is in the two phases below:
1. Learning phase. *at some point, after no more than a polynomial number of oracle-calls, $\mathcal{A}$ 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; then, the next phase begins*
2. Attack phase. *1. $\mathcal{C}$ terminates all sessions, and adds a marker to the session list Sessions to denote the start of the attack phase; 2. The adversary $\mathcal{A}_{loc_1}$ is removed from loc_1: only $\mathcal{A}_{loc_2}$ is left in loc_2, but can continue to call all the oracles; 3. $\mathcal{C}$ checks the location of P in $LOCL$. If P is close to $d\mathbb{V}$ (i.e., its position is 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: there must exist a verifier-session $(sid, d\mathbb{V}, t) \in Sessions$ started after the beginning of the attack phase, such that such that $result(sid) = (true, P)$, i.e., there is a verifier-session in which $d\mathbb{V}$ accepted the far-away prover P during the attack phase. The* advantage of an adversary $\mathcal{A}$ in the GDF game *is his success probability α.*

If all provers but P are black-box, then the above game is a weak GDF game*. Otherwise, it is a* strong GDF game*.*

Note that, in generalised distance-fraud, as made explicit by Definition 11 (and summarised in Table 2), the adversarially-chosen WB prover P is not allowed near to the verifier. Indeed, if P was nearby the verifier, $\mathcal{A}$ (via P) could simply run the protocol and win. Yet, other provers, possibly white-box, are allowed near the verifier. Recall that, as in our model, WB provers can have their memory read, but no prover-device can be modified. So, $\mathcal{A}$ cannot, for instance, replace the key of another white-box prover by the one of the attacking prover P.

The above also implies that our Definition 11 correctly covers the threat of distance-hijacking (DH) [20]. In fact, generalised distance-fraud captures all classical definitions of distance fraud and more. Weak generalised distance-fraud is similar to traditional formalisations of DF. Meanwhile, strong generalised distance-fraud allows the adversary to know the secret keys (at least of some if not all) the provers that are not actually used to mount the attack. Again, strong GDFs are used to show the attacks on DB3 and the 13 DB protocol in Section 3.

Secure DB in BoxDB. For each security property above, $SP \in \{SE, GDF, GMF\}$, a realised DB protocol Π^{real} is *SP-secure* if the advantage of the adversary in the corresponding SP game is negligible in the security parameter defining Π.

What Security Property Does One Need? We now look at which combination of the above security definitions would yield all-encompassing DB security. However, from this "all-encompassing DB security", we except terrorist-fraud resistance, as in Section 7 we show that we need not consider it.

– DB Security with Only Black-box Provers. If all the provers are guaranteed to be black-box, *i.e.*, no adversary can know the secret of any prover, then two relevant security properties are weak SE and weak GMF. In this case, secret extraction (SE) is also crucial: if the provers are designed to be black-box/tamper-proof, then the protocol should never leak there secret key.

Yet, if all the provers are guaranteed to be black-box, GDF however does not need to be considered. This is because, in this case, GMF is more generic than distance fraud. Indeed, in a GMF, the distant adversary $\mathcal{A}_{loc_2}$ is allowed extra help through the collaboration of $\mathcal{A}_{loc_1}$ (whereas in GDF, $\mathcal{A}$ is "alone" in $\mathcal{A}_{loc_2}$).

– DB Security with Some White-box Provers. If some provers are white-box, while the majority is black-box, then we require strong SE security. This is so since knowing some secret keys should not lead to the adversary learning others.

Due to the presence of white-box provers, strong GMF should also be analysed.

Finally, GDF should be considered. Indeed, ifamongst the provers registered in the system at least one is WB, then this prover can be already used mount a GDF. So, if only one registered prover is white-box, then weak GDF should be considered. If more than one prover is WB, then the attacker could exploit one close to the verifier $d\mathbb{V}$ and one far from $d\mathbb{V}$. As such, more than one prover is WB, then strong GDF should be considered.

– DB Security with Only White-box Provers. If all provers are WB (i.e., the adversary can extract all secret keys), then some security can still be preserved through strong GDF.

However, GMF and SE become irrelevant, as there are no black-box provers to attack. This discussion is summed up in Table 4.

All provers are BB	weak GMF, weak SE
Some provers are WB, some are BB	strong GMF, strong GDF, strong SE
All provers are WB	strong GDF

Table 4. Relevant properties for different corruption scenarios

Even though it is possible to use our model for cases where all provers are black-box, we strongly advocate to consider that some provers might be white-box (e.g., a mobile phone running the EMV protocol). Moreover, any attack on the tamper-proofness of a device is likely to remain undisclosed. So, it is sensible to consider that some provers might be white-box in most applications (as other DB models [14,29] do) and strengthen further the security analysis accordingly, i.e., largely, prove security using the strong variants of our security properties (to avoid attacks like the ones we showed herein).

6 Security Analyses of EMV-RRP

In Section 3, we showed that the contactless payment protocol EMV with relay protection `EMV-RRPv2` is vulnerable to proximity-based attacks. In fact, that attack is a generalised distance-fraud (GDF), in our BoxDB. Now, we first propose a minor modification of the `EMV-RRPv2` into a protocol called *`PayPass+`*. Then, we prove `PayPass+` fully secure in our strongest corruption model.

6.1 Securing Contactless Payments

Our new protocol `PayPass+` is depicted in Fig. 6.

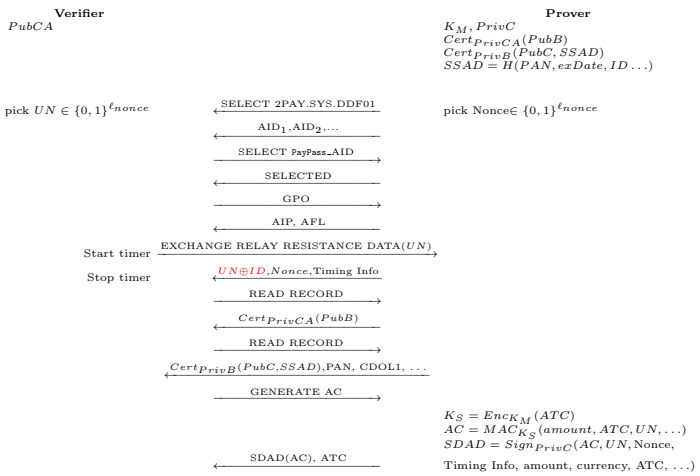

Fig. 6. The `PayPass+` Contactless Payment and DB Protocol.

It differs from EMV-RRPv2 only in that, during the timed phase, the card responds with $(UN \oplus ID, Nonce,$ Timing Info) instead of just $(Nonce,$ Timing Info), i.e., $UN \oplus ID$ is added; this is marked with red on Figure 6. The bitstring ID is a public, pseudo-unique identifier (as defined in Definition 15). Also, this ID is written on the card and in its $SSAD$. Therefore, the card's certificate certifies this ID too. Moreover, the lengths of nonces become 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. 5.

Secret-Extraction Security. First note that, in PayPass+, the prover has 2 cryptographic keys, K_M and $PrivC$, that are of interest for our secret-extraction definition introduced in Sec. 5. Each of them is used only once: one in an encryption and one in a signature. On the way to prove our secret-extraction property, we first need to introduce a new concept: *secret recovery (SR)*. Then, we recall the GMR-SKS notion used in signature-security [42].

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.* [9], 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 12. ***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 [42]. 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'.

Using all these two notions we can prove strong secret-extraction security for PayPass+, We also prove its security w.r.t. strong GMF and strong GDF.

Theorem 13. ***Secret-Extraction Security.*** *Let S and E respectively denote the signature and encryption scheme used in EMV-RRPv2. If the E is SR secure, and S is GMR-SKS secure, then PayPass+ is strong secret-extraction secure.*

Capturing Classical DB Security. Now we look to capture and generalise classical DB-security definitions.

Theorem 14. ***Generalised Mafia-Fraud Security.*** *Let S be the signature scheme in* `EMV-RRPv2`*. If S is GMR-SKS secure, then* `PayPass+` *is strong generalised mafia-fraud secure.*

To prove GMF resistance, we need the ID values in *prot* be *pseudo-unique*, *i.e.*, generated such that any two IDs have a high Hamming distance. We first formalise this notion.

Definition 15. ***Pseudo-unique identifiers.*** *A set of identifiers $\mathcal{I}$ has the pseudo-unique property if, for any pair of identifiers $(a, b) \in \mathcal{I}^2$ such that $d =$* `HammningDistance`(a, b)*, it holds that 2^{-d} is negligible in a givne security parameter.*

Pseudo-unique identifiers can be instantiated with Reed-Solomon codes [47]: 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 16. ***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.*

All proofs are given in Appendix A.

7 "Boxed" Provers and Terrorist Frauds

In this section, we show that TF-analysis need not be considered in any formal DB-security model[20] which, like ours, distinguishes between white-box and black-box prover-corruption. We do this with a structured but high-level discussion, as we actively avoid formalising TF in BoxDB.

Terrorist-fraud (TF) resistance nurtured long-lasting controversies in distance bounding. The first protocol seemingly resistant to TF to be proposed in [17], 20 years after TF was defined in [23]. However, this protocol was later proven TF-insecure [33]. Following that, many definitions of terrorist-fraud resistance appeared – in a quest to find TF-resistant constructions in provable way, e.g. [30,**?**,15,**?**,5,29]. These TF definitions differ, are sometimes incomparable, and the resulting designs are often unnatural [5] or based on non-standard assumptions [14,15]. This section settles these debates.

Recall that a TF attacker is successful if any help that a distant, rogue prover-owner P^*, using a prover-device P, can give an accomplice $\mathcal{A}_{TF}$ is either insufficient to authenticate or it permits $\mathcal{A}_{TF}$ to impersonate P^* in further sessions.

No Terrorist Frauds with Black-box Provers If the prover-device P is black-box, then P^* cannot extract any meaningful help from it to give to $\mathcal{A}_{TF}$, besides what can be learnt from a honest protocol execution. This help is therefore trivial, and does not give any significant advantage to $\mathcal{A}_{TF}$.

Another way of stating the above result is the following. By interacting with this BB P, P^* and any adversary learn exactly the same (by the def. of BB provers in Def. 1). Therefore,

[20] TF may still be relevant in other corruption models where provers can be manipulated in ways between WB and BB.

P^* can be seen as an adversary, so that a TF experiment where P is black-box is equivalent to a MiM attack.

So, from the above, if the provers are black-box, then the following is the case:
— terrorist-fraud resistance is conceptually irrelevant, as no non-trivial help can be given by P^*;
— the security notion of terrorist-fraud resistance is shown to be redundant, as it is equivalent to a MiM attack.

Always Terrorist Frauds with White-box Provers Now assume that P is a white-box prover-device. So, the rogue prover-owner P^* now knows the secret information of P and P's algorithm is public (as per the DB-specification definition). Thus, P^* can write these inside a tamper-proof, black-box device. This leads to the "all-reaching" TF attack below.

The White-box Terrorist-fraud (WB-TF). The rogue owner P^* of the WB prover-device P produces a black-box copy of P, further referred to as a terrorist–device. The terrorist–device is programmed to self-destruct (*i.e.*, wipe its memory) after one successful authentication. The rogue owner P^* passes the terrorist–device on to $\mathcal{A}$. Then, $\mathcal{A}$ (or the part of $\mathcal{A}$ that is close to V) executes P-alg via terrorist–device to authenticate. Since terrorist–device is black-box, $\mathcal{A}$ learns nothing other than in a honest session of the protocol. Also, as terrorist–device wipes its memory after one use, $\mathcal{A}$ cannot use it to authenticate more than once.

The WB-TF attack above works on all protocols of the literature. WB-TF circumvents the formal models, such as [13,25], as those did not consider a black-box clones as a means towards a TF-attack.

Thus, via the WB-TF attack, we obtain the following result. If white-box provers can be copied into black-box provers, then a terrorist-fraud attack is always possible. This is with the exception of certain cases which are un-important herein. These are:
(a) if protocols identify the holders of devices (which we already argued as a measure that part-defeats the purpose of DB);
(b) if protocols are not MiM-resistant (which is un-interesting as that would make the protocols already widely insecure). Also, in this case, the help of P^* is irrelevant, as $\mathcal{A}$ can authenticate without it.

More on 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 P during the authentication. However, this behaviour can be emulated with a variant WB-TF. To this end, we can integrate a remote activation/deactivation mechanism to the terrorist–device, so that it only works when P^* intends it to. Adding a remote activation mechanism to the terrorist–device even permits to perform more advanced types of terrorist frauds. For instance, P^* 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. 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.

Also, the 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, it could be argued that WB-TF is equivalent to P being near the verifier. However, the terrorist–device can embed any algorithm, it does not necessarily need to be an exact copy of P. For instance, it can implement the algorithm ran by $\mathcal{A}_{TF}$ in any of the terrorist frauds published in the literature. In this sense, our attack generalises terrorist fraud.

The above TF-attacks using WB provers not only apply to all traditional DB protocols (which use cryptographic keys), but they apply to the two existing DB protocols with PUFs [38,35] as well. This is by considering these PUF-based DB protocols in the *"simulatable PUF"* [48] model. In this model, the PUF is indistinguishably replaced by a PRF, for instance by a rogue manufacturer. P^* can therefore copy the key to this PRF inside the terrorist–device. We believe that protocols using PUFs should be considered in this model with regards to white-box provers, in order to capture white-box corruption: otherwise, the prover is simply black-box, and therefore, trivially TF-resistant [35].

8 Discussions and Conclusions

Throughout the manuscript we compared our properties and model with all existing DB lines, be it for computational security [14,25,29] or for symbolic verification [22,41,18,21,20]. As such, now we discuss different aspects.

A BoxDB for Computer-Aided cryptography. Our model is designed with automated verification in mind. Hence, we carefully define how the challenger implements the oracles he provides to the adversary. First, oracle-based constructions lend themselves to machine-checked cryptographic proofs, e.g., in tools like `EasyCrypt` [7]. Second, the system of locations (e.g., the two locations in a universe of locations) also eases automatic verification (compared to the two main cryptographic models for DB [14,25] where the expression of distance is respectively too fine vs. implicit). Moreover, our security definitions were given, on top of the challenger and its list, are defined in a manner that makes reasonings in tools like `EasyCrypt` [7] easier. Third, BoxDB has the advantage (which matters in machined-checked analysis) that to ascertain provable DB-security in the classical sense, one has to carry out cryptographic proofs for only 2 standard DB properties –GDF and GMF (as opposed to 3, 4 or more in other models [14,25]). Lastly, the session-based aspects of BoxDB make the modelling in tools like `EasyCrypt` unnatural. However, note that our session-based approach is very refined (compared to other session-based DB models [25]): we keep "logs" of all session-based activities in a series of well-coordinated lists. In fact, this also stemmed from the need to model the use of sessions in `EasyCrypt`. In future work, we will develop a full mechanisation of BoxDB in `EasyCrypt`.

On Terrorist-Fraud Resistance. In Section 7, we argue that formal models, including ours, should discard the analysis of this terrorist fraud. Kilinc and Vaudenay [36] also looked at the equivalence between terrorist fraud and MiM attacks in the black-box corruption-model. More specifically, they consider a secure hardware module (SHM) that performs the cryptographic operations upon provers' request. This is very similar to an adversary interacting with a black-box prover in our setting. A notable difference is that their model seems to possibly allow protocols in which the prover picks the nonce instead of the HSM (even though they only give examples where the HSM picks the provers' nonces). Their conclusions differ from ours: 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 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 need no longer be a DB-security requirement. Yet, Kilinc and Vaudenay's work opens perspectives on a form of "grey-box" corruption, in which, for instance, P (possibly manipulated by $\mathcal{A}$) picks the nonces, while the other cryptographic operations are performed in a black-box manner. However, such a "grey-box" corruption seems of limited relevance in practice, as the choice of random values should be considered as a cryptographically critical operation and left to the HSM. Nonetheless, a variant of this "grey-box" corruption, cast in the Dolev-Yao [24] setting, is already used in symbolic verification of DB [18], however not for TF-resistance but for DF-analysis.

Conclusions. We presented new attacks against variants of contactless-payments EMV-RRP protocols, as well as 13 existing distance-bounding protocols. For this, we use new attack strategies inexistent in other DB-security models. So, we proposed a new formal security model for DB, BoxDB. Unlike traditional DB models which generally only consider white-box prover, BoxDB explicitly distinguishes white- vs black-box provers. BoxDB also provides fine-tuned threat levels: weak and strong flavours of each security property it advances. We proposed a variant of the EMV-RRP protocol that is provably secure in the strongest adversary model of BoxDB.

Some aspects are left for future work. (1) Privacy. some DB protocols consider privacy-protection for the provers. It would be interesting to include such privacy properties in an extension of BoxDB. (2) Grey-box provers. We only consider black-box or white-box provers. Other models are possible, and should be investigated, where the adversary can query e.g., a hardware module to perform part of the cryptographic operations, but does not have access to the key, or only has access to one of several keys used by the prover. (3) Automatic verification. While $BoxDB$ is designed with computer-aided cryptographic proofs in mind, its actual mechanisation inside verification tools is yet to occur.

Our model opens for new research in distance bounding. For instance, we proposed new generalised distance-fraud attacks on some protocols, but the list is not exhaustive. Also, strong GMF may lead to new types of attacks. Finally, our proposition to stop considering terrorist-fraud (TF) resistance in DB removes the difficult task of adding provable resistance to TF. This will lead to new, simpler and more efficient DB protocols.

References

1. D. Adamy. *A First Course in Electronic Warfare*. Boston, Mass. ; London : Artech House, 2001. Includes index.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P. Strub. Easycrypt: A tutorial. In *Proceedings of FOSAD 2013*, pages 146–166, 2013.
8. 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.
9. 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.
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. L. Bussard and W. Bagga. Distance-bounding proof of knowledge to avoid real-time attacks. In *Proc. of Security and Privacy in the Age of Ubiquitous Computing*, IFIP International Federation for Information Processing, pages 222–238. Springer-Verlag, 2005.
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. A. Debant and S. Delaune. Symbolic verification of distance bounding protocols. Research report, Univ Rennes, CNRS, IRISA, France, Feb. 2019.

22. A. Debant, S. Delaune, and C. Wiedling. Proving physical proximity using symbolic models. Research report, Univ Rennes, CNRS, IRISA, France, Feb. 2018.
23. Y. Desmedt. Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity and how to overcome them. In *SecuriCom*, pages 15–17. SEDEP Paris, France, 1988.
24. D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactionson Information Theory 29*, 29(2), 1983.
25. 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.
26. EMVCo. Book C-2 kernel 2 specification v2.7. EMV contactless specifications for payment system. `www.emvco.com/wp-content/plugins/pmpro-customizations/oy-getfile.php?u=/wp-content/uploads/documents/C-7\_Kernel\_7\_V\_2\_7\_Final.pdf`, Feb, 2018.
27. R. Entezari, H. Bahramgiri, and M. Tajamolian. A mafia and distance fraud high-resistance rfid distance bounding protocol. In *ISCISC*, pages 67–72, 2014.
28. 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.
29. M. Fischlin and C. Onete. Terrorism in distance bounding: modeling terrorist-fraud resistance. In *International Conference on Applied Cryptography and Network Security*, pages 414–431. Springer, 2013.
30. M. Fischlin and C. Onete. Terrorism in distance bounding: Modeling terrorist fraud resistance. In *Proceedings of ACNS 2013*, volume 7954 of *LNCS*, pages 414–431. Springer Verlag, 2013.
31. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 148–160, New York, NY, USA, 2002. ACM.
32. 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.
33. G. Hancke. Distance-bounding for RFID: effectiveness of 'terrorist fraud' in the presence of bit errors. In *2012 IEEE International Conference on RFID-Technologies and Applications, RFID-TA 2012, Nice, France, November 5-7, 2012*, pages 91–96, 2012.
34. G. P. Hancke and M. G. Kuhn. An RFID distance bounding protocol. In *Proceedings of SecureComm 2005*, pages 67–73. IEEE, 2005.
35. 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.
36. 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.
37. 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.
38. 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.
39. S. Lee, J. S. Kim, S. J. Hong, and J. Kim. Distance bounding with delayed responses. *IEEE Communications Letters*, 16(9):1478–1481, 2012.
40. MasterCard. Contactless paypass reader specifications v3.1,. not publically available, 2017.
41. 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.
42. A. Menezes and N. P. Smart. Security of signature schemes in a multi-user setting. *Des. Codes Cryptography*, 33(3):261–274, 2004.
43. 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.
44. 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.
45. R. A. Poisel. *Modern Communications Jamming Principles and Techniques*. Artech House, Inc., Norwood, MA, USA, 2nd edition, 2011.

46. 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.
47. 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.
48. 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.
49. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. `http://eprint.iacr.org/2004/332`, 2004.
50. L. C. Tom Chothia, Ioana Boureanu. Making contactless emv payments robust against rogue readers colluding with relay attackers. In *the 23rd International Conference on Financial Cryptography and Data Security (Financial Crypto 2019)*, 2019, to appear.
51. 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.
52. R. Trujillo-Rasua, B. Martin, and G. Avoine. Distance-bounding facing both mafia and distance frauds: Technical report. *CoRR*, abs/1405.5704, 2014.
53. 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.

A Main Security Proofs

Proof of Theorem 13.

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 $\frac{p^{\mathcal{A}}}{q_p}$. □

Proof of Theorem 14.

Proof. This is a game-based proof [49]

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. □

Proof of Theorem 16.

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 $UN' \oplus ID_P = UN \oplus ID_{P^{\mathcal{A}}}$ (where $P^{\mathcal{A}}$ is the prover chosen by $\mathcal{A}$ during the learning phase), with UN 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 $ID_{P^{\mathcal{A}}} \oplus UN$. 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 $x = HD(ID_{P^{\mathcal{A}}}, ID_P)$. 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. □