Fine-Grained & Application-Ready Distance-Bounding Security

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

[1] University of Surrey, UK
[2] Université de Clermont-Auvergne, France

Abstract. Distance-bounding (DB) protocols are being adopted in different applications, e.g., contactless payments, keyless entries. For DB to be application-ready, "pick-and-choose" corruption models and clear-cut security definitions in DB are needed. Yet, this is virtually impossible using the four existing formalisms for distance-bounding (DB), whereby each considers around five different security properties, arguably intertwined and hard to compare amongst each other.

In particular, terrorist-fraud resistance has been notoriously problematic to formalise in DB. Also, achieving this property, often weakness a protocol's general security. We demonstrate that –in fact– terrorist-fraud resistance cannot be achieved, under standard assumptions made for DB protocols. Our result wraps up terrorist-fraud resistance in provable-security in DB.

As a consequence of terrorist-fraud resistance being made irrelevant, and to address application-ready DB, we present a new, provable-security model for distance-bounding. It formalises fine-grained corruption-modes (i.e., white-box and black-box corrupted provers) and this allows for clearer security definitions driven by the separation in corruption-modes. Also, our model explicitly includes a security-property generalising key-leakage, which per se –before this– was studied only implicitly or as a by-product of other DB-security properties. In all, our formalism only requires three, clear-cut security definitions which can be "picked and chosen" based on the application-driven prover-corruption modes.

1 Introduction

Relaying between two legitimate entities Alice and Bob is an attack whereby a man-in-the-middle Charlie sends Alice's messages to Bob and/or Bob's messages to Alice, unbeknown to them, so that Charlie obtains a facility meant for Alice and granted by Bob or vice-versa. One such privilege could be that the malicious Charlie achieves to be incorrectly authenticated by Bob, as if Charlie was Alice. Another example of a facility that could be illicitly gained by Charlie via relaying is that of Charlie fraudulently spending the funds associated to Alice's bank-card at an electronic-payment terminal embodied by Bob. Certainly, the most widely used electronic-payment protocol, namely EMV (Europay, Mastercard and Visa), in their contactless version, is susceptible to relay attacks [13]. Moreover, last year, relays in passive keyless entry systems (PKES) in cars leading to car-theft made international news[3]. Clearly, relay-attacks' heaven is contactless communication, where due to the spontaneous initiation of the transmission and the lack of user inputs, a relay attack is totally transparent. Meanwhile, relay attacks are notoriously hard to detect and deter, as they subvert all cryptographic mechanisms potentially employed in the underlying protocols.

Yet, an attacker Charlie relaying between parties Alice and Bob generally introduces delays to the expected round-trip times of exchanges between Alice and Bob. Indeed, imposing an upper-bound on the round-trip times (RTTs) of message-exchanges is a known, physical-layer mechanism for effectively lowering the probability of successful relay attacks. This mechanism is often referred to as *proximity-checking*: i.e., a prover tag (RFID card, smartcard, NFC-enabled device, . . .) should prove that they are within a short distance from a given verifier/reader. Often,

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

within the proximity-proof, the former also authenticate themselves to the latter. The primitive where *proximity-checking composed with a unilateral-authentication mechanism* bares the name of *distance-bounding* (DB) was introduced by Brands and Chaum [12] to combat relay attacks against ATM systems.

A Recap of the DB Threats. In DB protocols, a tag (RFID card, smart card, etc.) should prove an upper-bound on the distance between them and a reader, and authenticate themselves in front of this reader. The authentication part is generally based on a pre-established secret, such as a cryptographic key hard-coded on the tag and retrievable by the reader from a backend database. In DB, the tag and the reader are often referred to as the *prover* and the *verifier*. In the vast literature covering such protocols [1], three "classical" types of possible attacks have been distinguished. In a *distance-fraud* (DF), a dishonest prover P^* tries to convince that he is within the distance-bound when in fact he is not. A *mafia-fraud* (MF) attack involves three entities: a honest prover P, found far-away from an honest verifier, and an adversary; the latter communicates to make the verifier believe the prover is in the verifier's proximity and authenticate as this prover. Finally, in a *terrorist-fraud* (TF), a dishonest prover P^* positioned out of the distance bound from the verifier colludes with an accomplice, so that this accomplice authenticates as P. Since hardly anything can be done if the prover is willing to give permanent access to his accomplice, for instance by giving him his device or his credentials, only provers who do not want to leak their credentials are considered. Hence, the attack is valid if P^* colludes in such a way that it does not disclose long-term secret material, such as keys, or any other information that may facilitate later impersonations of the tag by the accomplice. In recent years, the DB threat-model has been widened. [15] coined *impersonation attacks*, and [14] advanced a strengthening of distance-frauds under the name of *distance-hijacking* (i.e., the dishonest, out-of-bound prover P^* mounts his fraudulent proximity-proofs by abusing honest provers located close to the verifier). And, [10] gives further generalisations of these all: e.g., mafia-fraud captured via more generic and more powerful *man-in-the-middle (MiM)* attacks, and terrorist-fraud up-cast to *collusion-fraud* (whereby if the prover colludes repeatedly he eventually leaks his long-term secret key).

The Rise of Application-Specific Distance-Bounding. As even application-specific relaying becomes an extremely accessible fraud (i.e., relay-kits for defeating PKES cost just tens of dollars[4]), proximity-checking as well as distance-bounding are incorporated in various ubiquitous applications. For a decade, we had proximity-checking available in commercial access-control solutions e.g., via the Mifare Plus cards/readers from NXP. In the last five years, the solutions from 3DB Technologies (`www.3db-access.com`) offer not just proximity-checking, but distance-bounding, in versions of their "3DB6830 UWB IC" product to be used in e.g., PKES. Moreover, proximity-checking is present in Mastercard's latest specification of the contactless version of its EMV (Europay, Mastercard and Visa) protocol called PayPass [19]. Whilst PayPass v3.1 has been augmented just with proximity-checking, the unilateral authentication of the card to the payment terminal is already present in EMV and therefore in all previous versions of PayPass. So, one can argue that the security-threats specific to distance-bounding apply *entirely* to PayPass v3.1. And, the likelihood is that with the advent of 5G (the 5th generation of mobile network) and with it of autonomous driving, connected vehicles, smart cities, "massive" IoT, proximity-checking and distance-bounding, as well as generalisation thereof will be incorporated in even more applications.

Our Thesis. With this rise of application-specific DB makes, the formal DB-security should be firstly closely revisited, and secondly re-appointed, in such a way that "pick-and-choose" distinctions are made in the threat/corruptions' model. These distinctions can aid an application-

[4] `https://www.wired.com/2017/04/just-pair-11-radio-gadgets-can-steal-car/`

designer make informed decisions as to which type of threat/corruption is aligned to his/her application and consider it therein.

Organisation & Contributions. In Section 2 we give our critical view on the main issues of DB security (motivating this line), and in Section 2.4, we discuss related work. Our contributions are as follows: (1) we show that, under reasonable assumptions, terrorist-fraud resistance cannot actually be achieved (Sect.3); besides, we also introduce a new type of terrorist-fraud resistance, which we call `Real-Life Terrorist-Fraud`-resistance. (2) we give a new DB model that treats corruptions in a fine-grained manner (Section 4), yielding clear-cut DB-security properties (Section 5); we show how to carry out proofs in our new model; (3) our model treats key-leakage explicitly, which –despite the fact that DB is an authentication primitive and key-leakage attacks have been exhibited– has not been explicitly treated in DB security.

2 Revisiting Formal DB Security

Now, we focus on three areas of *formal* DB-security: terrorist-fraud security, key-leakage, fine-grained prover-corruptions. The first is unsettled and the last two have been shortchanged in *formal* DB-security. In this paper, we also overcome these limitations.

2.1 Wrapping Up Terrorist-fraud Resistance

One can start by arguing the legitimacy of requiring that a protocol attains TF resistance in the first place, by claiming that TF is a "too-exotic" threat to worry about, and that the assumption that the dishonest prover wants to help while retaining his secret key is too strong. But, in provable security, this argument per se will not hold enough weight. In provable-security models, resistance to terrorist-fraud should encode the following: "*whatever* malicious, far-away prover P^*, *whatever* adversary A, if P^* helps A to succeed in proving that P^* is close to a verifier V, then *there exists* a future execution by A in which the said help gives A *some* advantage to authenticate as P when it comes A doing it unaided". This property is not easy to disprove, not in the way of "standard" provable-security where to show an attack one would simply exhibit some attacker whose execution succeeds with a probability that is not negligible. This hurdle entails three shortcomings of DB-security, which we describe below and number (A) to (C).

(A). Firstly, to attain TF-resistance, irrespective of its formal formulation, one needs to construct a protocol where some help by any dishonest P^* gives some future advantage to a MiM attack. This in itself is not trivial. It intuitively leads to either a protocol with some backdoor [3, 16] (which generally lowers MiM-resistance per se), or in designs whereby the dishonest help will leak bits of the prover's secret key which also tend to be more communication-expensive than necessary (e.g., larger sizes of challenges [4]). Even so, Hancke disproved the alleged TF-resistance of this protocol in [4] with multi-bit challenges, in cases where there is communication noise [17]. Then, to ensure such TF-protection in provable-security setting (all settings consider, including noise), one can end up with rather convoluted notions and designs: e.g., see the notion go "leakage scheme" used in the protocols in [10].

(B). Secondly, there exist multiple formal definitions [10, 11, 15, 16] of TF-resistance, which often do not express the same security statement or they are even not fully comparable due to differences in the adversarial communication model (e.g., simTF in [15] w.r.t. collusion-resistance in [10]).

(C). Thirdly, these formal definitions have, to some extend, been tailored such that one proves the security of specific designs – which is not the way in which security definitions should come about. This seems to be the case of *collussion fraud* in [10] and/or (γ,γ',m)*-soundness* in [11], which are used to prove resistance to a notion slightly stronger than TF in the protocols called

SKI and DBOpt, respectively (as it appears that to achieve all-encompassing security under those designs, these notions were needed). Moreover, since the proofs of security/(in)security bears the "non-falsifiable" character mentioned above, security proofs and attacks in these models are hard to follow and certainly error-prone.

Our Highlight on TF-resistance. Whilst in provable-security for DB, this TF-resistance narrative continues both on attaining TF-resistant designs and on formalising the notion, we cannot but wonder: is it possible to settle this matter once and for all? Better still, is it possible to *formally prove* that –under reasonable assumptions– TF-resistance can simply not be ensured? In Section 3, we show that the answer to this question is "yes": one can do away with endeavouring in formalising or attaining TF-resistance.

2.2 Shading Light on Key-Leakage in Formal DB-Security

Key-leakage/key-learning is an attack whereby a MiM learns (part of) the secret-key of the authenticating party, hence being a vulnerability that traditionally concerns authentication primitives. But, as we explained before, DB is in a composition of an authentication primitive and a proximity-checking primitive. Indeed, active attacks resulting in key-leakage have been shown in various DB protocols, e.g., see [7]. Yet, the way key-leakage has been treated within DB security raises three intertwined issues, noted as (A) to (C) below.

(A). In some DB-security models/frameworks, key-leakage is sadly altogether excluded as a threat for DB protocol. That is, distance-based attacks are considered valid only if the secret-key does not leak (see the black-box prover-model in [2]).

(B). Yet, the aforementioned thesis is surely doubtful. This is because one can show a number of cases where distance-frauds and/or mafia-fraud will have as a by-product the (partial) leakage of the prover's key. For instance, this is the case of distance-fraud attack in [8] against the DB protocol [4] whereby as the dishonest prover mounts this valid attack, he produces a protocol-transcript that blatantly leaks his key to any attacker who is simply observing the execution.

To this end, in the newest DB formalisms [10,16], security against key-leakage is included, albeit not explicitly. Still, key-leakage attacks can be found using these models as they are implicitly covered by the notions of MiM security (e.g., see the "learning phase" in MiM-security in [10]).

However, allowing key-leakage as a possible action during distance-based frauds yields security requirements baring a mixture of authentication and proximity-based properties which is not easy to treat in proofs.

(C). Whilst a DF/MF can lead to the prover's key leaking, one should also stress that distance-based attacks, such as DF, do not necessarily imply key-compromise. For instance, a dishonest prover can mount a DF by simply answering early if timed responses to do not depend on the challenges.

Our Highlight on Key-Leakage in Formal DB. Given the above in conjunction with the fact that DB is being adopted in different applications (some of which may be particularly concerned if key-learning occurs), a formal DB-security model should treat key-leakage explicitly as well as separately, i.e., as a standalone threat. And, we pursue this view in our model herein.

2.3 Finessing Formal DB-security w.r.t. Prover-Corruption

A *white-box* (WB) prover has full access to its algorithm and its inner data. A *black-box* (WB) prover has no access to its algorithm or its inner data.

These types of dishonest provers appeared in [2]. However, [2] showed no formals means of embedding these different corruption-modes into the larger threat-model, towards formal security definitions: i.e., how to formalise an attacker that can manipulate several such corrupted provers, some in WB, some in BB "mode". Also, [2] showed no model for carrying out proofs or security analyses using these corruption modes formally.

Moreover, [2] did not explore these notions correctly. E.g., [2] states that a DF is valid only if the key does not leak. Yet, as aforementioned, this is wrong: [8] shows a valid distance-fraud attack exercised by (what is in fact a white-box) prover against the DB protocol [4] whereby the provers produces a protocol-transcript that leaks his key to any attacker who is simply observing the execution.

In fact, the implications/separations in DB-security given by using WB vs BB provers have not been incorporated into any (other) formal frameworks/model for DB either.

Our Highlight on Prover Corruption in Formal DB. DB security refined around WB/BB provers would move DB-security closer towards the finesse needed for application-aware security analyses, where the corruption model varies from application to application. E.g., one may dismantle a key-fob in the way of WB provers, but would not do so for a bankcard – in the way of BB provers. Or one may "open up" some smartcards in an access-control application, but not all – i.e., have WB provers coexist next to BB. ones. We will pursue these view herein.

2.4 Our DB-Revisiting Methodology. (Related Work)

First, we are inspired by the communication model and explicit time-modelling like Boureanu et al. [10]. Second, in the resulting interactive-proof-like model, we single out the Bellare-Rogaway notion of protocol-session, as Dürholz [15] did. But, unlike [15], we do not use sessions to define the communications or an induced time-model; we only use sessions as inputs and outputs to adversarial oracles. Third, the adversarial oracles we introduce can be used to set up what we call a *(particular) DB environment*, i.e., a multi-party, multi-instance, concurrent execution of a DB protocols, with instances placed at positions which can be decided by the attacker. Moreover, these oracles allow for fine-grained corruptions, i.e., provers be controlled in the white-box (WB) or in the black-box (BB) mode. Whilst the notions of WB/BB provers in DB appeared in [2] and each one was explored in isolation, we use them in a formal setting, to build a complete threat-model which can combine both in advanced ways. The particular links between a terrorist-fraud notion we use herein and established ones is discussed in Section 3.2. Using our oracles and the notion of DB-environment, our DB security-definitions appear natural. We give a new security-definition for *key-leakage*, with active MiM attackers in mind. So, by combining elements from existing DB frameworks [2, 10, 15], we yield a model that captures DB-security via a small number of clear, security definitions, which can be "picked and chosen" by application-designer, upon the threats that their proving devices can be exposed to.

3 Fine-Grained Provers: No TF-Resistance Needed.

We now discuss DB from the viewpoint of proving devices and their holders, and their respective identifiability in front of verifiers. We start from the observation that the distinction between white-box and black-box provers (which we also pursue formally in Section 5) leads us to a systematic analysis of the (im)possibility to achieve terrorist-fraud resistance. We show that "*classical terrorist-fraud resistance*" is impossible with white-box provers in most DB protocols, i.e., all where the prover algorithms do not use PUFs. The case of white-box provers in protocols uses PUFs is discussed in the appendix. We then show that in the case of black-box prover a new and convenient type of terrorist-fraud resistance can always be achieved.

3.1 Fine-Grained Provers, Non-Identifiable Prover Holders & Tamper-Proof Devices

Fine-Grained Provers. We assume that the provers can either be *white-box* or *black-box*. Provers have holders, i.e., a person operating them. The holder of a whitebox prover knows its secret material, and can implement any algorithm of his choice on the device. Instead, black-box provers are completely tamper proof, and only execute their predefined algorithm, without the holder being able to modify it or to extract the secret material. Indeed, white-box (WB) provers are typically employed in the distance-bounding literature, such as in the formal models [10, 15], and in most articles presenting new protocols/attacks [8, 11]. But, in industry, for instance in the case of the EMV protocol, the provers are assumed to be black-box (BB), i.e., tamper-proof [19].

Tamper-proof Devices: A Plausible DB Assumption. As explained above, it is commonplace in "academic DB" to consider WB provers and it is commonplace in the industrial place to consider tamper-proof provers. As such, if one is ready to employ a threat-model where dishonest provers can manipulate their devices at the hardware level (i.e., the white-box model), then arguably one should be ready to accept security solutions and vulnerabilities based on hardware setups and the use tamper-proof device. So, in this section, we make what we believe to be a plausible assumption, that is that tamper-resistant devices can be built. We exhibit a security impossibility result based on this assumption. In essence, what we use in our attack is a device which holds the secret material of the dishonest prover, can run the algorithm of the prover, but from which the secret material is physically protected, so that it cannot be extracted.

Non-identifiable Device-Holders & Non-identifiable Counterfeiting. Provers have holders, i.e., a person operating them. As in most distance-bounding literature, we consider that verifiers are unable to recognise the legitimate holder of a proving device. A DB verifier generally checks simply that a device performs the protocol and it does not make any assessment on the legitimacy of the holder of such a device. For instance, in a protocol vulnerable to mafia-fraud, a person other than the legitimate holder of the proving device is present in the verifier's proximity and can make the protocol succeed. Hence, if an adversary came by the verifier with the proving device of a far-away user, this adversary would be accepted. Similarly, the appearance of the device can be counterfeited, and is not checked by the verifier.

Any DB Formal Model with MiM Security. The descriptions, i.e., (in)security results, below are irrespective of the formal DB-model in which they are cast, as long as it can capture reasonable notions of MiM security, an terrorist attacker $\mathcal{A}$ and a colluding prover P^*. Most DB models [10, 16] respect this.

3.2 Our Notion of Classical Terrorist-Fraud (CTF) Resistance

As explained in Section 2.1, terrorist-fraud resistance is a security property that is notoriously difficult to guarantee in a provable manner. A distance-bounding protocol is said to be terrorist-fraud resistant if no distant prover can help an accomplice found close to the verifier pass the protocol without giving an advantage for further access. This is usually granted by forcing the prover to leak his secret key if he gives enough information for his accomplice to succeed.

In this section, we demonstrate that, under normal assumptions made for security of DB, terrorist-fraud resistance –as it is usually defined– can never be achieved. To prove this, we exhibit a generic manner of performing a terrorist fraud, using a tamper resistant device. Hence, the so-called "classical definition", relying on the fact the prover should not leak his key, can never be reached, and should therefore not be considered anymore as a relevant security property.

TF-resistance, for instance, in the $SimTF$ notion defined in [15] informally says the following: "*If the dishonest, far-away P^* helps the attacker $\mathcal{A}$ to pass the protocol, then $\mathcal{A}$ can pass the protocol again without the help of P^*, with a non-zero probability that is not negligible.*" Yet, if a protocol is not man-in-the-middle resistant, then $\mathcal{A}$ can pass with a probability which is not negligible, even if the help of P^* is void. Hence, if a protocol has MiM insecurity, it also has $SimTF$-resistance by the triviality of the above implication. As explained in Section 2.1, multiple definitions of terrorist-fraud resistance exist in the literature and they vary widely. We therefore capture their spirit with a property we call *CTF resistance*, which we use to carry our analysis. This property is rather straightforward. It only differs from existent TF-resistance notions in that, to overcome the hurdle presented above w.r.t. $SimTF$, our notion does not consider a protocol to be terrorist-fraud resistant if it is vulnerable to a man-in-the-middle attack. Indeed, we believe that if an adversary can pass the protocol without the help of a prover, in a mafia-fraud, then any help from the prover is irrelevant, and terrorist fraud resistance becomes meaningless. In this case, to achieve terrorist-fraud resistance, the information from the prover should be a hinderance, i.e., it should remove capabilities from the adversary, which is senseless.

This definition for "classical terrorist-fraud resistance" (CTF) states that if the success probability of a MiM adversary is negligible plus there exists a pair of arbitrary PPT algorithms (P^*,$\mathcal{A}$) such that P^* is far-away and $\mathcal{A}$ can authenticate on behalf of P^*, then $\mathcal{A}$ can use his thuswise gained knowledge later to mount a MiM, which succeeds with a non-zero probability which is not negligible.

Definition 1. Classical terrorist-fraud (CTF) resistance. *Let Π be any DB protocol modelled in DB formalism, and let s be a security parameter therein in which all asymptotic measures below are given. Let V be a designated verifier, w.r.t. which authentication is taking place. Let p_{Π}^{MiM} denote the success probability of the best MiM attack against the protocol Π.*
Consider the following executions of Π.

- `Phase1`*: In a multi-party, multi-session execution of Π, a dishonest prover P_1 located far-away from V running an arbitrary PPT algorithm helps and an adversary $\mathcal{A}$, who runs an arbitrary PPT algorithm and is located close to V, pass the protocol (except for some negligible probability).*
Let $help_P$ denote the arbitrary help given to $\mathcal{A}$ in `Phase1`*.*
- `Phase2`*: Consider a multi-party, multi-session execution of Π, in which the far-away prover P_1 can now also be present but can only run the honest prover algorithm whilst being far-away from V. In this setting, the attacker $\mathcal{A}$ running some PPT algorithm, potentially using $help_P$, attempts to authenticate to V.*

The protocol Π attains "classical terrorist-fraud (CTF) resistance" *if it holds that:*
If p_{Π}^{MiM} is negligible and $\mathcal{A}$ authenticates on behalf of P_1 during `Phase1`*, then $\mathcal{A}$ can authenticate in* `Phase2` *with a non-zero probability that is not negligible.*

N.B. on Definition 1. This definition is encapsulating established TF-resistance definitions. This is also the reason why we call it "classical". By quantifying probability of passing in `Phase1` and `Phase2` and by excluding the MiM-resistance requirement from our final statement, CTF-resistance captures $simTF$ and $strSimTF$ in the model by Dürholz et. al [15] and that of Fischlin et al. [16] (this requirement, as explained, needs to be included in any $simTF$-like definition, to avoid trivial TF-resistance). In the same way, Definition 1 captures the definitions of terrorist-fraud resistance given original in the models by Boureanu et al. [9]. If the `Phase1` was to be repeated several times, then it can be lifted to capture resistance of collusion-fraud (i.e., generalised TF-resistance) by Boureanu et al. [10]. Our definition does not explicitly speak of TF-resistance by key-extraction [11], but a link between the MiM-resistance requirement within Definition and key-extraction is discussed further below. At the same time, our definition is not about reinventing the wheel; it is about just stating clearly *the* generic and commonplace TF-resistance definition (one that can be cast in any DB formal model), for which one can analyse the realisability in the context of WB/BB provers even though this is per se in line with existing TF-resistance definitions.

3.3 Impossibility of Terrorist-Fraud Resistance with in Protocol with No PUFs and WB Provers

A Generic Terrorist-Fraud Attack with WB Provers. In any DB protocol, if the prover algorithm does not use any physically uncloneable functions (PUFs) as part of its normal execution, then the prover device can be cloned. Moreover, a WB prover P^* can build a tamper-proof device that runs exactly as the proving device and exactly one time after which it self-destructs (i.e., a WB prover builds a one-time executing clone). P^* then gives this to this clone accomplice $\mathcal{A}$. The accomplice can then present the tamper-proof clone to the verifier and successfully authenticate. However, the attacker $\mathcal{A}$ cannot learn anything from the device, since it is tamper-proof and $\mathcal{A}$ cannot use it again since the clone self-destructs after one execution. This is a *generic CTF attack*. It cannot be prevented in any DB protocols which does not use PUFs on the provers' side, if provers can be corrupted in the white-box manner.

The self-destruction mentioned above equates in fact to a program wiping all data (secret and otherwise) and software on the device.

As reiterated by the above "note on Definition ", whilst we exemplify our TF-attack using CTF, the attack strategy we present above is generic and shows TF-resistance impossibility w.r.t. most commonplace TF-resistance definitions (e.g., [9]). Indeed, the attack can also be trivially generalised to apply to TF-resistances stronger than CTF-resistance. I.e., to make our CTF attack into a $(1,1,m)$-collusion-fraud attack [10], the cloned device needs to be made such that it would execute itself m times.

Offline vs. Online Help in our CTF. Classical terrorist-frauds are performed online: the prover helps his accomplice during one specific session, at a given time. In contrast, our generic CTF permits the accomplice get the disposable device offline, and to use it at a time of his choice. However, this limitation can overcome by including a secure, remote activation and deactivation mechanism on the disposable clone, in order to make it unusable except at a time chosen by the prover. By doing so, the prover could delegate his authentication rights while keeping full control on his credentials and the actions of his accomplice.

More On CTF-resistance Requiring MiM-resistance. If a protocol Π against which we mount our CTF were not MiM resistant, then it could happen that an adversary extracts the secret key from a disposable device by leveraging a flaw in the protocol. For instance, consider a dummy protocol in which at each challenges c_i, the prover responds with the corresponding bits x_i his the secret key x. In this protocol, a dishonest prover attempting to use our strategy to perform a CTF would automatically reveal his secret key to his accomplice, by the latter eavesdropping the messages emitted by the clone. Hence, the prover would not be able to delegate his authentication abilities without revealing his secret key and therefore the strategy will no longer amount to a CTF However, such MiM-vulnerable protocols are ruled out by our CTF-resistance definition, hence transferring the clone to the accomplice can safely count towards a CTF as we explained originally.

N.B. on our TF-resistance Impossibility Results. The analysis above covered only the case where the provers are WB and the DB protocols do not use PUFs on the prover's side. Section 3.4 just below treats the case of DB provers irrespective of whether the protocol uses PUFs or not. The next section, which introduces a new DB model leveraging formally the distinctions between WB and BB provers, also treats primary protocols where the authentication is based on cryptographic keys (and PRFs) rather than on evaluating PUFs. Whilst these are the great majority on the DB protocols in existence, there are two PUF-based protocols in existence, such as the one in [18]. So, in Appendix B, we take this analysis on attaining TF-resistance further into the case of WB provers in protocols using PUFs as well.

As Appendix B will detail, our study on TF-resistance impossibility therefore covers all types of DB protocols we know in the literature: i.e., our TF-resistance impossibility results herein are shown for protocols where the authentication is either based only cryptographic keys or only PUFs [18]; moreover, we make a compelling argument on the impossibility of TF-resistance even for those that would mix these two authentication methods, if they were ever to be proposed. In Appendix B, we also introduce a new notion of TF-resistance which we believe to be the suitable one for DB protocols using PUFs.

3.4 The Redundancy of Classical Terrorist-Fraud Resistance, with Black-box Provers

When the proving devices are black-box, they cannot use our strategy to mount a CTF.

However, in this setting, we show that classical terrorist fraud resistance is irrelevant. First, note that for a BlackBox (BB) prover, its holder and an adversary have the same capabilities. The last two can only interact with the device through the API of the protocol. From this, it follows that whatever help the holder gives to the adversary can be obtained by the adversary on his own, by interacting with the device in the same way the holder would. Hence, if the protocol is vulnerable to a terrorist fraud, then it is also vulnerable to a mafia fraud. In other words, in the setting of black-box provers, classical terrorist-fraud resistance is included in MiM-resistance, and therefore we do not need to consider it as a separate security property.

4 DB Model with Fine-Grained Provers

We start by introducing the constituent of a DB protocol and finish by formally defining a DB protocol in this sense.

4.1 DB Parties, Instances & Protocols

Parties, Instances & Holders. We consider two types of *parties* modelling devices: provers $\mathbb{P}$ and verifiers $\mathbb{V}$. Devices or parties have predefined, DB probabilistic polynomial-time (PPT) algorithms implemented on them. Also, devices or parties have *unique* long-term cryptographic secrets written on them; therefore, the long-term cryptographic material on it uniquely identifies a party. We generally use small letters such as x,y,... and respectively $\langle x,X\rangle$, $\langle y,Y\rangle$ to denote symmetric keys vs. the pair of secret key x and public key X, written on such devices. To this end, when we explicitly refer to one prover-party w.r.t. its unique key/key-pair, we write the prover-party $\mathbb{P}(x,...)$. When it is irrelevant that there is material other than the key on the prover-device, we simply write $\mathbb{P}(x)$. Some prover and verifier devices have pre-shared/agreed keys, which we denote as follows: the prover-party $\mathbb{P}(y)$ and the verifier-party $\mathbb{V}(y)$ to stipulate that the devices holds a long-term symmetric-key y (e.g., this may be a key-fob and a car), or the prover-party $\mathbb{P}(x)$ and the verifier-party $\mathbb{V}(X)$ to denote that a verifier-instance can retrieve the prover's public key (e.g., as it the case with contactless cards and EMV readers).

One of the verifier parties is called *designated verifier* and we will use notations stemming from $d\mathbb{V}$ to refer to it.

An *instance* Z of a party is one (possibly partial) execution of that party's PPT algorithm, together with all the inputs, outputs, internal states. If an instance is running on a prover device $\mathbb{P}$ or a verifier device $\mathbb{P}$, then we call it explicitly a *prover-instance* P or a *verifier-instance* V, respectively, or generically a *party-instance*. Each prover-instance and verifier-instance is uniquely identifiable via an *ID*. To denote a specific prover-instance or verifier-instance with the identifier id or id', we sometimes write $P_{id}(...)$ and $V_{id'}(...)$, respectively.

Prover-instances have a *holder*, e.g., the person manipulating them. In this case, we overload "instances" to mean both one execution of the algorithm of a specific-party and the holder of such a device once it is executing. When we need to make any distinction between the holder of a prover-instance and the algorithm running on the device, we write "$\mathcal{P}_{\mathsf{holder}}$" for the holder of the proving device, and "P_{alg}" for algorithm running on the device itself.

DB Positioning. We consider a fixed integer constant $\mathbb{B}$, denoting the *distance-bound*. Instances are placed (and therefore executed) at a physical location. We assume that the instance and its holder are at the same location.

The instances of the designated verifier are all run at a fixed location, and the location of this $d\mathbb{V}$ party; thus, we speak of the *the location of the designated verifier* $d\mathbb{V}$.

We define two *positions* pos of instances w.r.t. the location of the designated verifier $d\mathbb{V}$, close or far, denoting that the distance between the location of the said instance and $d\mathbb{V}$'s location is smaller-than-or-equal-to or larger than $\mathbb{B}$, respectively. To denote the explicit position of an instance Z, we sometimes use the notation $Z_{\mathsf{pos}}(...)$, where $\mathsf{pos} \in \{\mathsf{close},\mathsf{far}\}$.

Specific instances $P_{id}(...)$ or $V_{id'}(...)$ are assumed to stay at one position for the whole of their execution, even though one party can have instances at different locations in different executions.

Definition 2. ***DB Protocol.*** *A* distance-bounding protocol *Γ is a tuple $\Gamma = (\mathcal{K},P,V,\mathbb{B})$, as follows: probabilistic polynomial-time (PPT) algorithms defining the prover party $\mathbb{P}$ and the verifier party $\mathbb{V}$; pairs (x,X) are correctly sampled from $\mathcal{K}$ such that there is a* key-setup *resulting in corresponding pairs of parties $\mathbb{P}(x)$ and $\mathbb{V}(X)$ (with $X = x$ in the symmetric-key case); a distance-bound $\mathbb{B}$.*

At the end of any execution[5] of the two-party probabilistic polynomial-time (PPT) algorithms in Γ, the verifier-instance $V(X)$ involved in the execution outputs a final message Out_V. This output denotes that $V(X)$ accepts *a prover instance $P(x)$ (i.e., $\mathsf{Out}_V = 1$) or* rejects *a prover instance $P(x)$ due to failed authentication on (x,X) or timing errors w.r.t. the bound $\mathbb{B}$ (i.e., $\mathsf{Out}_V = 0$).*

Honest and Adversarial Communication Models. These run close to [10]: messages are broadcast, channels are insecure and unauthenticated, there is an explicit global-clock and the adversary can intercept, block, modify messages with no delay inflicted, yet he cannot defeat the laws of physics (i.e., make messages travel faster etc.). The details of these models are given in Appendix A.

Fine-Grained Corruption Model. The adversary (possibly embodied by the $\mathcal{P}_{\mathsf{holder}}$as well) can corrupt prover instances, in which case these will not necessary follow the $\mathbb{P}$ algorithm anymore.

There are two types of prover-corruption à la [2]: *white-box* and *black-box*. Unlike in [2] where they were just intuitively introduced, in Section 5.2 we formalise the interaction with these in terms and how this plays a role in formal DB-security definitions. Now, we explain these corruption mechanisms.

Black-box Prover-Corruption. If the $\mathcal{P}_{\mathsf{holder}}$ has no control over the algorithm run by P_{alg}, then that the prover is *black-box (BB).*

For instance, a BB prover is that where the P_{alg} is tamper-proof. This can be assumed to be the case in proximity-enhanced contactless payments [13, 19] whereby the card-holder cannot normally get to the secret-key written on the bank-card.

White-box Prover-Corruption. A *white-box (WB)* prover $\mathcal{P}_{\mathsf{holder}}$ is able to access any secret key or cryptographic material written on the P_{alg}, and also to run any algorithm of

[5] Note any execution, including an unfinished/"hanging" one, can be extended to a full execution by denoting that ones finish unsuccessfully.

$\mathcal{P}_{\mathsf{holder}}$'s choice on P_{alg}. I.e., the holder/adversary can adaptively pick nonces, or send messages at arbitrary times. In essence, the $\mathcal{P}_{\mathsf{holder}}$ is able to build his own proving device which would implement the algorithm of his choice, and build possibly multiple copies thereof.

Indeed, the majority formal DB models [10, 11, 15, 16], the $\mathcal{P}_{\mathsf{holder}}$ is assumed to have white-box access to the proving device and this was used in showing a number of DB attacks, e.g., see [8].

4.2 DB Executions

First we define DB-protocol sessions, then their concurrent execution in adversarial settings as per the above.

Definition 3. ***DB-Protocol Session.*** *Let $\Gamma = (\mathcal{K}, P, V, \mathbb{B})$ be DB protocol and two party herein be $\mathbb{P}(x)$ and $\mathbb{V}(X)$.*

A DB session *is defined by one of the following constructs:*

– *A P-$(\mathcal{A})$-V* DB session *is message-exchange correctly ordered as per Γ between a prover-instance $P_{id'}(x)$ of $\mathbb{P}(x)$ and an instance $V_{id}(X)$ of $\mathbb{V}(X)$. The adversary can use his ability to redirect, block, inject messages in order to replace P-V messages with his own, all without breaking the physical limitations (i.e., bypassing time-bounds).*
– *An $\mathcal{A}$-V* DB session *is message-exchange correctly ordered as per Γ between an adversarial instance $\mathcal{A}_{id'}$ and $V_{id}(X)$ as per the above.*
– *A P-$\mathcal{A}$* DB session *is message-exchange correctly ordered as per Γ between a prover-instance $P_{id'}(x)$ as per the above and an adversarial instance $\mathcal{A}_{id''}$.*

The messages produced by the instances involved in a session (using their keys, and random coins, etc.) uniquely define the message-exchange and identify it via a *session handle*, generally denoted by π.

In other words, similar to Bellare-Rogaway models [6] but in a somewhat non-standard ways, eithr two or three instances uniquely define a DB session: two of honest parties in a P-$(\mathcal{A})$-V session, or two of honest parties intermediated by an adversary in a P-$(\mathcal{A})$-V session, or otherwise one adversarial and one of an honest party. Note that, as we mentioned above, we considered that all adversarial instances $\mathcal{A}_{id}$ implicitly communicate with one another and therefore these instances do not underline protocol-sessions amongst themselves. As normal, we further consider that, for each party-instance, we can determine the one session in which it is involved, and the adversary can be involved in more than one session at once. Also, note that there is no authentication within the notion of session: e.g., an instance $V_{id}(X)$ of $\mathbb{V}(X)$ may "think" that it is communicating with a prover-instance $P_{id'}(x)$, when in fact $V_{id}(X)$ of $\mathbb{V}(X)$ is running a session with the adversary $\mathcal{A}$.

N.B. on Instances and IDs. We assume that party-instances identifiers are a priori assigned from a unbounded list. Then, each such identifier can be launched into a session. Also, instances are a priori placed at different positions (i.e., close or far) from the designated verifier. Lastly, each party-instances can get involved in one session at one time, and –as mentioned in Subsection 4.1– it cannot change position during that one session.

Now, we define an environment in which a DB protocol can be run in a multi-party setting (e.g., prover, verifier running on different keys) and where each party can be executed in multiple concurrent and sequential sessions, with instances placed at different positions.

Definition 4. Distance-bounding Environment. *Let s be a security parameter, poly denote a polynomial, and $\Gamma = (\mathcal{K}, P, V, \mathbb{B})$ be a DB protocol and $d\mathbb{V}$ be a designated verifier party.*

Consider tuples of the form $[ID_P, pos, mode]$*, where* ID_P *is a prover's ID,* $pos \in \{Close, Far\}$ *and* $mode \in \{white-box, black-box, honest\}$*. This denotes that any prover-instance* ID_P *in can be either corrupted (i.e., the tuple* $[ID_P,pos,mode]$ *has* $mode \in \{white-box,black-box\}$*) or honest (i.e., the tuple* $[ID_P,pos,mode]$ *has* $mode \in \{honest\}$*), and that any prover-instance* ID_P *can be placed at* $Close$ *or* Far *from* $d\mathbb{V}$*, i.e., tuple* $[ID_P,Close,\cdot]$ *vs. tuple* $[ID_P,Far,\cdot]$*.*

Consider tuples of the form $[ID_V,pos]$*, where* ID_V *is a verifier's ID,* $pos \in \{Close,Far\}$*.*

A distance-bounding (DB) environment for Γ *is a tuple* $(\Gamma,d\mathbb{V},\mathcal{VL},\mathcal{UL})$ *modelling of (possibly concurrent) sessions of* Γ*, as follows:*

- $\mathcal{VL}$ *is a list of tuples* $[ID_V,pos]$ *as per the above, with all the instances* ID_V *are honest; the list* $\mathcal{VL}$ *can be empty and it is no larger than* $poly(s)$*;*
- $\mathcal{UL}$ *is a list of tuples* $[ID_P,pos,mode]$ *as per the above; the list* $\mathcal{UL}$ *cannot be empty and it is no larger than* $poly(s)$*;*
- *there are several instances of the adversary at different positions;*
- *all instances follow the communication model above.*

Definition 5. Particular Environment. *We say a DB environment* $(\Gamma,d\mathbb{V},\mathcal{VL},\mathcal{UL})$ *is* particular *if we restrict that the positioning of the certain of the prover instances in* $\mathcal{UL}$ *is the case, and/or that the positioning of the some other the prover instances in* $\mathcal{UL}$ *is certainly not the case, and specific adversarial instances are* $\mathcal{A}_{close}$ *whilst certain others are* $\mathcal{A}_{close}$*.*

Intuitively, a DB environment is particular if it encapsulates the right setting for a distance-fraud or a mafia-fraud, etc.

Now, we introduce some terminology and notations in order to be able to single out certain specific sessions out of a DB environment.

Definition 6. DB Experiment. *A* DB experiment exp^{id} for $\Gamma = (\mathcal{K},P,V,\mathbb{B})$ *is number of DB sessions which include one specific prover-instance* id *and (possibly several) designated-verifier instances, executed in a particular DB environment for* Γ*.*

The simplest denotation of a DB experiment exp *is as follows:*

$$exp^{id} = (\breve{P}_{id}(x) \longleftrightarrow d\mathbb{V}(X)),$$

where the prover-instance $\breve{P}_{id}(x)$ *is the instances of interest. The* $\breve{}$ *on top of the said instance denotes the said "interest".*

Details of a DB experiment can be made more precise. For instance, if we write $exp = (\breve{P}_{id_3,\mathsf{close}}(x,...) \longleftrightarrow d\mathbb{V}(X))$, then this means that the prover's instance of interest, identified by id_3 is close-by to the designated verifier $d\mathbb{V}$.

To denote that exp is executing in a particular DB environment, we enlarge the notation to explicate these particular settings of the DB environment. For instance, $exp = (P_{id',\mathsf{close}}(y),...,P^{\mathsf{WB}}_{\mathsf{far}}(x),\breve{P}(x) \longleftrightarrow d\mathbb{V}(X))$, denotes that in this experiment exp: (a) we are interested in whether the instance $\breve{P}(x)$ is accepted/rejected by an instance of the designated verifier $d\mathbb{V}(X)$ (hence the $\breve{}$ on top of the said instance); (b) there are number of other instances which are noted, i.e., the instance $P_{id',\mathsf{close}}(y)$ who is honest and close-by to $d\mathbb{V}(X)$; the instance $P^{WB}_{\mathsf{far}}(x)$ who is corrupted in the WB mode and who is far from $d\mathbb{V}(X)$, etc.

N.B. on Experiments. When specifying experiments, recall that they happen in DB environments hence there are implicitly other instances executing alongside, but we do not mention them if they are not of interest to the experiment.

5 DB Security with Fine-Grained Provers

5.1 Threat Model: High-level Descriptions

5.1.1 Black-Box-driven Attacks. If we assume that a black-box prover-device (i.e., in a tamper-proof device), then the $\mathcal{P}_{\mathsf{holder}}$ does not have any extra advantage compared to an arbitrary adversary. So, when the prover algorithm is implemented on a tamper-proof device, there are actually only two threats to consider: (1) a **generalised mafia fraud**, which includes man in the middle (MiM) attacks, and (2) a **secret-extraction**.

Our generalised mafia-fraud (Definition 11). covers both mafia-fraud and distance-fraud for black-box provers: when the prover is black-box, the holder can only interact with it through an API, just like a MF adversary would. Hence, a dishonest prover can be seen as an adversary interacting with the device. It follows that a mafia fraud adversary has more resources than a distance fraud adversary, and is more general. This argument also holds for distance hijacking, and since terrorist fraud resistance is not considered (as per Section 3), we do not need to include it. Stated differently, this fraud generalises all the classical attacks for black-box provers.

Secret extraction (Definition 9) did not exist previously in the distance-bounding literature as a security property on its own. It speaks of the threat of recovering bits (one, more or all) of secret information, such as the long-term secret key.

5.1.2 White-Box-driven Attacks. White-box provers can implement any algorithm chosen by $\mathcal{A}$, and $\mathcal{A}$ obtains their secret material. Adding white-box capabilities to the adversary therefore allows him to perform more advanced attacks. In particular, in the white-box context, distance-fraud and distance-hijacking differ from mafia fraud, since the prover can maliciously pick nonces, possibly depending on his secret key, to gain some advantage. So, this needs to be considered separately, in what we call **generalised distance-fraud** (Definition 10).

A critical point w.r.t. generalised distance-fraud in WB model is that the attacker $\mathcal{A}$ should not be allowed to control any algorithm near the verifier, otherwise he could simply give it the secret material to authenticate. Hence, in our generalised distance-fraud, we explicitly rule out the presence of adversary or white-box prover instances near the verifier at the time of the attack, and only allow for honest (or black-box) provers to be located close.

Note that the generalised mafia-fraud property also applies to white-box provers, since ultimately, the prover that is targeted must be honest: otherwise, $\mathcal{A}$ would know its secret material and the attack would be trivial.

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

5.2 Threat Model: Formal Definitions

To update an DB environment, i.e., to (adaptively) determine and control the executions therein, we define a series of oracles.

Definition 7. Distance-Bounding Oracles *An DB environment* $(\varGamma,\, d\mathbb{V},\, \mathcal{VL}, \mathcal{UL})$ *can be generated and controlled/updated via the following oracles:*

– $Launch(ID_V) \longrightarrow \pi$. The oracle checks that ID_V is in the pre-defined list $\mathcal{VL}$. If so, the oracle responds by making the reader with ID_V launch a new session[6] π of the protocol $\varGamma$ and the handle to this protocol-session π is returned in identifiable form to $\mathcal{A}$.

[6] Any protocol can be modified such that it is the verifier who starts the protocol.

– $Join^{BB}(ID_P,pos)$. The oracle generates a new black-box corrupted prover $P^{BB}_{ID_P}$, with the identity ID_P, at position pos, by adding a record $[ID_P, pos, black-box]$ to the existing list $\mathcal{UL}$.
– $Join^{WB}(ID_P,pos) \longrightarrow X_{ID_P}$. The oracle generates a new white-box corrupted prover $P^{WB}_{ID_P}$, with identity ID_P, at position pos, by adding a record $[ID_P,pos,white-box]$ to $\mathcal{UL}$. It additionally responds with all the corresponding secret material X_{ID_P}.
– $Join(ID_P,pos)$. The oracle generates a new honest prover P_{ID_P} with identity ID_P, at position pos, by adding a record $[ID_P,pos,honest]$ to $\mathcal{UL}$.
– $SendToVerifier(m,ID_V,[\pi]) \longrightarrow m'$. The oracle sends m to the verifier ID_V, potentially as part of the protocol session π. If m is accepted as correct, then the corresponding answer m' as per π and Γ is returned. Otherwise, $\perp_1$ is returned.
– $SendToProver(m,ID_P,[\pi]) \longrightarrow m'$. If $ID_P \in \mathcal{UL}$ and m is accepted as correct w.r.t. the (stage of) session π and the protocol Γ, then the corresponding message/answer m' as per π and Γ is returned. Otherwise, the oracle responds with $\perp_2$.
The fact that, in the last two oracle-call, π is written between brackets denotes that we also allow a version of this oracle which does not take π as an argument, such that the attacker can send messages to ID_V which are not session-specific.
– $Result(\pi) \longrightarrow a$. When π is finished, it returns 1 or 0 if the ID_V output was 1 or 0, respectively. If π is not finished, it returns $\perp_3$.
– $Move(ID_P,pos)$. The oracle "moves" the prover ID_P from its current position to the position pos. If ID_P is involved in a session, then this session is terminated prematurely. The challenger modifies the corresponding entry in the user list $\mathcal{UL}$.
– $Identity(\pi) \longrightarrow a$. If the session π is finished in that the verifier has produced its output, it returns the identifier ID_P of the prover which was authenticated by the verifier. Otherwise, it returns $\perp$.
– $ChangeDestination(Id_1,Id_2)$. The oracle changes the destination of the next message such that if the purported destination was Id_1, then it becomes Id_2.
Note that Id_2 can be set to null, which results in the message not being sent.

N.B. on Oracles and Provers. The $Join(\cdot)$ and $Join^{BB}(\cdot)$ oracles essentially have the some outcome; as such, honest provers and provers corrupted in black-box mode equate to the same type of provers, to which we sometimes also refer as "uncorrupted".

We now give the more formal description of the security properties of interests (as per being the strongest attacks to counteract), as explained in Section 5.1: i.e., key extraction, BB-generalised mafia-fraud and WB-generalised distance-fraud.

Definition 8. DB Adversaries & DB Security Games. *Let* $(\Gamma,\ d\mathbb{V},\ \mathcal{VL},\ \mathcal{UL})$ *be a DB environment.*

An adversary $\mathcal{A}$ *against the protocol* Γ *is a probabilistic polynomial-time (PPT) algorithm following the communication and corruption model in Section 4.1, which receives as input the designated-verifier party* $d\mathbb{V}$*, the finite set of verifier identifiers* $\mathcal{UL}$ *and all the public parameters of* Γ*.*

In a DB security game $\mathcal{G}$ against a protocol Γ*, a* challenger $\mathcal{C}$ *gives access the adversary* $\mathcal{A}$ *access to a subset of the oracles above to transform* $(\Gamma,\ d\mathbb{V},\ \mathcal{VL}, \mathcal{UL})$ *into a particular environment of his choosing, modulo the oracles that* $\mathcal{A}$ *has been given access to. The challenger and the adversary may have other interactions.*

A winning condition*, stipulating when the adversary wins the game is defined.*

By using different DB security games $\mathcal{G}$, we will now define different security properties for DB, which should be considered in the different corruption model, i.e., white-box vs. black-box.

5.3 Black-box-driven Security Properties

While a mafia-fraud adversary may be able to recover at least parts of the secret-key of the prover, we believe that key-extraction/secret-extraction resistance should also be considered as a DB security-property on its own. That is, if the P_{alg} is built to be tamper-proof, then not even part of the secret material should possibly leak from the protocol. Hence, if a prover is capable of extracting even just one bit of a secret by using the protocol, then the protocol violates the "tamper-proofness" of the device, even if this bit is not enough for a mafia fraud adversary to pass the protocol. We formalise this in Definition 9 below.

Definition 9. Secret-extraction. *Let $\mathcal{G}$ be a DB security game against a protocol Γ in which the challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ access to all the oracles. As per any DB game, the adversary $\mathcal{A}$ sets up a particular environment using these oracles and a specific experiment exp^{ID_P}.*

The game $\mathcal{G}$ is a secret-extraction game*, if the play is as follows:*

– *$\mathcal{A}$ choses a session π, inside an experiment exp^{ID_P};*
– *If ID_P corresponds to a black-box prover, then $\mathcal{C}$ defines a set S^{π} containing the long-term secrets and the ephemeral secrets linked to π and belonging to ID_P. Otherwise, the experiment is aborted;*
– *$\mathcal{A}$ then outputs a value x*

The winning condition *is that $x \in S^{\pi}$. The* advantage of the adversary *$\mathcal{A}$ in winning $\mathcal{G}$ with probability α in the secret-extraction game is defined as $|\alpha - \frac{1}{2^{|x|}}|$.*

The protocol Γ is secure against secret-extraction *if the advantage of an adversary $\mathcal{A}$ in the secret-extraction game is negligible in the security parameter defining Γ.*

5.4 White-box-driven Security Properties

We now define the security property that is only relevant for white-box provers: *generalised distance fraud*. For distance-frauds, having white-box access to a prover is a great advantage. For instance, in some protocols, it allows the dishonest prover to adaptively choose values in the execution to force responses be independent of the challenge (i.e., $r_i^0 = r_i^1$), for all rounds, and thus successfully perform a distance-fraud. As such, we formalise WB distance-frauds, in a generalised setting, below in Definition 10.

Definition 10. Generalised Distance-fraud.

Let $\mathcal{G}$ be a DB security game against a protocol Γ in which the challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ access to all the oracles.

As per any DB game, the adversary $\mathcal{A}$ sets up a particular environment using these oracles and a specific experiment exp^{ID_P}.

The game $\mathcal{G}$ is a generalised distance-fraud game*, if the play is in two phases, as follows:*

– *Learning phase*
  • *$\mathcal{A}$ outputs a prover identifier ID;*
– *Attack phase*
  • *$\mathcal{A}$ loses access to the oracle $Move(\cdot,\cdot)$*
  • *$\mathcal{C}$ checks the location of ID in $\mathcal{UL}$ (i.e. pos, in $[ID, pos, mode]$ in $\mathcal{UL}$): if it is close (i.e., pos=close), the game $\mathcal{G}$ is aborted.*
  • *$\mathcal{C}$ checks the location of all $\mathcal{A}$ and P_{WB} instances: if any is close, then game $\mathcal{G}$ is aborted.*

The winning condition *on an (unaborted) generalised distance-fraud game $\mathcal{G}$ is as follows: in the attack phase, there exists a session π such that, during the whole session π, no adversarially controlled algorithm (either $\mathcal{A}$ or P_{WB}) was close to the verifier, $Identity(\pi)$ is registered as $white\!-\!box$ in $\mathcal{UL}$, and $Result(\pi)=1$.*

The advantage in the generalised distance fraud game of an adversary $\mathcal{A}$*, who succeeds with probability α, is defined as $|\alpha-\frac{1}{2}|$.*

The protocol Γ is secure against generalised distance-fraud *if the advantage of an adversary $\mathcal{A}$ in the generalised distance-fraud game is negligible in the security parameter defining Γ.*

The generalised distance-fraud game $\mathcal{G}$ can be graphically represented via the following experiments:

- the learning phase: $(P(y),\mathcal{A},P^{WB}\longleftrightarrow d\mathbb{V}(X))$,
- the attack phase: exp^{ID}=$(P(y),\mathcal{A}_{far},\breve{P}^{WB}_{ID,far}(x,\ldots)\longleftrightarrow d\mathbb{V}(X))$.

For this generalised distance-fraud property in the WB model, as per Definition 10 above, no adversarially-controlled entity are allowed near to the prover. Yet, honest provers, which could also be referred to as black-box provers, are allowed near the verifier.

Hence, Definition 10 also covers the threat of distance-hijacking (DH) [14]. If one wishes to distinguish clearly whether an attack found against Definition 10 is a type of distance-fraud (albeit cast in our multiparty/multisession setting) or a DH attack, then one needs to look at the type of verifier-accepted session that π (as per $Result(\pi)=1$ in Definition 10) is. If this is a P-$\mathcal{A}$-V session than we are faced with a DH attack, otherwise if it is $\mathcal{A}$-V it is a distance-fraud attack.

5.5 "AnyBox" Security Properties

We now define the property that is relevant in WB and BB models: generalised mafia-fraud, in which an adversary tries to make a distant, uncorrupted prover be accepted. For generalised mafia-fraud, we will formalise an *attack phase* whereby a far-away adversary, possibly holding several provers, tries to make the verifier accept the authentication of one of them. He can leverage the help of another adversarial instance located close to the verifier, along with other honest provers. This follows in Definition 11 below.

Definition 11. Generalised Mafia-fraud

Let $\mathcal{G}$ be a DB security game against a protocol Γ in which the challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ access to all the oracles.

As per any DB game, the adversary $\mathcal{A}$ sets up a particular environment using these oracles and a specific experiment exp^{ID_P}.

The game $\mathcal{G}$ is a generalised mafia-fraud game*, if the play is in two phases, as follows:*

- *Learning phase*
  - *$\mathcal{A}$ outputs a prover identifier ID;*
- *Attack phase*
  - *$\mathcal{A}$ loses access to the oracle $Move(\cdot,\cdot)$,*
  - *$\mathcal{C}$ checks the characteristics of ID in $\mathcal{UL}$: if it is close (i.e., pos=*close*), or white-box, then the game $\mathcal{G}$ is aborted.*

The winning condition *on an (unaborted) generalised mafia-fraud game $\mathcal{G}$ is as follows: in the attack phase, there exists at least one session π such that $Result(\pi)=1$ and $Identity(\pi)=ID$.*

The advantage in the generalised mafia-fraud game of an adversary $\mathcal{A}$*, who succeeds with probability α, is defined as $|\alpha-\frac{1}{2}|$.*

The protocol Γ is secure against generalised mafia-fraud *if the advantage of an adversary $\mathcal{A}$ in the generalised mafia-fraud game is negligible in the security parameter defining Γ.*

The generalised mafia-fraud game $\mathcal{G}$ can be graphically represented the following experiments:

- the learning phase: $(P(y),\mathcal{A},P^{WB}(\cdot)\longleftrightarrow d\mathbb{V}(X))$,
- for the attack phase: exp^{ID}=$(P(y),\mathcal{A},\breve{P}^{honest}_{ID,far}(x,\ldots)\longleftrightarrow d\mathbb{V}(X))$.

In this game's formalisation in Definition 11, like in [10], prior to the execution of the attack phase whereby the fraudulent authentication is attempted, the adversary is given access to a *learning phase*. During this phase, he can place the target-prover close to the verifier. We mean that this learning phase is typically used to recover some secret material, by modifying messages during the challenge response part of the protocol, which is possible if both the prover and the adversary are close to the verifier.

Note that we can easily and naturally enhance this model with oracles and definitions that treat privacy and anonymity notions in DB; this is left for future-work.

In Section 6, we show how to use our new model to do cryptographic proofs of a DB protocol, namely DB3 from [11].

6 Security Proofs in the New DB Model: A Case-study in the DB3 Protocol

In this section, we revisit the security proofs of the DB3 protocol [11]. Our proofs are designed as sequences of games, as introduced in [20]. I.e., from the initial security game Γ_0, which is the one defined in the definition of the security property, we move step by step to a new game. The transitions are such that the adversary can only notice them with a negligible probability. The final game is one in which the proof is easier to carry out, as most of the complexity of the protocol was removed.

In what follows, $PR[\Gamma_i]$ denotes the success probability of $\mathcal{A}$ in the game Γ_i.

6.1 Protocol description

The DB3 (q=2) protocol is depicted in Figure 1 and it self-explained as per this figure. A more complete description can be found in [11].

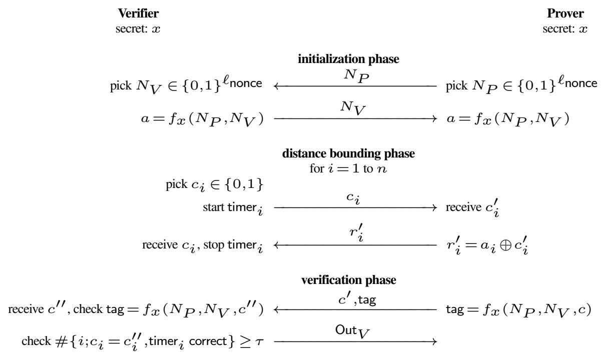

Fig. 1. The DB3 Distance-Bounding Protocol with $q = 2$.

Pseudorandom Functions & The Pseudorandom Function Assumption. A *pseudorandom function (PRF)* is a family of (polynomially computable) functions: a set $(f_k)_{k \in \mathcal{K}}$ of functions of arbitrary-length input and arbitrary-length output indexed on a set of keys $\mathcal{K}$. On this family, a computational assumption is taken, which is denoted as the *pseudorandom function (PRF) assumption*, i.e.,: for an instance sampled uniformly from the family, there exists no polynomial algorithm that distinguishes this instance from a real random function based on a black-box interaction with an oracle simulating them. More formally, the *PRF assumption holds* if the adversary has a probability negligibly close to $\frac{1}{2}$ to win in the following *PRF-security game*. Let k be a key sampled uniformly at random from the key-domain $\mathcal{K}$ and let f_k be the associated PRF-instance from the PRF family $(f_k)_{k \in \mathcal{K}}$. A challenger picks at random a bit b, and depending on it, gives the adversary oracle-answers $O(\cdot)$, such that $O(\cdot) := f_k(\cdot)$ if $b = 0$, and $O(\cdot) := F(\cdot)$ otherwise, where F is a random function. The adversary is given a polynomial number of queries to the oracle O, and wins if he guesses b. His advantage in this game is $|\frac{1}{2} - Pr[\mathcal{A}\ wins]|$, which is the absolute value of the difference the probability of guessing b at random, and the success probability of $\mathcal{A}$.

6.2 Secret-Extraction Security for DB3

Consider the security-definition of secret-extraction in Def. 9, Section 5.2. Recall that this notion defines security against the threat of recovering bits (one, more or all) of secret information, such as the long-term secret key. In DB3, the set of secret information S^{π} only contains x.

Theorem 1. *If the PRF assumption holds, then DB3 has secret-extraction security.*

Proof. Let s be the security parameter of DB3, and $negl$ be a negligible function.

From the initial secret extraction game Γ_0 (see Def 9), we build Γ_1:

Γ_1: This game is the initial game Γ_0, where no N_P value is indeed used more than once, over all honest or black-box prover instances.

Let q be the number of N_P values issued by oracles involving honest or black box prover-instance, during the experiment encapsulating this game. The probability that one N_P repeats is upper bounded by $\frac{q^2}{2^n}$, which is negligible. Hence, $Pr[\Gamma_1] - Pr[\Gamma_0] \leq \frac{q^2}{2^n}$.

Γ_2: This game is the game Γ_1, where:
– each PRF-instance $f_{x \in \mathcal{X}_{BB}}$ is replaced by a random function, with $\mathcal{X}_{BB}$ is the set of all secret keys used by black-box prover instances,
– the PRF-instances $f_{y \in \mathcal{X}_{WB}}$ are left as they are, with $\mathcal{Y}_{WB}$ being the set of all secret keys used by white-box prover instances.

Intuitively, Γ_2 is a now a game in which every value sent by black-box provers is independent of their secret key, so that $\mathcal{A}$ has no better choice than guessing it.

Note that this transition is possible (i.e., "programmable PRF" issues [8] that may arise in DB and prevent this step do not apply here): the adversary does not know the key of the concerned PRF, and its output is never combined with any other key-dependent value.

This transition is actually a hybrid argument: it implicitly assumes as many game hops as there are honest provers, $i.e\ \Gamma_{2_{0 \dots 2_{qhp}}}$, where qhp is the number of honest provers, Γ_{2_0} is the same as Γ_1, and $\Gamma 2_{qhp}$ is the same as Γ_2. In each of these games, one more PRF is replaced by a random function.

We now prove that $|Pr[\Gamma_{2_i}] - Pr[\Gamma_{2_{i-1}}]| \leq negl(s)$, for i from 1 to qhp. To prove it, we use $\mathcal{A}$ to build a ppt. adversary $\mathcal{A}'$, such that the success probability of $\mathcal{A}'$ in the the PRF-security game is proportional to $|Pr[\Gamma_{2_i}]-Pr[\Gamma_{2_{i-1}}]|$. Hence, any non negligible difference between two games would contradict the assumption that the PRF is secure.

To build $\mathcal{A}'$, we use a slightly modified environment for the secret-extraction experiment. We store a counter i, which is the same as in the notation Γ_{2_i}. The oracles $Join$ and $Join^{BB}$ are modified as follows:

– Instead of generating a secret key x, the $Join$ or $Join^{BB}$ oracles start a new PRF experiment when they are called for the i^{th} prover, and receive the corresponding $O(\cdot)$ oracle, which will be used by the corresponding prover instead of f_x. For the j^{th} call, if $j<i$, the prover is the same as in the $j-1^{th}$ game ($i.e$, the PRF experiment created in Γ_{2_j} is used for the j^{th} prover. If $j>i$, then the oracle behaves as in Γ_1;

The $Join^{WB}$, on the other hand, is not modified and still uses a PRF.

Hence, only one thing changes between two consecutive games: one more honest (or black-box) prover uses the oracle O provided by a PRF challenger instead of his PRF.

Note that, from the point of view of the adversary, black-box and honest provers are indistinguishable, as the single difference between them is the record in $\mathcal{UL}$, which is private: hence, the counter can be the same for both $Join$ and $Join^{BB}$, $i.e$ we do not need to distinguish the case of black-box and honest provers in two separate games.

Finally, if $\mathcal{A}$ wins in Γ_{2_i}, then the distinguisher $\mathcal{A}'$ in the PRF-security game replies 0 (i.e., the PRF challenger chose a PRF). If $\mathcal{A}$ loses, $\mathcal{A}'$ replies 1 (i.e., the PRF challenger chose a random function).

This distinguisher $\mathcal{A}'$ wins if $b=0$ and $\mathcal{A}$ wins, or if $b=1$ and $\mathcal{A}$ loses. Note that if $b=0$, then $\mathcal{A}$ is in the game $\Gamma_{2_{i-1}}$ (with a PRF), and otherwise, $\mathcal{A}$ is in Γ_{2_i} (with a random function). So, the success probability of $\mathcal{A}'$ in the PRF experiment is $Pr[b=0]\cdot Pr[\Gamma_{2_{i-1}}]+Pr[b=1]\cdot(1-Pr[\Gamma_{2_i}])=\frac{1}{2}\cdot(1+Pr[\Gamma_{2_{i-1}}]-Pr[\Gamma_{2_i}])$. Hence, any non negligible advantage of $\mathcal{A}$ in winning in $\Gamma_{2_{i-1}}$ over Γ_{2_i} can be directly leveraged as an advantage in the PRF experiment, which concludes the proof.

Hence, $|Pr[\Gamma_1]-Pr[\Gamma_2]|$ is negligible.

We now prove that the success probability of $\mathcal{A}$ in Γ_2 is negligible. In Γ_2, every variable is independent of x. Hence, there is no better strategy for $\mathcal{A}$ for outputting any subset y of x than guessing, with probability $\frac{1}{2^{|y|}}$.

From this, we obtain that the success probability of a secret extraction adversary against DB3 is $\frac{1}{2^{|y|}}+negl(s)$, where the negligible factor is a polynomial function of the advantage of $\mathcal{A}$ against the PRF.

□

6.3 Generalised Mafia-Fraud Security for DB3

Consider the security-definition of generalised mafia-fraud in Def. 11, Section 5.2. Recall that this notion defines security against a generalised form of mafia fraud, in which the adversary aims at making the verifier accept a honest, far away prover.

Theorem 2. *If the PRF assumption holds, then DB3 has generalised mafia-fraud security.*

Proof. Let s be the security parameter of DB3, and $negl(s)$ be a negligible function of s.

For this proof, we first eliminate the possibility of each nonce being used more than once, and then replace each PRF f_x instance for which $\mathcal{A}$ does not know the key with a random function. From the initial game Γ_0 (the generalised mafia-fraud in Def. 11), we obtain a final game Γ_3, in which all the a vectors are therefore unpredictable by the adversary.

Γ_1: This game is the initial game Γ_0, where each N_V is indeed never used more than once.

Let q be the number of N_V values issued by oracles involving verifier-instance (i.e., $SendToVerifier$ and $Launch$), during the experiment encapsulating this game. The probability that one N_V repeats is upper bounded by $\frac{q^2}{2^n}$, which is negligible. Hence, $Pr[\Gamma_1] - Pr[\Gamma_0] \leq \frac{q^2}{2^n}$.

Γ_2: This game is the game Γ_1, where N_P is never used more than once by honest /black-box prover-instances.

Let q be the number of N_P values issued by the black-box/honest provers through the oracles during the experiment encapsulating this game. The probability that one N_P repeats is upper bounded by $\frac{q^2}{2^n}$, which is negligible. Hence, $Pr[\Gamma_2] - Pr[\Gamma_1] \leq \frac{q^2}{2^n}$.

Γ_3: This game is the game Γ_2, in which the PRF-instances are replaced by a random function for each uncorrupted prover-instances, but remains a PRF for the corrupted ones.

This transition is the same the one in the secret-extraction proof (for Γ_2 therein), hence we do not reexplain it, and obtain, $|Pr[\Gamma_3] - Pr[\Gamma_2]| \leq negl(s)$

We are left to prove that the probability for $\mathcal{A}$ to win in Γ_3 is negligible.

In the first step of the generalised mafia-fraud game, $\mathcal{A}$ picks a prover P. Since P is uncorrupted, he uses a random function instead of a PRF instance, and since N_P and N_V do not repeat, the vector a is unpredictable for $\mathcal{A}$ except by guessing at random. In the second step, $\mathcal{A}$ must authenticate on behalf of P. To pass the distance bounding phase with P being far, at each round, he can either pre-ask ($i.e.$ send a random challenge to P in advance, in order to obtain the response in time to send it to V), post ask ($i.e.$ forward the challenge from V to P and send a random response before P responds), or not ask, $i.e.$, wait for the challenge and send a random response. In the three cases, he needs to guess one random bit, either the challenge or the response, which succeeds with probability $\frac{1}{2}$. In the two strategies where $\mathcal{A}$ guesses the response, any wrong guess makes the authentication fail, so his probability is $\frac{1}{2^n}$. If he guesses the challenge, then a wrong guess changes the input used to compute tag. Since f is a random function, tag is modified in a unpredictable way, then $\mathcal{A}$ only wins if $f(N_P, N_V, C) = f(N_P, N_V, C')$, for $C' \neq C$, which occurs with the negligible probability $\frac{1}{2^n}$.

So, given this last statement plus the game-hops above, the probability of $\mathcal{A}$ winning in this MF security game is $\frac{1}{2^n} + negl(s)$.

□

6.4 Generalised Distance-Fraud Security for DB3

Consider the security-definition of generalised distance fraud in Def. 10, Section 5.2. Recall that this notion defines security against a generalised form of distance fraud, in which a far away, white-box corrupted prover is accepted by the verifier, with the condition that no adversarially controlled algorithm is near the verifier.

Theorem 3. *If the PRF assumption holds, then DB3 has generalised distance-fraud security.*

Proof. Let s be the security parameter of DB3, and $negl$ be a negligible function.

For this proof, initial game Γ_0 (the generalised distance-fraud in Def. 10), we obtain a final game. We first replace the PRF instances used by every non-adversarially controlled entity by random functions, and then prove that $\mathcal{A}$ is left with a negligible winning probability.

Γ_1: This game is the game Γ_0, in which the PRF-instances are replaced by a random function for each uncorrupted prover-instances, but remains a PRF for the corrupted ones.

This transition is the same as the one in the secret-extraction proof (for Γ_2 therein), hence we do not reexplain it, and obtain, $|Pr[\Gamma_1]-Pr[\Gamma_0]| \leq negl(s)$

We now prove that the success probability of $\mathcal{A}$ is negligible in Γ_1. The prover P^* picked by $\mathcal{A}$ during the learning phase is located far from $d\mathbb{V}$, but there are other provers P located near the designated verifier, but running with different keys.

For P^* to pass the distance bounding phase, he needs to send the correct a_i at each round (since there is no error tolerance). He can either send his own response, or trigger the response from a nearby honest prover. In both cases, the response is correct with probability $\frac{1}{2}$: In the first case, because in order for the response to arrive on time, $\mathcal{A}$ must send it before receiving the challenge c_i, and a wrong guess yields a wrong response $(a_i \oplus c_i)$. In the second case, the response is correct iff a_i (as picked by $\mathcal{A}$) is equal to a'_i (chosen by the honest prover). However, a' is the output of a random function, so $Pr[a_i = a'_i] = \frac{1}{2}$. The probability to succeed in all rounds is therefore $\frac{1}{2^n}$, which is negligible.

So, the probability of $\mathcal{A}$ winning in this DF security game is $\frac{1}{2^n}$ + $negl(s)$.

$\square$

N.B. on Proofs. Our proofs are done for a noiseless environment, where the noise resistance factor τ is set to 1: all responses and times must be correct. However, we can adapt them to work in noisy environments.

7 Conclusion

Firstly, we have showed that – under reasonable assumptions – one cannot achieve terrorist-fraud resistance in the standard senses of this notion. Actually, Appendix B discusses this in more detailed manner, considering different ways of using of PUFs in the protocols, and proposes a new TF-resistance that can be attained by some PUF-based protocols.

Secondly, by combining elements from existing DB frameworks [2, 10, 15], we yielded a new model that captures DB-security via a small number of clear security definitions, where the prover-corruption models is fined-grained (i.e., an adversary can play at once with white-box and black-box provers). We also showed how to carry out security proofs in this new model. This model has a number of advantages: (1) our definitions can be "picked and chosen" by an application-designer who wishes to integrate DB, upon the threats that their proving devices can be exposed to; (2) we explicit model the security-notion of key-leakage, which has been so far shortchanged in distance-bounding; (3) we believe that our oracle-style adversary model, our notion of DB environment and our security-definitions make our model much suited to mechanising our security proofs in a tool like Easycrypt [5]; (4) our model is easily augmented to treat fine-grained privacy and anonymity properties in DB. The last two points constitute our future work on this line. Another interesting future work-line would be to find a generic terrorist-fraud attack that does not rely on tamper-proof devices.

References

1. 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.

2. 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.
3. 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.
4. G. Avoine, C. Lauradoux, and B. Martin. How Secret-sharing can Defeat Terrorist Fraud. In *Proceedings of the 4th ACM Conference on Wireless Network Security – WiSec'11*, Hamburg, Germany, June 2011. ACM, ACM Press.
5. G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P.-Y. Strub. Easycrypt: A tutorial. In A. Aldini, J. Lopez, and F. Martinelli, editors, *Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures*, pages 146–166. Springer International Publishing, Cham, 2014.
6. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proc. of Advances in Cryptology – CRYPTO '93*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, 1994.
7. I. Boureanu, D. Gerault, P. Lafourcade, and C. Onete. Breaking and fixing the HB+DB protocol. In *WiSec 2017-Conference on Security and Privacy in Wireless and Mobile Networks*, pages 241–246, 2017.
8. 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.
9. 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.
10. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical and Provably Secure Distance-Bounding. *Journal of Computer Security*, 23(2):229–257, 2015.
11. I. Boureanu and S. Vaudenay. Optimal proximity proofs. In *Proc. of Inscrypt*, pages 170–190. Springer, 2015.
12. 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.
13. T. Chothia, F. D. Garcia, J. de Ruiter, J. van den Breekel, and M. Thompson. Relay cost bounding for contactless EMV payments. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pages 189–206, 2015.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. MasterCard. Contactless paypass reader specifications v3.1,. not publically available, 2017.
20. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. `http://eprint.iacr.org/2004/332`, 2004.

A Honest and Adversarial Communication Models

We herein define our DB communication model.

DB Communication Model. Exchanges of messages occur between instances, from a location to another location. Each communication takes time, proportional to the distance between said instances. We assume a *global clock*, which can measure the time-of-flight of all exchanges.

Communication is done via broadcast. An instance $S_{id_1}(...)$ who wants to send a message m to an instance $R_{id_2}(...)$ just broadcasts the message, setting R's ID as the "purported destination"; this "purported destination" is a virtual pointer and it is not protected cryptographically. However, subparts of m can well include elements that only $R_{id_2}(...)$ can decipher. The message m reaches other instances closer to $S_{id_1}(...)$ than $R_{id_2}(...)$ first and these instances can read m.

Communication channels are un-authenticated, i.e., there is no element in the channel that identifiers the sender of a message. Also, there is no element in the algorithms of the parties which can distinguish the holder $\mathcal{P}_{\mathsf{holder}}$ of a prover devices from the instance of that proving device P_{alg} run at the location of the holder.

Communication channels are insecure, i.e., messages can be read, intercepted, blocked or modified whilst in transit.

Adversarial Communications. We consider an adversarial party $\mathcal{A}$, described via an PPT algorithm. It can also have several instances. We make no distinction in the font/letter used to denote the adversary's party $\mathcal{A}$ and its instances.

Adversarial instances can run arbitrary PPT algorithms, at both locations: close, far. When it is not clear from the context, we specify $\mathcal{A}_{close}$ or $\mathcal{A}_{far}$. The attacker $\mathcal{A}$ (and all its instances) can modify the "purported destination" of messages, or block them, irrespective of where they come from (i.e., irrespective if the message comes from an instance close to the designated verifier or from one far-way from it, etc.). Adversarial modification/blocking per se is instantaneous, i.e., adversarial modification/blocking does not introduce any extra delay. However, making a modified message travel (e.g., from an instance $\mathcal{A}_{close}$ to an instance $\mathcal{A}_{far}$) is subject to distance/locations. In essence, all communications from any instance $\mathcal{A}_{close}$ to any instance $\mathcal{A}_{far}$ and vice-versa is subject to time-measurement, i.e., the adversary cannot defeat the laws of optics/mechanics and make his signal travel faster between his instances or make them appear be at a lesser distance from one another.

To mount a MiM/mafia-fraud attack, at least two instances of $\mathcal{A}$ are necessary: one instance $\mathcal{A}_{far}$ to impersonate the verifier to the far-away prover and one instance $\mathcal{A}_{close}$ to impersonate the prover in the proximity of the designated verifier. Whilst the communication between an instance $\mathcal{A}_{close}$ to an instance $\mathcal{A}_{far}$ is subject to time/distance (as aforementioned), in we do not explicitly quantify the communication between these two instances: in the case of MiM/MF, we speak of just one adversarial entity embodying these two instances and their communication restrictions (see Section 5.2).

B (Im)possible TF Resistances in Protocols using PUFs

This is cast under the same assumptions on DB models as in Section 3.

Note that our generic attack in Section 3 cannot be mounted onto protocols using PUFs on the provers' side (since PUFs make cloning is impossible). As such, we believe that another type of TF-resistance should be introduced for protocols using PUFs. It is, on the other hand, not achievable without a PUF.

B.1 `Real-Life Terrorist-Fraud`-resistance: A Possible Way to TF-resistance for PUF-based Protocols.

In some protocols, the prover can simply not provide any helpful information for an accomplice to pass the protocol. For instance, consider a protocol with only one time critical round, in which the response to a challenge C is $PUF(C)$, and C is n bit long, with n depending on the security parameter. Since we consider polynomially bounded adversaries, and the number of possible challenge response pairs is exponential in the security parameter, the prover can simply not give his accomplice enough data to be able to respond with a good probability. Moreover, if the accomplice forwarded C to the far away prover, then he would not receive the response early enough to respond within the time-bound. Additionally, if the same challenge is never used more than once, then the protocol is generalised mafia-fraud resistant. It is also generalised distance-fraud resistant, and trivially secret-extraction resistant, since there are no secret keys.

Hence, in such a protocol, the prover has no physical way of giving his accomplice any non-negligible advantage to succeed. This is in fact form of terrorist-fraud resistance (by triviality). We define it as RTF. A protocol is *`real-life terrorist-fraud`-resistant* if whatever help is given by a malicious prover to an accomplice does not allow the accomplice to perform significantly better than if the prover was honest. In this attack model, the prover is not forbidden to give his secret key as in a regular terrorist-fraud. More formally, `real-life terrorist-fraud`-resistance is as follows.

Definition 12. `Real-Life Terrorist-Fraud`. *Let Π be any DB protocol modelled in DB formalism, and let s be a security parameter therein in which all asymptotic measures below are given. Let V be a designated verifier, w.r.t. which authentication is taking place. Let p_{Π}^{MiM} denote the success probability of the best MiM attack against the protocol Π.*

A `Real-Life Terrorist-Fraud` adversary *is a pair of any PPT algorithms P and $\mathcal{A}$, such that P is located far-away from the designated verifier, and $\mathcal{A}$ is at an arbitrary position.*

Let $p_{\Pi}^{\mathcal{A}}(help_P)$ denote the probability of an accomplice $\mathcal{A}$ to pass the protocol with the help of a malicious prover P, where $(P,\mathcal{A})$ is a `Real-Life Terrorist-Fraud` adversary.

A protocol MiM-resistant protocol Π is `Real-Life Terrorist-Fraud` resistant *if, for all PPT algorithms P and $\mathcal{A}$, $|p_{\Pi}^{\mathcal{A}}(help_P) - p_{\Pi}^{MiM}| \leq negl(s)$, where $negl(s)$ denotes the set of negligible functions over the security parameter s.*

Note that protocols vulnerable to MiM attacks are excluded from this definition and cannot be said to be `Real-Life Terrorist-Fraud` resistant.

B.2 Impossibility of Classical Terrorist Fraud Resistance with PUFs.

Now, we make an argument that no DB protocol "purely" based on PUF can be CTF resistant.

Definition 13. ***DB Protocol Purely PUF-based.*** *A DB protocol is* purely PUF-based *if the authentication mechanism of a prover P is based only on a specific PUF held by the prover P and it is specifically not based on any cryptographic keys (shared or private-public pairs).*

Lemma 1: No purely PUF-based DB protocol can be CTF resistant.

Proof. Let Π be a purely PUF-based DB protocol. Assume by contradiction that Π is CTF-resistant.

Since Π uses a PUF, P^* cannot build a complete disposable clone of his device: the best he can do is build an incomplete clone, that does not embed the PUF.

By our R.A. assumption, Π is CTF resistant, so the accomplice $\mathcal{A}$ gains a non negligible advantage for passing the protocol after being successfully helped by the malicious prover P^*.

This help, whatever it is, is denoted h. By definition of a PUF, $\mathcal{A}$ cannot compute the output of the PUF of P^* on a new challenge. Hence, there exists a PPT algorithm $\mathcal{A}'(h)$ that can win the MF game without being able to compute the PUF.

Hence, P^* can build a disposable tamper proof device based on $\mathcal{A}'(h)$ and give it to $\mathcal{A}$, so that $\mathcal{A}$ presents it to the verifier in order to authenticate without learning anything more than what appears from a protocol session.

Hence, the protocol is not CTF resistant, so our assumption is false and the statement is proven.

□

Discussions on our TF-resistance Impossibility Results: PUFs vs cryptographic Keys. Our TF-resistance impossibility results were shown for protocols where the authentication is either based only cryptographic keys (Section 3) or only PUFs (as per the Lemma 1 above). All protocols we know in the literature fall into this. (For the ones based only on PUFs, see e.g. [18]).

But, we are left to consider the case of "hybrid" protocols, which use both a PUF and keys for the authentication mechanisms. In this case, idea of a generic CTF attack on such protocols would be to build a disposable clone in which all calls to the PUF are replaced by queries to P^* over an encrypted channel[7]: if these calls occur during the slow phases, then there is no timing issue and the change goes unnoticed for the verifier. If the calls are made during the online phase, then two possibilities exist. Either the size/entropy of the challenge make it possible to query all possible responses before the round, which the clone does, or the challenges are too long/unpredictable, and then, the protocol is RTF resistant, since P^* can simply not help $\mathcal{A}$ pass.

Conclusions on our TF-resistance (Im)possibility Results. On the one hand, we show TF-resistance impossibility results for protocols where the authentication is based only cryptographic keys (Section 3), based only on PUFs (Lemma 1), and above we give a strong intuition that classical terrorist-fraud resistance notion cannot be achieved, even when the authentication resides both on PUFs and on cryptographic keys.

On the other hand, a stronger notion, RTF resistance, in which the prover can simply not help the accomplice, can be attained if one uses a DB protocol purely PUF-based.

[7] We make the extra assumption that encryption only introduces a negligible delay, for instance via a one time pad.