A Terrorist-fraud Resistant and Extractor-free Anonymous Distance-bounding Protocol

Gildas Avoine, Xavier Bultel, Sébastien Gambs, David Gerault, Pascal Lafourcade, Cristina Onete, Jean-Marc Robert

A Terrorist-fraud Resistant and Extractor-free Anonymous Distance-bounding Protocol*

Gildas Avoine[1], Xavier Bultel[2], Sébastien Gambs[3], David Gérault[2], Pascal Lafourcade[2], Cristina Onete[1], and Jean-Marc Robert[4]

[1] *INSA/IRISA Rennes*
[2] *LIMOS, Université Clermont d'Auvergne*
[3] *UQAM Montréal*
[4] *ETS Montréal*

April 4, 2017

Abstract

Distance-bounding protocols have been introduced to thwart relay attacks against contactless authentication protocols. In this context, *verifiers* have to authenticate the credentials of untrusted *provers*. Unfortunately, these protocols are themselves subject to complex threats such as *terrorist-fraud* attacks, in which a malicious prover helps an accomplice to authenticate. Provably guaranteeing the resistance of distance-bounding protocols to these attacks is a complex task. The classical countermeasures usually assume that rational provers want to protect their long-term authentication credentials, even with respect to their accomplices. Thus, terrorist-fraud resistant protocols generally rely on artificial *extraction mechanisms*, ensuring that an accomplice can retrieve the credential of his partnering prover.

In this paper, we propose a novel approach to obtain *provable* terrorist-fraud resistant protocols without assuming that provers have any long-term secret key. Instead, the attacker simply has to replay the information that he has received from his accomplice. Based on this, we present a generic construction for provably secure distance-bounding protocols, and give three instances: (1) an efficient symmetric-key protocol, (2) a public-key protocol protecting the identities of the provers against external eavesdroppers, and finally (3) a fully anonymous protocol protecting the identities of the provers even against malicious verifiers trying to profile them.

1 Introduction

In recent years, contactless communications have become ubiquitous. They are used in access control cards, electronic passports, payment systems, and nu-

*This research was conducted with the support of the FEDER program of 2014-2020, the region council of Auvergne, and the Digital Trust Chair of the University of Auvergne. This work was also funded by NSERC Discovery Grant.

merous other applications, which often require some form of *authentication*. In authentication protocols, the device to authenticate is typically an RFID tag, a contactless card or more and more frequently an NFC-enabled smartphone, acting as a *prover*. Before accessing some resources, this device has to authenticate to a reader, which plays the role of a *verifier*.

An important concern for contactless communications are *relay* attacks, in which an adversary forwards the communications between a prover and a verifier to authenticate [4]. These attacks cannot be prevented only by cryptographic tools and thus mechanisms ensuring the physical proximity between a verifier and a prover must be used. Distance-bounding (DB) protocols [10] have been proposed to allow the verifier to estimate an upper bound on his distance to the prover by measuring the time-of-flight of short challenge-response messages exchanged during *time-critical* phases. At the end of such a protocol, the verifier should be able to determine if the prover is legitimate *and* in his vicinity.

A typical scenario for contactless authentication devices is a public transport system in which users authenticate to access buses or subway stations through their NFC-enabled smartphones. The transportation company must deploy controls to prevent misuses of its system. A legitimate user might want to help a friend to use his credentials illegally for a single trip while he is not using them, which is known as a *terrorist fraud (TF)*. Nevertheless, this user would not accept that his friend uses them at will afterwards as the original user may get caught and accountable. This attack targets the transportation company. Another threat against DB protocol is a fraudster using the presence of a legitimate user to authenticate. This is known as a *mafia fraud (MF)* and targets the transportation company as well as the end user as he may have to pay for this extra fare. These two attacks are typical examples of relay attacks against contactless authentication protocols. Another important aspect for such a system is the protection of user privacy. Most users would not accept that their whereabouts can be tracked down by other users or by the transportation company itself due the wealth of personal information that can be inferred from such data. Another scenario could be the access to a restricted building. In this context, third parties may want to enter (MF attacks), or legitimate workers may want to help friends to access (TF attacks) the building. However, the verifier is not directly a threat against the privacy of workers.

In this paper, we propose a new approach for developing provably secure DB protocols resisting to all classical threats against such protocols. The novelty of our approach is that a prover can fully control the responses to the time-critical challenges and still prove his proximity. This is particularly appropriate for coping with terrorist-fraud attacks, since these selected responses can be reused by malicious parties, only if they have been helped by the prover beforehand. Moreover, this approach is more flexible than traditional countermeasures to TF attacks, which rely on extraction mechanisms (*e.g.*, zero-knowledge proofs, secret-sharing schemes or fuzzy extractors). Indeed, these mechanisms are more complex than the ones used in this paper and the DB protocols based on them require more elaborated proofs. Furthermore, these protocols rely on long-term secret keys, thus inherently exposing the privacy and the anonymity of the provers.

Note that the TF-resistance property is a concept that is hard to formalize and numerous attempts have been made [**?**, 5, 16, 25]. Far from claiming that the approach that we propose is the only viable alternative to attaining TF-resistance, our efforts expand the fundamental understanding of this problem and how to counter it in practice. Eventually, the best approach will emerge from all these attempts.

Our main contributions can be summarized as follows.

Novel Approach. The main contribution is to propose a new approach for provable TF resistance in which the prover can unilaterally select the binary responses used during the time-critical challenge-response phases. If a malicious prover transfers this information to his accomplice, the accomplice can then *adapt and replay* the information received for a new session. We therefore obtain an intuitive TF resistance proof without any artificial extraction mechanism. Surprisingly, this idea has not been considered so far in the literature. As shown in this paper, it can be used to design protocols achieving the simulation-based TF resistance notion [15], which is a stronger notion than the ones used for most existing TF-resistant protocols. Fortunately, even if the prover is responsible for selecting the response vectors, this impacts only slightly the other security properties of our protocols.

Generic Construction. The second is TREAD (for *Terrorist-fraud Resistant and Extractor-free Anonymous Distance-bounding*), which is a generic construction implementing the proposed approach. It can be instantiated in several ways including a lightweight symmetric-key protocol, a public-key protocol protecting the privacy of provers in the presence of eavesdroppers, and a protocol based on group signatures protecting the anony-mity of the provers even against *malicious* verifiers. The latter one can be used in the public transportation scenario, whilst the first two are more adapted to the scenario of the restricted-access building.

Extension of DFKO. As a final contribution, the DFKO framework [15] is extended to deal with *distance-hija-cking (DH)* attacks [14], in which a malicious prover tries to fool a verifier on their mutual distance, by taking advantage of nearby honest provers. This provides a framework to deal with all the potential attacks against DB protocols. The security of TREAD is proven in this extended framework.

We provide a comparative analysis of our results and other well-known solutions existing in the literature in Table 1. These results are grouped into three categories: best unproved protocols, best formally-proven protocols and best privacy-preserving formally-proven protocols.

TREAD compared favourably to the best published solutions. The instance based on the group-signature scheme is fully anonymous and provides TF-resistance, in contrast to the solution presented in [18], while simply having to slightly relax the MF-resistance probability (from $\left(\frac{1}{2}\right)^n$ to $\left(\frac{3}{4}\right)^n$, which forces the number of time-critical phases to at least double to achieve the same security level). In fact, it has the best security properties of any fully anonymous protocol *without* any artificial and inefficient extraction mechanism. It almost matches the TF, MF and distance-fraud (DF) resistance of the best proven solutions [7, 16] while providing full anonymity. Finally, the instance based on the public-key scheme achieves slightly less MF resistance than the Swiss-Knife protocol attains with a symmetric key. However, note that the Swiss-Knife protocol has not been formally proven. In fact, a minor attack has been presented against it [6].

Related Work. Since the introduction of DB protocols in 1993 by Brands

Table 1: Comparison. TF denotes the terrorist-fraud resistance. The probabilities of successful mafia-fraud and distance-fraud attacks depend on the number n of time-critical rounds. P and A respectively denote privacy with respect to an eavesdropper and anonymity with respect to a malicious verifier. R denotes if a user can be revoked easily.

Protocol	**TF**	**MF**	**DF**	**P**	**A**	**R**
Not formally proven						
Swiss Knife[20]	✓	$\left(\frac{1}{2}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✗	✓
Proven-security						
SKI[7]	✓	$\left(\frac{3}{4}\right)^n$	$\left(\frac{2}{3}\right)^n$	✗	✗	✓
FO[16]	✓	$\left(\frac{3}{4}\right)^n$	$\left(\frac{3}{4}\right)^n$	✗	✗	✓
Proven-security and privacy						
privDB[24]	✗	$\left(\frac{1}{2}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✗	✓
GOR[18]	✗	$\left(\frac{1}{2}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✓	✓
PDB[1]	✓	$\left(\frac{1}{2}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✓	✗
SPADE[12]	✓	$\left(\frac{1}{2^{0.37}}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✓	✓
TREAD						
Secret key	✓	$\left(\frac{3}{4}\right)^n$	$\left(\frac{3}{4}\right)^n$	✗	✗	✓
Public key	✓	$\left(\frac{3}{4}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✗	✓
Group Signature	✓	$\left(\frac{3}{4}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✓	✓

and Chaum [10], new threats have emerged against contactless communications. They can be classified depending on whether the adversary is an external entity or a legitimate but malicious prover. The former case includes attacks in which the adversary illegitimately authenticates, possibly using a far-away honest prover (*Mafia Fraud*), or in which the adversary plays against a simplified version of the protocol without any distance estimation (*Impersonation Fraud*). The latter case includes attacks featuring a legitimate but malicious prover who wants to fool the verifier on the distance between them (*Distance Fraud*), sometimes using the presence of an honest prover close to the verifier (*Distance Hijacking*). It also tackles a malicious prover who helps an accomplice authenticate (*Terrorist Fraud*). This attack is the most difficult one to characterize and counter.

The classical countermeasure against TF relies on the assumption that a malicious prover does not have enough trust in his accomplice to simply give him directly his authentication credentials (*i.e.*, any potential long-term secret key). TF resistance is generally implemented by making the authentication of the accomplice very difficult if the prover does *not* leak away a significant fraction of his long-term key. While intuitively achieving this objective is not difficult, *proving* that a protocol is TF resistant is problematic. So far, all the proofs proposed in the literature have relied on artificial mechanisms, such as *trapdoors*, *secret leakage*, *secret sharing schemes* and *extractors*. These mechanisms allow an accomplice to extract the long-term secret key of his companion prover if he can authenticate with a non-negligible probability. Thus, once the accomplice has retrieved this key, he can impersonate at will the targeted prover. Hence, these artificial mechanisms are mainly used to deter rational provers from helping potential accomplices. For instance, Fischlin and Onete [16] proposed a special mode (*i.e.*, a trapdoor) allowing the adversary to authenticate if he knows a targeted string close in terms of Hamming distance to the long-term secret key of the prover. Very recently, Bultel and co-authors [12] used the same approach to introduce SPADE, a fully anonymous TF-resistant protocol. Unfortunately in this case, there is a trade-off to set in the analysis of the MF and TF resistance probabilities. This trade-off balances the information given to the accomplice by the prover and the information inferred from the trapdoor, which leads to unusual resistance probabilities for some properties. An important drawback of this approach is that it does not support easily scattered verifiers. In such a case, the verifiers may have to share a common decryption key to respond to the trapdoor queries. Otherwise, the accomplice would be able to impersonate his partnering prover only with the given verifier, which is a threat that the prover may accept. Finally, in this solution, a *malicious* verifier is unfortunately able to replay the received information and impersonate a given prover, representing a major threat against the latter.

In their SKI protocols [5], Boureanu, Mitrokotsa and Vaudenay employed a *leakage scheme* allowing an adversary to retrieve the long-term secret key used several times by a prover. The same technique is reused in the DB_{opt} protocols [9]. Avoine, Lauradoux, and Martin [3] used a classical *secret-sharing scheme* to resist to terrorist frauds. Their approach consists in sharing the prover's long-term secret using a (n, k) threshold cryptographic scheme. Upon reception of a challenge, the provers should send a share back to the verifier. The key-point is that an accomplice must know all the shares to be able to successfully respond to any challenge, but then he could retrieve the prover's long-term secret. In this case, the challenges sent during the time-critical phase can no longer be binary messages and in addition the scheme neither considers distance fraud, nor addresses the privacy issue. Finally, Vaudenay [25] relied on *extractor schemes* to recover a string close to the long-term secret key from the view of all nearby participants after a TF attempt. These solutions depend on computationally-expensive primitives. Overall, TREAD has therefore a simpler analysis than any of these protocols with the same security properties. Furthermore, as these solutions rely explicitly on long-term shared secret keys, they present serious challenges for developing privacy and anonymity-preserving solutions.

While a lot of effort has gone into designing secure DB protocols, the research community has only recently investigated *privacy issues* linked to distance bounding. Considering the amount of information that can be inferred from the location history of an individual [17], protecting privacy becomes a critical issue for the wide acceptance of such technology. To address this concern, two aspects have to be considered: (1) the protection of the privacy of the provers with respect to eavesdroppers and (2) the protection of the anonymity of the provers with respect to curious verifiers.

Anonymous DB protocol against external adversaries has been introduced recently [19]. Gambs, Onete, and Robert [18] extended this notion to deal with *honest-but-curious* and *malicious* verifiers, which represent a threat against the privacy of the legitimate provers as they might profile provers by linking their authentication sessions. The authors proposed an extension of the HPO protocol [19] in which the provers are managed as a group. Though they addressed the classical MF, DF and IF attacks, they did not at consider TF. Recently, Vaudenay [24] proposed a generic solution to add the privacy property to DB protocols with respect to external eavesdroppers, which relies on an authenticated key-exchange added on top of a one-time secure DB protocol. Unfortunately, it does not provide TF resistance nor anonymity against honest-but-curious or malicious verifiers.

Finally, Ahmadi and Safavi-Naini [1] gave a TF-resistant DB protocol PDB, which protects the anonymity of the prover, by fixing the weaknesses of the DBPK-log protocol [13]. Hence, the prover shows with a classical *zero-knowledge proof* that he possesses the secret key used during the protocol and its signature issued by a trusted authority. Unfortunately, this solution does not permit to revoke the credential of a specific prover without adding too much complexity and damaging the robustness of the overall scheme. In particular, since the authentication is supposed to be anonymous, there is no way to distinguish whether a session uses a given stolen secret key or not. Compared to this protocol, TREAD guarantees the anonymity of its users through a group signature scheme. This enables an efficient management of users (*i.e.*, adding and revoking users) and a clear separation of duties (*e.g.*, adding, revoking and lifting the anonymity of a prover can be done by separate authorities).

Note that overall more than forty DB protocols have been proposed since 1993. Unfortunately, based on a recent survey [11] only few of them have not been broken yet. We refer the reader to this paper for more details.

Outline. In the next section, we describe our generic construction providing TF resistance and three of its possible instantiations. Then in Section 3, we introduce the different security models and then prove the essential security properties of our construction before concluding in Section 4.

2 The TREAD instantiations

In this section, we present TREAD, a generic construction, which encompasses all the desirable properties of a secure DB protocol. To counter terrorist-fraud attack, the usual strategy is to ensure that if a malicious prover gives his accomplice both responses for a given challenge, he can recover one bit of the prover's long-term secret key x as shown in Figure 1. If the accomplice is able to authenticate with a non-negligible probability, he probably knows a large part of x and can use it to retrieve the full secret through the available extraction mechanism. Thus, any rational prover should not accept to go that far. Even though intuitively clear in general, the security of such approach is hard to prove formally. Our approach aims at avoiding this pitfall.

2.1 The Generic Construction TREAD

TREAD requires as building blocks an IND-CCA2-secure encryption scheme E (either symmetric-key or public-key) and an EUF-CMA-secure signature scheme S. The instantiations gradually move from a computationally-efficient symmetric protocol to a prover-anonymous one, in which a secure group-signature scheme is required.

As shown in Figure 2, our scheme relies on strong design choices. Our *first design choice* is to enable a prover to choose the values of the response strings

Verifier V		**Prover** P
	shared secret key: x	
$NV \xleftarrow{\$} \{0,1\}^n$	$\xleftarrow{\quad NP \quad}$	$NP \xleftarrow{\$} \{0,1\}^n$
	$\xrightarrow{\quad NV \quad}$	
	$\alpha = PRF_x(NP, NV)$	
	for $i = 0$ **to** n	
Pick $c_i \in \{0,1\}$		
Start clock	$\xrightarrow{\quad c_i \quad}$	$r_i = \begin{cases} \alpha_i & \text{if } c_i = 0 \\ \alpha_i \oplus x_i & \text{if } c_i = 1 \end{cases}$
Stop clock	$\xleftarrow{\quad r_i \quad}$	

Figure 1: The classical countermeasure against terrorist fraud: if the prover gives both possible responses, *i.e.* α_i and $\alpha_i \oplus x_i$ to his accomplice for a given c_i, he leaks one bit of his long-term authentication secret x. Note that PRF is a pseudorandom function keyed with x.

α and β, which he then sends signed and encrypted in his initial message e to the verifier. The encryption hides these values from an eavesdropper, but they can be used by the prover (or a TF accomplice) to replay the protocol. In addition, a malicious verifier could also do the same and replay the information against another verifier. The verifier simply responds to the initial message with a random binary string m to prevent trivial DF attacks in which a malicious prover selects $\alpha = \beta$. During the time-critical phases, the response to challenge c_i is computed as α_i if $c_i = 0$ and $\beta_i \oplus m_i$ otherwise.

Most existing DB protocols do not enable the prover to generate the response strings α and β, due to the fact that provers are potentially malicious and may attempt to cheat by selecting convenient values. Hence, these strings are usually computed as the output of a pseudo-random function (PRF) on nonces selected independently by the verifier and the prover. Unfortunately, this is not sufficient to prevent provers from influencing the values $\alpha||\beta$ [6, 11]. Indeed as mentioned earlier, there is a potential attack against the Swiss-Knife protocol [20] based on the use of a weak PRF [6].

Our first design choice is motivated by a simple observation. If a malicious prover can control the PRF in some cases, we can further assume that he *chooses* the response strings. If a protocol can thwart such provers, it should *a fortiori* resist to provers only manipulating the PRF.

A Novel Approach. Our *second design choice* is to allow for limited replays to achieve stronger TF resistance. This is a fundamental *shift* compared to approaches existing in the distance-bounding literature. More precisely, our strategy is not to force the prover to leak his secret to his accomplice. Rather, we design the protocol such that, if the prover helps his accomplice to authenticate, the latter can simply *replay* this information in future sessions. The difficulty is to ensure that *only* TF accomplices benefit from this strategy, and not regular external Man-in-the-Middle (MiM) adversaries.

In our construction, anyone knowing proper responses corresponding to a given initial message e (which is authenticated and encrypted by the prover, and remains opaque to a MiM adversary) can *adapt* them to any new string m generated by the verifier. This seems to go against the intuition that authentication protocols need to ensure freshness (usually through a verifier-generated nonce) to prevent replay attacks. Indeed, a MiM adversary can observe a session and learn about half the responses corresponding to a specific e. Then, he may replay e and the responses that he knows. However, this adversary must still guess on average $\frac{n}{2}$ values, which he can only do with negligible probability.

The counter-intuitive second design choice has interesting implications with regards to TF-resistance. Consider the scenario in which an accomplice is helped by a malicious prover to authenticate. If the accomplice replays the initial message e in a latter session, he would be able to adapt the information given by the prover, which allows him to re-authenticate without the help of the prover with at least the same probability as in the first attempt. Moreover, if this probability is non-negligible, he is even able to *amplify* it in such a way that, after a polynomial number of interactions with the verifier (without the prover), he gains the ability to impersonate the prover with an overwhelming probability.

Based on our design choices, we propose our generic construction TREAD. It can be instantiated with a public identity ($\mathsf{idpub}(P)$) in the classical non-anonymous case (in which the private identity $\mathsf{idprv}(P)$ is useless and can be set to *null*) or with a private identity ($\mathsf{idprv}(P)$) in the private and the anonymous settings (in which the public identity must be set to *null*). More details are given in the next section. These identities are used (among other things) to retrieve the corresponding decryption/verification keys.

Verifier V		**Prover** P
dk: decryption key vk: verification key		ek: encryption key sk: signature key $\mathsf{idpub}(P)$: public identity of P $\mathsf{idprv}(P)$: private identity of P
	Initialisation	
		$\alpha\|\|\beta \xleftarrow{\$} \{0,1\}^{2\cdot n}, \sigma_p = \mathsf{S.sig}_{\mathsf{sk}}(\alpha\|\|\beta\|\|\mathsf{idprv}(P))$
$(\alpha\|\|\beta\|\|\mathsf{idprv}(P)\|\|\sigma_p) = \mathsf{E.dec}_{\mathsf{dk}}(e)$	$\xleftarrow{e\|\|\mathsf{idpub}(P)}$	$e = \mathsf{E.enc}_{\mathsf{ek}}(\alpha\|\|\beta\|\|\mathsf{idprv}(P)\|\|\sigma_p)$
if $\mathsf{S.ver}_{\mathsf{vk}}(\sigma_p, \alpha\|\|\beta\|\|\mathsf{idprv}(P)) = 0$ then abort		
$m \xleftarrow{\$} \{0,1\}^n$	$\xrightarrow{m}$	
	Distance Bounding for $i = 0$ to n	
Pick $c_i \in \{0,1\}$		
Start clock	$\xrightarrow{c_i}$	$r_i = \begin{cases} \alpha_i & \text{if } c_i = 0 \\ \beta_i \oplus m_i & \text{if } c_i = 1 \end{cases}$
Stop clock	$\xleftarrow{r_i}$	
store Δt_i		
	Verification	
If $\#\{i : r_i \text{ and } \Delta t_i \text{ correct}\} = n$ then		
$\mathsf{Out_V} := 1$; else $\mathsf{Out_V} := 0$	$\xrightarrow{\mathsf{Out_V}}$	

Figure 2: Our generic and provably secure DB construction TREAD built from an $\mathsf{IND\text{-}CCA2}$-secure encryption scheme E and an $\mathsf{EUF\text{-}CMA}$-secure signature scheme S. We use $||$ for the concatenation operation.

Definition 1 (TREAD)**.** The construction TREAD is composed of five algorithms and parametrized by an $\mathsf{IND\text{-}CCA2}$-secure encryption scheme E, an $\mathsf{EUF\text{-}CMA}$-secure signature scheme S, as well as a definition for $\mathsf{idprv}(\cdot)$ and $\mathsf{idpub}(\cdot)$ and a distance bound d_{max} such that messages cover this distance within a time $\frac{t_{\mathsf{max}}}{2}$.

$\mathsf{DB.gen}(1^\lambda)$ is the algorithm run by an honest party, setting up the encryption scheme E and the signature scheme S for a security parameter λ. It returns the number of the time-critical phases n, which is a function of λ.

$\mathsf{DB.prover}(\mathsf{ek}, \mathsf{sk})$ is the algorithm executed by the prover described in Figure 2. The prover starts by drawing a random value $\alpha||\beta$ from the uniform distribution on $\{0,1\}^{2 \cdot n}$. Then, he computes a signature σ_p on it with $\mathsf{S.sig}_{\mathsf{sk}}(\alpha||\beta||\mathsf{idprv}(P))$. Afterwards, he generates $e = \mathsf{E.enc}_{\mathsf{ek}}(\alpha||\beta||\mathsf{idprv}(P)||\sigma_p)$ and sends $e||\mathsf{idpub}(P)$. Finally, during the n time-critical phases, he receives a challenge bit c_i and responds with $r_i = (\alpha_i \wedge \neg c_i) \vee ((\beta_i \oplus m_i) \wedge c_i)$.

$\mathsf{DB.verifier}(\mathsf{ID}, \mathsf{dk}, \mathsf{vk}, \mathsf{UL}, \mathsf{RL})$ is the algorithm executed by the verifier interacting with a prover identified as ID. Depending on the context, this identifier can be directly the identity of a prover ($\mathsf{idpub}(P)$), but it can also be the name of a group ($\mathsf{idprv}(P)$) for anonymous authentication. Moreover depending on the context, the verifier has access to the lists of legitimate provers UL and revoked ones RL. He expects an initial message e and deciphers it as $(\alpha||\beta||\mathsf{idprv}(P)||\sigma_p) = \mathsf{E.dec}_{\mathsf{dk}}(e)$. If σ_p is invalid (*i.e.*, $\mathsf{S.ver}_{\mathsf{vk}}(\sigma_p, \alpha||\beta||\mathsf{idprv}(P)) = 0$), the verifier aborts. Otherwise, he picks a random bit string m from the uniform distribution on $\{0,1\}^n$ and sends it. Afterwards, during the n time-critical phases, he generates a random bit c_i from a uniform distribution, starts his clock, sends c_i, gets back r_i, stops his clock and stores the corresponding time Δt_i. Finally, he verifies that (1) $\Delta t_i \leq t_{\mathsf{max}}$ and (2) $r_i = (\alpha_i \wedge \neg c_i) \vee ((\beta_i \oplus m_i) \wedge c_i)$, for all $i \leq n$. If these conditions hold, he sends an accepting bit $\mathsf{Out_V} = 1$, while otherwise he sends $\mathsf{Out_V} = 0$.

$\mathsf{DB.join}(\mathsf{ID}, \mathsf{UL})$ is the algorithm to register a new prover with identifier ID in the list UL. It returns the keys $(\mathsf{ek}, \mathsf{dk})$ for E and $(\mathsf{sk}, \mathsf{vk})$ for S. Depending on the primitives E and S, dk and vk may be public or private, and can sometimes be equal respectively to ek and sk.

$\mathsf{DB.revoke}(\mathsf{ID}, \mathsf{UL}, \mathsf{RL})$ is the algorithm to revoke a legitimate prover with identifier ID in UL and transfer him to the revoke list RL.

These last two algorithms depend heavily on the instance of the protocol and are described in more details in the following section. Note that TREAD adopts the *sign-then-encrypt* paradigm instead of the more usual *encrypt-then-sign*. If the latter approach were used, an eavesdropper would be able to infer easily the identity of any prover, by simply verifying the signature on the message e with all the public keys listed in UL. The security is nonetheless preserved, at the cost of using an IND-CCA2 secure encryption scheme.

2.2 Instantiations

Our instantiations go from a computationally-efficient sym-metric-key protocol to a prover-anonymous one.

Efficient symmetric-key scheme. Computational efficiency is critical for the design of DB protocols as they are usually used in resource-limited devices. To obtain an optimal construction, TREAD can be instantiated with an IND-CCA2 symmetric-key encryption scheme SKE and an EUF-CMA message-authentication code scheme MAC. In this case, the public identity $\mathsf{idpub}(P)$ is the identity of the prover and the private identity $\mathsf{idprv}(P)$ is set to *null*. Since SKE and MAC are symmetric, we have $\mathsf{ek} = \mathsf{dk}$ and $\mathsf{sk} = \mathsf{vk}$. Thus, the prover and the verifier have the same *symmetric* key $\mathsf{k} = (\mathsf{ek}, \mathsf{sk})$. In this construction, the verifiers have access to a private list UL containing all the secret keys of legitimate provers. An authority should add any prover in the private list UL or in the revokation public list RL.

Prover privacy and public-key encryption. In applications such as contactless payment schemes, shared secret keys cannot be used. Thus, with the emergence of NFC-enabled smartphones, public-key DB protocols are crucial.

TREAD can be instantiated with an $\mathsf{IND\text{-}CCA2}$ public-key encryption PKE and an $\mathsf{EUF\text{-}CMA}$ digital signature scheme $\mathsf{S\text{-}SIG}$. In this case, the public identity $\mathsf{idpub}(P)$ is set to *null*, and the private one $\mathsf{idprv}(P)$ is the identity of P (or his verification key). The keys ek and dk are respectively the public and the private keys of the verifier, and sk and vk are the (private) signature key and the (public) verification key of the prover. With such a protocol, two sessions by the same user are not linkable for an *external* eavesdropper as the only information sent about the prover's identity is encrypted with the public-key of the verifier. However, verifiers have the power to link sessions. In this construction, the verifiers have access to a public list UL containing the public keys of legitimate provers. An authority should add any prover in the public list UL or in the revokation public list RL.

Prover anonymity and group signature. TREAD can be used to provide full prover-anonymity with respect to a malicious verifier. As profiling users is now common, it is crucial to develop anonymity-preserving DB protocols. Both prover anonymity and revocability can be achieved by instantiating TREAD with an $\mathsf{IND\text{-}CCA2}$ public-key encryption scheme PKE and a revocable group signature scheme $\mathsf{G\text{-}SIG}$. In this case, the public identity $\mathsf{idpub}(P)$ is set to *null*, and the private identity $\mathsf{idprv}(P)$ is set to the identity of the group ID_G. Independent groups may coexist but prover-anonymity with respect to the verifier is only guaranteed up to a prover's group. The keys ek and dk are respectively the public and private keys of the verifier, sk is the prover's signing key and vk is the public group verification key. Group signature schemes allow a user to anonymously sign on behalf of a group he belongs to. Hence, the verifier can check if the prover belongs to the claimed group, but cannot identify him precisely nor link his sessions. In this scenario, the join and revoke algorithms take their full meaning.

Let $(\mathsf{gpk}, \mathsf{msk})$ be the group/master key pair of the group signature scheme $\mathsf{G\text{-}SIG}$. Then,

$\mathsf{DB.join}_{\mathsf{msk}}(\mathsf{ID}, \mathsf{gpk}, \mathsf{UL})$ *returns a prover signing key* $\mathsf{sk}_{\mathsf{ID}}$ *for* P_{ID}*. It also outputs a value* $\mathsf{reg}_{\mathsf{ID}}$ *and adds* P_{ID} *to* UL*.*

$\mathsf{DB.revoke}_{\mathsf{msk}}(\mathsf{ID}, \mathsf{gpk}, \mathsf{RL}, \mathsf{UL})$ *computes the logs* $\mathsf{rev}_{\mathsf{ID}}$ *for* P_{ID}*, using* $\mathsf{reg}_{\mathsf{ID}}$ *and* msk*, and moves* P_{ID} *from* UL *to* RL*.*

3 Models and Security Proofs

In this section, we describe the models for defining DB protocols and to characterize the classical threats against these protocols. Then, we prove the main security properties of the instantiations of our TREAD construction.

3.1 Formal Security Models

To the best of our knowledge three security models exist for distance bounding: the original one by Avoine and co-authors [2], the second one by Dürholz, Fischlin, Kasper and Onete [15] (DFKO), and the last one by Boureanu, Mitrokotsa and Vaudenay [5]. In this paper, we use the DFKO model and its extension for a strong TF-resistance notion (SimTF) proposed by Fischlin and Onete [16]. The DFKO model is even extended to deal with DH attacks [14]. Finally, we introduce the privacy and anonymity models derived from the work of Gambs, Onete and Robert [18]. These models are compatible with the proposed extension of the DFKO model and rely on classical security definitions given in Appendix A.

Distance-Bounding Protocols. DB protocols are interactive protocols running between two participants. The objective of the *prover* P is to prove to the *verifier* V that he is legitimate and located at a distance at most d_{max}. The participants interact during *rounds*, defined as sequences of messages. For some of these rounds, the verifier uses a *clock* to measure the time elapsed between the emission of a challenge c_i and the reception of the corresponding response r_i. These back-and-forth rounds are referred to as *time-critical* rounds while otherwise they are refer to as *slow phases*. In most protocols, the *DB phase* of a protocol is composed of either n independent time-critical rounds or only one combined time-critical round. Having measured the elapsed time at the end of each time-critical round, the verifier then compares this value to a threshold t_{max} associated with the maximal allowed distance d_{max}. If at least one of these tests fails, the prover will not be considered in the vicinity of the verifier.

The verifier is assumed to behave honestly during the authentication of a prover. However, he may try to lift the *anonymity* of a prover if this is possible. In such a case, the verifier may try to link sessions to a given prover. Additionally, provers can potentially behave maliciously and attempt to fool the verifier, either by themselves or by using (voluntary or unwilling) accomplices.

Adversary Model. In this DFKO model, an adversary can interact with provers and verifiers in three kinds of sessions:

- *Prover-verifier* sessions, in which he observes an honest execution of the protocol between a prover and a verifier.
- *Prover-adversary* sessions, in which he interacts with a honest prover as a verifier.
- *Adversary-verifier* sessions, in which he interacts with a legitimate verifier as a prover.

Each session is associated with a unique identifier sid.

The adversaries are defined in terms of their computational power t, the number of prover-verifier sessions q_{obs} they may observe, the number q_{v} of adversary-verifier sessions and the number q_{p} of prover-adversary sessions they initiate, and their winning advantage for the corresponding security games.

To capture the notion of relays, the DFKO framework uses an abstract clock keeping track of the sequence of the adversary's actions. It is given as a function

$\mathsf{marker} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, such that $\mathsf{marker}(\cdot, \cdot)$ is strictly increasing. It can be used to define *tainted* time-critical rounds. This indicates that an attack scenario is ruled out by definition, due for instance to the verifier's ability to detect pure relays through his accurate clock. More precisely, an adversary cannot win a game in a tainted session. In the following definitions, $\Pi_{\mathsf{sid}}[i, \ldots, j]$ denotes a sequence of messages $(m_i, \ldots, m_j)$ exchanged during the session sid of the protocol.

Following the terminology introduced by Vaudenay [23] and later re-used to define prover-anonymity [19], if an adversary is assumed to know the final result of an authentication session (*i.e.*, accept or reject), he is said to be *wide*, while otherwise he is *narrow*. Orthogonally, if the adversary may never corrupt provers, he is considered to be *weak* while if a corruption query is only followed by other such queries, the adversary is *forward*. Finally, if there is no restriction on the corruption queries, the adversary is said to be *strong*. In this paper, we consider the strongest adversary model possible, namely *wide-strong* adversaries.

Security analysis. We give the proofs of the main properties of our construction: (1) TF resistance, (2) MF resistance, (3) DH resistance (implying DF resistance), (4) prover privacy and finally (5) prover anonymity. In the context of this paper, the last property is the strongest one as it protects the privacy of the provers against the verifiers themselves.

The slow-phase impersonation-security threat is discarded in our analysis [15]. This notion has been introduced for resource-limited provers and states that the authentication of a prover should be difficult even if only a reduced number of time-critical rounds is supported. It is relatively ambiguous and makes slow-phase impersonation resistance hard to achieve. Furthermore, having multiple rounds of communication is no longer a problem for contactless devices, which are faster and more efficient in their interactions. Therefore, we believe that the need for slow-phase authentication is no longer a constraint for the design of DB protocols.

Game structure. The threat models are represented as security games involving an adversary $\mathcal{A}$ and a challenger simulating the environment for him. All these game-based proofs start with the challenger building the simulation environment using $\mathsf{DB.gen}(1^\lambda)$. For clarity, this step is omitted in their descriptions. The adversary interacts with the simulated environment through oracles that he is allowed to run concurrently. These include a prover oracle (for prover-adversary sessions), a verifier oracle (for adversary-verifier sessions) as well as a session oracle to simulate an honest exchange between the prover and the verifier. The challenger may have to simulate the following oracles:

$\mathsf{Verifier}$ *runs the protocol* $\mathsf{DB.verifier}(\mathsf{ID}, \mathsf{dk}, \mathsf{vk}, \mathsf{UL}, \mathsf{RL})$.

$\mathsf{Prover}(\cdot)$ *runs the protocol* $\mathsf{DB.prover}(\mathsf{ek}, \mathsf{sk})$.

$\mathsf{Session}(\cdot)$ *returns the transcript of a new honest run of the protocol* $\mathsf{DB.auth}(R, n)$.

$\mathsf{Join}^c(\cdot)$ *simulates the join of a corrupted prover* U_i *by running* $\mathsf{DB.join}(i, \mathsf{UL})$ *and returning the secret keys.*

$\mathsf{Corrupt}(\cdot)$ *simulates the corruption of a prover* U_i *by returning his secret keys.*

Notation. In what follows, q_{p} (respectively q_{v}) refers to the number of times the prover (respectively verifier) is used.

3.2 Terrorist-Fraud Resistance

Dürholz, Fischlin, Kasper and Onete defined the simulation-based TF-resistance notion SimTF [15]. In this model, a far-away malicious prover P wants to use an *accomplice* $\mathcal{A}$ close to the verifier to authenticate. For any rational prover P, $\mathcal{A}$ should not receive during the attack enough information allowing him to impersonate P later on in any MF or IF. This is formalized as a two-phase game. During the first phase, $\mathcal{A}$ tries to authenticate with the help of P. Let $p_{\mathcal{A}}$ denote his success probability. During the second phase, a simulator S takes the internal view of $\mathcal{A}$ and tries to authenticate without any interaction with any other legitimate prover. Let p_S denote its success probability. The TF attack by the pair $(P, \mathcal{A})$ is *successful*, if the help of P during the attack does make any difference in the attack (*i.e.*, if $p_{\mathcal{A}} > p_S$).

In this attack model, the malicious prover is not allowed to communicate with his accomplice at all during the time-critical phases. Thus, any communication between them during any time-critical phase taints the session, which can be formalized by the following definition:

Definition 2 (Tainted Session (TF))**.** An adversary-verifier session sid, with time-critical phases $\Pi_{\mathsf{sid}}[k, k+1] = (m_k, m_{k+1})$, for $k \geq 1$, with the k-th message being received by the adversary, is *tainted* if there is a session sid' between the adversary and P such that, for any i,

$$\mathsf{marker}(\mathsf{sid}, k) < \mathsf{marker}(\mathsf{sid}', i) < \mathsf{marker}(\mathsf{sid}, k+1).$$

This definition is very strong since a *single* interaction between the accomplice and the prover, while the accomplice is running a time-critical round in an adversary-verifier session sid, is enough to taint *all* the time-critical rounds of sid. The malicious prover may *not* have any feedback from his accomplice during the time-critical phases of the protocol, making the prover's strategy non-adaptive to the challenges sent by the verifier. This simplifies the construction of a simulator that can match the adversary's winning probability.

The TF-resistance notion SimTF can be defined as follows:

Definition 3 (TF Resistance)**.** For a DB authentication scheme DB, a $(t, q_{\mathsf{v}}, q_{\mathsf{p}}, q_{\mathsf{obs}})$-terrorist-fraud adversary pair $(\mathcal{A}, P)$ and a simulator S running in time t_S, the malicious prover P and his accomplice $\mathcal{A}$ win against DB if $\mathcal{A}$ authenticates in at least one of q_{v} adversary-verifier sessions *without tainting it* with probability $p_{\mathcal{A}}$, and if S authenticates in one of q_{v} sessions with the view of $\mathcal{A}$ with probability p_S, then $p_{\mathcal{A}} \leq p_S$.

As stated in Table 1, TF resistance is a binary property. Indeed, the accomplice (*i.e.*, the simulator) is either able to impersonate independently the prover with at least the same probability in later sessions having the initial information received from the prover (*i.e.*, TF-resistant) or not.

We first prove that the TREAD construction is SimTF-resistant without using any artificial extraction mechanism. This simply means that if the prover gives some information to his accomplice to succeed in the first phase of the TF attack, his accomplice can succeed similarly later without the help of the prover.

Theorem 4. *If the challenges c_i are drawn uniformly at random by the verifier,* TREAD *is* SimTF*-resistant.*

Proof. The theorem simply states that, for any prover P helping an adversary $\mathcal{A}$ to authenticate in a session sid, there exists a simulator sim that can perform at least as good as $\mathcal{A}$ by using him as a black-box.

Let p be the initial success probability of $\mathcal{A}$ with the help of P in a session sid. Let sid' denote a new session played *a posteriori* by the simulator sim with the verifier V. Assume that m is the initial message sent by V in sid and m' is the corresponding message sent by V in sid'.

To build sim, the idea is to place $\mathcal{A}$ in the same situation as in sid. The first step is to *rewind* $\mathcal{A}$ to his initial state, after it received information from P and sent e in sid. Then, sim sends m to $\mathcal{A}$, even though V has sent a different m' to sim. If P sent any additional message to $\mathcal{A}$ in sid before the beginning of the time-critical phases, sim relays it to $\mathcal{A}$. Hence, from $\mathcal{A}$'s point of view, this is the same as in sid.

Next, the simulator sim simply forwards the challenges c_i from V to $\mathcal{A}$. If $c_i = 0$, sim sends the response r_i of $\mathcal{A}$ to V. Otherwise, if $c_i = 1$, sim needs to adapt the response to m': he sends $r'_i = r_i \oplus m_i \oplus m'_i$.

Using this strategy, it is clear that sim can respond to any challenge with a probability at least equal to that the success probability of $\mathcal{A}$. Hence, sim can authenticate sid' with a probability $\mathsf{p}_{\mathsf{sim}}$, such that $\mathsf{p}_{\mathsf{sim}} \geq \mathsf{p}$. □

This result relies on a naïve simulator, which can only win with the same probability as the accomplice $\mathcal{A}$. While this is sufficient to prove the result, a more advanced simulator can *amplify* any non-negligible advantage of $\mathcal{A}$ until it becomes overwhelming after a polynomial number of sessions to the verifier oracle and no further session with the prover himself. Therefore, no rational prover should attempt any TF attack with an accomplice, since any non-negligible success probability in the first phase of the attack can lead to successful impersonation attacks by the accomplice.

Theorem 5. *For any adversary $\mathcal{A}$ authenticating with the help of a prover with non-negligible probability, there is an algorithm* $\mathsf{amplify}$ *using the internal view of $\mathcal{A}$ and oracle access to a verifier, such that after a polynomial number of steps,* $\Pr[\mathsf{amplify}$ *authenticates* $] = 1$*, almost surely.*

The objective of the proof is to show that the simulator can retrieve the response vectors associated with the message e, allowing successful impersonations afterwards.

Proof. Let $\mathcal{A}$ be the accomplice of a malicious prover P trying to conduct a TF attack, and $Sim_{\mathsf{TF}}()$ be a simulator having access to the same *internal view* of $\mathcal{A}$. Assume that $\mathcal{A}$ can only access the prover before the time-critical phases. Then, he starts this phase with an initial knowledge IK given by P, and succeeds with a probability p_{TF}. This information IK (*i.e.*, the internal view of $\mathcal{A}$) can be described as one of these two possibilities:

- The prover sends two n-bit vectors to his accomplice: $\mathbf{c}^0$ and $\mathbf{c}^1$. These vectors represent respectively the (not necessarily correct) responses to the 0-challenges and the 1-challenges.

- The prover sends the description of an algorithm A to generate these vectors.

Intuitively, if A is *memoryless* (*i.e.*, the response to the i^{th} challenge does not depend on the previous challenges), the two cases are equivalent. Since the challenges are drawn honestly by the verifier, they are unpredictable and independent. Thus, the memoryless hypothesis is reasonable and the information IK can be described as $\mathsf{IK} = (\mathbf{c}^0, \mathbf{c}^1)$.

The information provided to $\mathcal{A}$ can be described by two different scenarii. In the *missing bits* scenario, either $\mathbf{c}_i^0 = \alpha_i$ and $\mathbf{c}_i^1 = \beta_i \oplus m_i$ ($\mathsf{Case\ 1}$ – *full information*), or $\mathbf{c}_i^0 = \alpha_i$ and $\mathbf{c}_i^1 = \perp$ or, equivalently, $\mathbf{c}_i^0 = \perp$ and $\mathbf{c}_i^1 = \beta_i \oplus m_i$ ($\mathsf{Case\ 2}$ – *partial information*). For a round, the probability that $\mathcal{A}$ responds correctly to the verifier's challenge is either 1 in $\mathsf{Case\ 1}$ or $\mathsf{q}_{\mathsf{Case2}} = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$ in $\mathsf{Case\ 2}$.

In the *flipped bits* scenario, $\mathsf{Case\ 2}$ is redefined as $\mathbf{c}_i^0 = \alpha_i$ and $\mathbf{c}_i^1 = \overline{\beta_i \oplus m_i}$ or $\mathbf{c}_i^0 = \overline{\alpha_i}$ and $\mathbf{c}_i^1 = \beta_i \oplus m_i$. The probability $\mathsf{q}_{\mathsf{Case2}}$ is then equal to $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0 = \frac{1}{2}$, *i.e.*, the probability that the verifier is asking for the unflipped bit.

The following lemma follows straightforwardly:

Lemma 6. *Assume that a malicious prover gives to his accomplice $\mathcal{A}$ the vectors $(\mathbf{c}^0, \mathbf{c}^1)$ s.t.* $\mathsf{Case\ 1}$ *has been used $n-r$ times and* $\mathsf{Case\ 2}$ *r times. The probability that the TF succeeds is* $\mathsf{p}_{\mathsf{TF}} = \Pr[\mathcal{A} \text{ is not detected}] = 1^{n-r} \cdot \mathsf{q}_{\mathsf{Case2}}^r$.

Assume now that r is such that $\mathsf{q}_{\mathsf{Case2}}^r$ is non-negligible (*i.e.*, $\exists c, \forall n_c, \exists n > n_c, \mathsf{q}_{\mathsf{Case2}}^r \geq \frac{1}{n^c}$). This hypothesis implies that $r \in O(\log n)$ – in both scenarii.

Consider now the simulator $Sim_{\mathsf{TF}}(e, \mathbf{c}^0, \mathbf{c}^1)$ that tries to impersonate P to the verifier with no further interaction with P. The simulator must authenticate with the same probability p_{TF} that $\mathcal{A}$ had to succeed before helped by P.

This advantage can be amplified arbitrarily close to one. We define a simulator $\mathsf{amplify}(e, \mathbf{c}^0, \mathbf{c}^1)$ using $Sim_{\mathsf{TF}}(e, \mathbf{c}^0, \mathbf{c}^1)$ internally. This simulator can try $k \cdot n \cdot n^c$ authentication experiments with the verifier, for some constant $k > 2$. Thus, $\mathsf{amplify}(e, \mathbf{c}^0, \mathbf{c}^1)$ should win at least n experiments with an overwhelming probability.

For any challenge, the prover can either send beforehand both valid answers, only one valid answer, or one valid one and one invalid one, which can not be distinguished one for another. Hence, for memoryless information, these scenarios are the only ones possible. There is no value to send only one invalid answer or both invalid answers to the accomplice. This assumes that the prover does get any clue from the verifier on which challenges have been rejected.

In fact, we have the following lemma:

Lemma 7. *For a valid view $(e, \mathbf{c}^0, \mathbf{c}^1)$, the probability that* $\mathsf{amplify}(e, \mathbf{c}^0, \mathbf{c}^1)$ *wins less than n of the $k \cdot n \cdot n^c$ experiments is less than $e^{-\frac{kn}{2}\left(\frac{k-1}{k}\right)^2}$.*

The last lemma is derived from the Chernoff bound. The average number of wins μ should be equal to $k \cdot n \cdot n^c \cdot \frac{1}{n^c} = k \cdot n$. In contrast, $n = (1 - \delta) \cdot \mu$, if $1 - \delta = \frac{1}{k}$ and thus $\delta = \frac{k-1}{k}$. The lemma follows directly and, as a corollary, if $k = 4$, the probability is smaller than $\frac{1}{e^{1.125n}} < \frac{1}{2^n}$.

Assume that $\mathsf{amplify}(e, \mathbf{c}^0, \mathbf{c}^1)$ has won n independent experiments. Their independence follows from the initial assumption on the independence of the challenges. Depending on the scenarios, two different questions have to be considered. In the *missing bits* scenario, $\mathcal{A}$ has to properly guess different missing bits (*i.e.*, α_{i_j} or $\beta_{i_j} \oplus m_{i_j}$, for $1 \leq j \leq r$). We need to compute the probability that $\mathcal{A}$ has discovered all his r missing bits. Consider a missing bit b_{i_j}, for $1 \leq j \leq r$. After n successful experiments, $\mathcal{A}$ does not have discovered the bit only if the verifier has always asked for the opposite known bit – let $\mathcal{E}_j$ be such an event. This happens only with probability $\Pr[\mathcal{E}_j] = \frac{1}{2^n}$.

The following result follows directly from the union bound for finite sets of events:

Lemma 8. *Assume that* $\mathsf{amplify}(e, \mathbf{c}^0, \mathbf{c}^1)$ *has won n experiments. Thus, he should have discovered all its r missing bits (of* Case 2*) with an overwhelming probability. In fact, the probability that some bits are still unknown is simply* $\Pr[\cup_j \mathcal{E}_j] \leq \sum_j \Pr[\mathcal{E}_j] = \frac{r}{2^n}$.

Using the last two lemmas, we obtain the next result:

Lemma 9. *Assume that* $\mathsf{amplify}(e, \mathbf{c}^0, \mathbf{c}^1)$ *has done $4 \cdot n \cdot n^c$ authentication experiments. After these experiments, he should have retrieved all its r* missing bits *and be able to impersonate P with an overwhelming probability. Thus,*

$$\mathsf{Adv}^{\mathit{MF}}_{\mathit{amplify},\mathsf{TREAD}}(n) \geq \left(1 - \frac{1}{2^n}\right) \cdot \left(1 - \frac{r}{2^n}\right) > 1 - \frac{r+1}{2^n}.$$

In the *flipped bits* scenario, the verifier should not have asked for any flipped bit in the winning experiments. In these experiments, $\mathsf{amplify}(e, \mathbf{c}^0, \mathbf{c}^1)$ may assume that the bits for which the verifier has always asked the same challenges in fact correspond to the instances of Case 2. The simulator would be wrong only if the verifier has asked always for the same challenges for at least one of the $n - r$ instances of Case 1 (this happens with a probability at most $\frac{n-r}{2^n}$). Thus, $\mathsf{Adv}^{\mathsf{MF}}_{\mathsf{amplify},\mathsf{TREAD}}(n) \geq 1 - \frac{n-r+1}{2^n}$ in this case, which concludes the proof of Theorem 5. $\square$

3.3 Mafia Fraud

During an MF, an active MiM adversary, interacting with a single prover and a single verifier during several sessions, tries to authenticate. However, he is not able to purely relay information between the verifier and the prover during the time-critical phases. To discard such attacks, the tainted time-critical phases are redefined as follows.

Definition 10 (Tainted Time-Critical Phase (MF))**.** A time-critical phase $\Pi_{\mathsf{sid}}[k, k+1] = (m_k, m_{k+1})$, for $k \geq 1$, of an adversary-verifier session sid, with the message m_k being received by the adversary as the k^{th} challenge from the verifier, is *tainted by the time-critical phase* $\Pi_{\mathsf{sid}^*}[k, k+1] = (m^*_k, m^*_{k+1})$ *of a prover-adversary session* sid^* if

$$\begin{aligned} & (m_k, m_{k+1}) = (m^*_k, m^*_{k+1}), \\ & \mathsf{marker}(\mathsf{sid}, k) < \mathsf{marker}(\mathsf{sid}^*, k), \\ \text{and} \quad & \mathsf{marker}(\mathsf{sid}, k+1) > \mathsf{marker}(\mathsf{sid}^*, k+1). \end{aligned}$$

Once this definition is given, the game-based definition of MF resistance notion can be stated as follows.

Definition 11 (MF Resistance)**.** For a DB authentication scheme DB, a $(t, q_{\mathsf{v}}, q_{\mathsf{p}}, q_{\mathsf{obs}})$-MF adversary $\mathcal{A}$ wins against DB if the verifier accepts $\mathcal{A}$ in one of the q_{v} adversary-verifier sessions sid, which does not have any critical phase tainted by a prover-adversary session sid^*. Thus, the MF-resistance is defined as the probability $\mathbf{Adv}^{\mathsf{MF}}_{\mathsf{DB}}(\mathcal{A})$ that $\mathcal{A}$ wins this game.

We now prove that TREAD is MF-resistant.

Theorem 12. *If the challenges are drawn randomly from a uniform distribution by the verifier,* E *is an* $\mathsf{IND\text{-}CCA2}$*-secure encryption scheme and* S *is* $\mathsf{EUF\text{-}CMA}$*-secure, then* TREAD *is MF resistant and*

$$\mathsf{Adv}^{MF}_{\mathsf{TREAD}}(\lambda) \leq \frac{q_p^2}{2^{2n}} + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{S}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{E}}(\lambda) + \left(\frac{3}{4}\right)^n.$$

The prover and verifier oracles are simulated as defined in Section 2, except that after generating e, the prover adds an entry to a *witness list* WL containing $(e, \alpha||\beta)$.

The proof of the above theorem is more complex than others. It can be reduced to the security analysis of a simpler version of the protocol, using the game-hopping technique formalized by Shoup in [22]. In essence, the initial security game Γ_0 is reduced to a final game in which the adversary has no information (other than by guessing) about the values α and β before the DB phase. This is done by reducing his means of attacks at each game (*e.g.* by forbidding nonces reuse from prover oracles), while showing that the resulting loss is negligible. More formally, if $Pr[\Gamma_i]$ represents the winning probability of the adversary $\mathcal{A}$ in the game Γ_i, the transition between Γ_i and Γ_{i+1} is such that $|\Pr[\Gamma_i] - \Pr[\Gamma_{i+1}]| \leq \epsilon_\lambda$, in which ϵ_λ is a negligible function of λ.

Proof. We start from the initial game Γ_0 as given in Definition 11 and build the following sequence of games.

Γ_1: *In this game, no value $\alpha||\beta$ is outputted more than once by the prover oracle.*

In the i^{th} session, the probability to have a collision with any of the previous $i-1$ $\alpha||\beta$ values is bounded by $\frac{i}{2^{2 \cdot n}}$. If $\mathcal{A}$ runs q_p prover sessions, the probability of a collision for a given session is bounded by $\frac{q_p}{2^{2 \cdot n}}$. From the union bound, the probability that a collision occurs at least once is bounded by $\sum_{i=0}^{q_p} \frac{q_p}{2^{2 \cdot n}}$, which is in turn bounded by $\frac{q_p^2}{2^{2n}}$. Thus, using Shoup's difference lemma, $|\Pr[\Gamma_0] - \Pr[\Gamma_1]| \leq \frac{q_p^2}{2^{2n}}$, which is negligible.

Γ_2: *This game aborts if σ_p was not generated by the prover oracle, and* $\mathsf{S.ver}_{\mathsf{vk}}(\sigma_p, \alpha||\beta) \neq 0$.

In this game, we rule out the possibility that $\mathcal{A}$ produces a valid signature without the key, which is trivially forbidden by the $\mathsf{EUF\text{-}CMA}$ resistance of S. The reduction simply consists in starting $\mathsf{EUF\text{-}CMA}$ experiments (one for each prover) with a challenger and using queries to the corresponding signing oracle to generate the signatures of a prover. Then, if $\mathcal{A}$ sends a valid signature on behalf of one of the provers, we can return it to the challenger and win the $\mathsf{EUF\text{-}CMA}$ experiment. From the difference lemma, we have $Pr[\Gamma_1] - Pr[\Gamma_2] \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{S}}(1^\lambda)$, which is negligible by hypothesis.

Γ_3: *In this game, e is replaced by the encryption of a random string (of equal length).*

This transition is based on indistinguishability, aiming at removing any leakage of $\alpha||\beta$ from e by making $\alpha||\beta$ only appear during the DB phase. We prove that the probability $\epsilon = Pr[\Gamma_3] - Pr[\Gamma_2]$ is negligible by building a distinguisher $\mathcal{B}$ such that its advantage against the $\mathsf{IND\text{-}CCA2}$ experiment is polynomial in ϵ. Hence, if ϵ is non-negligible, we reach a contradiction. By assumption, the advantage of any adversary against the $\mathsf{IND\text{-}CCA2}$ experiment on E is negligible.

To build $\mathcal{B}$, we replace $\mathsf{E.enc_{ek}}(\alpha||\beta||\mathsf{idprv}(P)||\sigma_p)$ by a string given by the $\mathsf{IND\text{-}CCA2}$ challenger. Using the adversary $\mathcal{A}$, the distinguisher $\mathcal{B}$ can be built as follows.

Prover simulation: the reduction $\mathcal{B}$ generates two challenge messages: $m_0 = (\delta||\mathsf{idprv}(P)||\mathsf{S.sig_{sk}}(\delta||\mathsf{idprv}(P)))$ and $m_1 = (\alpha||\beta||\ \mathsf{S.sig_{sk}}(\alpha||\beta||\ \mathsf{idprv}(P)))$, in which $\alpha||\beta$ and δ are random binary strings of length $2 \cdot n$. Then, he sends them to the challenger to obtain c_b, the encryption of m_b (depending on a random bit b picked by the challenger before the experiment). He also adds $(c_b, \alpha||\beta)$ to the list WL. Afterwards, he sends c_b as the initial message and uses $\alpha||\beta$ during the challenge-response phase.

Verifier simulation: When the verifier oracle gets the initial message e, he reads the tuple $(e, \alpha||\beta)$ in WL and uses the corresponding $\alpha||\beta$ to verify the responses. If no such tuple exists, then he is allowed to use the decryption oracle on e (as it is not a challenge c_b). As Γ_2 enforces that only invalid or prover generated signatures are contained in e, then either $\mathcal{A}$ loses for sending an invalid signature, or e is a new encryption for values contained in one of the challenges. In the latter case, $\mathcal{B}$ readily obtains the bit b by verifying whether the decryption of e corresponds to a m_0 or a m_1.

Return value: $\mathcal{B}$ returns $\mathsf{Out_V}$.

If $b = 1$, $\mathcal{B}$ simulates Γ_2 (e is the encryption of $\alpha||\beta$). In this case, $\mathcal{B}$ wins if $\mathsf{Out_V} = 1$. By definition, $Pr[\mathsf{Out_V} = 1]$ in $\Gamma_2 = Pr[\Gamma_2]$. Otherwise, if $b = 0$, then $\mathcal{B}$ simulates Γ_3 (e is the encryption of δ). In this case, $\mathcal{B}$ returns 0 if $\mathcal{A}$ loses (*i.e.*, with probability $1 - Pr[\Gamma_3]$). The winning probability of $\mathcal{B}$ is then $\frac{Pr[\Gamma_2]+1-Pr[\Gamma_3]}{2} = \frac{1+(Pr[\Gamma_2]-Pr[\Gamma_3])}{2}$, giving an advantage of $\epsilon = Pr[\Gamma_2] - Pr[\Gamma_3]$. It follows that any significant probability difference between the two games can be transformed into an $\mathsf{IND\text{-}CCA2}$ advantage and $|Pr[\Gamma_2] - Pr[\Gamma_3]| \leq \mathsf{Adv_E^{IND\text{-}CCA2}}(\lambda)$.

We are left to prove that $Pr[\Gamma_3]$ is negligible. First remark that in Γ_3, $\mathcal{A}$ has absolutely no way to predict the value r_i for any round i (as neither α_i nor β_i appears before round i). Hence, $\mathcal{A}$ can either try to guess c_i or r_i. His success probability in the second case is $\frac{1}{2}$. In the first case, he succeeds if he guesses the challenge properly (as he can obtain the response from the prover), but also if he wrongly guesses the challenge but guesses correctly the other response. The corresponding probability is $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$ for each round. As there are n such rounds, $Pr[\Gamma_3] \leq \left(\frac{3}{4}\right)^n$. □

3.4 Distance Hijacking

One of our contribution extends the distance-fraud (DF) model in the DFKO framework to take into account distance-hijacking (DH) attacks [14]. In DF attacks, the adversary is a malicious prover who aims to authenticate from a distance greater than d_{max}. In DH attacks, the adversary attempts to do the same, but he uses the unintentional help of *legitimate provers* located close to the verifier. The remote adversary may initiate the DB protocol and let the nearby prover complete the DB phase. Although this is generally true in most of the DB literature, it does not hold for DB protocols preserving anonymity. Indeed, such attacks make only sense if the verifier may differentiate between two provers. For instance, if a remote member of a group $\mathcal{X}$ of legitimate provers initiates the DB protocol and a nearby prover of the same group involuntarily completes the DB phase, the verifier would simply conclude that a member of $\mathcal{X}$ has just been authenticated. He would end up with the same conclusion if the nearby prover has completed the scheme without any intervention from the remote party. To capture DH in the DFKO framework, we consider an adversary (here a malicious prover) able to use the help of an honest prover in the verifier's proximity and having two choices.

In the DB phase, he commits to a response in advance, before the challenge of that specific round, and sends that commitment. These commitments do not refer to cryptographic commitments, with the properties of binding and hiding, but rather they indicate the prover's choice with regards to a response, which he must transmit to the verifier. In any phase, he commits to a special message Prompt, triggering the response by a close-by honest prover.

If the adversary either (1) fails to commit or prompt for one specific phase, or (2) sends a different value than committed *after* receiving the time-critical responses, he taints the phase and the session. More formally, when the adversary opens a verifier-adversary session sid, he also opens two associated dummy sessions $\mathsf{sid}_{\mathsf{Commit}}$ for committed responses and $\mathsf{sid}_{\mathsf{Prompt}}$ for the responses prompted from the prover. Technically, such an adversary is more powerful than in a typical DH attack [8], since the adversary can intercept time-critical responses that are sent by the honest prover, and replace them with his own committed responses. More precisely, the formal definition of tainted phases is as follows.

Definition 13 (Tainted Time-Critical Phase (DH))**.** A critical phase $\Pi_{\mathsf{sid}}[k, k+1] = (m_k, m_{k+1})$ of an adversary-verifier session sid,with the message m_k being received by the adversary as the k^{th} challenge from the verifier, is *tainted* if one of the following conditions holds.

- The maximal j with $\Pi_{\mathsf{sid}_{\mathsf{Commit}}}[j] = (\mathsf{sid}, k+1, m^*_{k+1})$ for $m^*_{k+1} \neq \mathsf{Prompt}$ and $\mathsf{marker}(\mathsf{sid}, k) > \mathsf{marker}(\mathsf{sid}_{\mathsf{Commit}}, j)$ satisfies $m^*_{k+1} \neq m_{k+1}$ (or no such j exists).
- The maximal j with $\Pi_{\mathsf{sid}_{\mathsf{Commit}}}[j] = (\mathsf{sid}, k+1, m^*_{k+1})$ for $m^*_{k+1} = \mathsf{Prompt}$ satisfies $m^{k+1} \neq m^{\mathsf{Prompt}}_{k+1}$, in which $m^{\mathsf{Prompt}}_{k+1}$ denotes the message m_{k+1} in $\mathsf{sid}_{\mathsf{Prompt}}$.

This definition rules out some potential actions of attackers. Once this is done, the game-based definition of DH resistance notion can be stated as follows.

Definition 14 (DH Resistance)**.** For a DB authentication scheme DB with DB threshold t_{max}, a $(t, q_{\mathsf{p}}, q_{\mathsf{v}}, q_{\mathsf{obs}})$-DH adversary $\mathcal{A}$ (with $\mathsf{id}_{\mathcal{A}}$) wins against DB

if the verifier accepts $\mathsf{id}_{\mathcal{A}}$ in one of q_{v} adversary-verifier sessions, which does not have any critical phase tainted. Thus, the DH resistance is defined as the probability $\mathbf{Adv}^{\mathsf{DH}}_{\mathsf{DB}}(\mathcal{A})$ that $\mathcal{A}$ wins this game.

The following theorem covers both DH and DF resistance. The idea is that a DF can be seen as a special case of DH in which the adversary does not use nearby provers. The proof consists in showing that the responses corresponding to an initial message e^* sent by the adversary have a negligible probability to match those of any nearby honest prover.

Theorem 15. *If the challenges follow a uniform distribution,* TREAD *is DH resistant, and*

$$\mathsf{Adv}^{DH}_{\mathsf{TREAD}}(\lambda) \leq \left(\frac{3}{4}\right)^n.$$

The proof of this theorem is given in Appendix B.1.

3.5 Privacy

We show now that the public-key instance of our protocol preserves the privacy of the provers against eavesdroppers. An adversary who intercepts information transmitted during the protocol cannot infer the identity of the prover from the information he has seen. Otherwise, he would be able to break the security of the encryption scheme.

The private construction is an instance of TREAD using $\mathsf{E} = \mathsf{PKE}$ and $\mathsf{S} = \mathsf{S\text{-}SIG}$, for a public key encryption PKE and a digital signature scheme S-SIG. In such protocols, $\mathsf{idpub}(P)$ is set to *null*. Since all the information allowing to identify the prover is encrypted, only the verifier can learn his identity. This property [18] is formalized as follows:

Definition 16 (Privacy Protection)**.** Let DB be a DB scheme. The privacy experiment $\mathsf{Exp}^{\mathsf{Priv}}_{\mathcal{A},\mathsf{DB}}(\lambda)$ for an adversary $\mathcal{A}$ on DB is defined as follows. $\mathcal{A}$ interacts with a challenger who runs the algorithm $\mathsf{DB.gen}(1^\lambda)$ to generate the set-up and sends all the public set-up parameters to $\mathcal{A}$. During the experiment, $\mathcal{A}$ has access to the following oracles:

$\overline{\mathsf{DB}}.\mathsf{Join}^c(\cdot)$: On input i, it returns the public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ of a new prover P_i using $\mathsf{DB.join}(\lambda)$.

$\overline{\mathsf{DB}}.\mathsf{Prover}(\cdot)$: On input i, it simulates a session by the prover P_i using sk_i.

$\overline{\mathsf{DB}}.\mathsf{Verifier}$ simulates a session by the verifier V using sk_v.

Then, $\mathcal{A}$ sends the pair of provers (i_0, i_1) to the challenger who picks $b \xleftarrow{\$} \{0,1\}$. Thereafter, $\mathcal{A}$ has now access to the following challenge oracle:

$\overline{\mathsf{DB}}.\mathsf{Prover}_b$ simulates a session by the prover P_{i_b} using sk_{i_b}.

Finally, $\mathcal{A}$ returns b'. If $b = b'$, the challenger returns 1 (*i.e.*, the guess of $\mathcal{A}$ is correct), while otherwise he returns 0.

We define $\mathcal{A}$'s advantage on this experiment as

$$\mathsf{Adv}^{\mathsf{Priv}}_{\mathcal{A},\mathsf{DB}}(\lambda) = \left|\mathsf{Pr}[\mathsf{Exp}^{\mathsf{Priv}}_{\mathcal{A},\mathsf{DB}}(\lambda) = 1] - \frac{1}{2}\right|$$

and the advantage on the privacy experiment as

$$\mathsf{Adv}_{\mathsf{DB}}^{\mathsf{Priv}}(\lambda) = \max_{\mathcal{A}\in\mathsf{Poly}(\lambda)}\{\mathsf{Adv}_{\mathcal{A},\mathsf{DB}}^{\mathsf{Priv}}(\lambda)\}.$$

DB is *privacy preserving* if $\mathsf{Adv}_{\mathsf{DB}}^{\mathsf{Priv}}(\lambda)$ is negligible.

Theorem 17. *If* PKE *is an* $\mathsf{IND\text{-}CCA2}$ *secure public key encryption scheme and if for any prover P values* $\mathsf{idpub}(P)$ *is set to null, then* $\mathsf{TREAD}^{\mathsf{Pub}}$ *is privacy-preserving and*

$$\mathsf{Adv}_{\mathsf{TREAD}^{\mathsf{Pub}}}^{\mathit{Priv}}(\lambda) \leq \mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{IND\text{-}CCA2}}(\lambda).$$

The proof of this theorem is given in Appendix B.2.

3.6 Prover Anonymity

We finally show that the anonymous version of our protocol preserves the *anonymity* of the provers against malicious verifiers. By anonymity, we mean that the verifier can not distinguish who is the user with whom he interacts during a session. In TREAD, the only information on a prover identity that a verifier can get during the protocol is the signatures produced by the prover. Since a secure group signature scheme is used, the protocol does not leak any information on the identity of the provers. Otherwise, a verifier would be able to break the security of the group signature scheme.

The anonymous construction is defined as an instance of TREAD using $\mathsf{E} = \mathsf{PKE}$ and $\mathsf{S} = \mathsf{G\text{-}SIG}$, for a public key encryption PKE and a group signature scheme $\mathsf{G\text{-}SIG}$. In such protocols, $\mathsf{idprv}(P)$ should only identify the corresponding group identity. Thus, the verifier should not get any information on a prover identity. This notion is formalized by the *Prover Anonymity* property defined in [12].

Definition 18 (Prover Anonymity)**.** Let DB be a DB scheme. The anonymity experiment $\mathsf{Exp}_{\mathcal{A},\mathsf{DB}}^{\mathsf{Anon}}(\lambda)$ for an adversary $\mathcal{A}$ on DB is defined as follows. $\mathcal{A}$ interacts with a challenger who runs the algorithm $\mathsf{DB.gen}(1^\lambda)$ to generate the set-up and sends all the public set-up parameters to $\mathcal{A}$. During the experiment, $\mathcal{A}$ has access to the following oracles:

$\overline{\mathsf{DB}}.\mathsf{Join}^h(\cdot)$: On input i, it creates a new prover P_i using $\mathsf{DB.join}_{\mathsf{MK}}(i, \mathsf{UL})$.

$\overline{\mathsf{DB}}.\mathsf{Join}^c(\cdot)$: On input i, it creates a corrupted prover P_i using $\mathsf{DB.join}_{\mathsf{MK}}(i, \mathsf{UL})$, returns the secret key psk_i, and adds P_i to CU.

$\overline{\mathsf{DB}}.\mathsf{Revoke}(\cdot)$: On input i, it runs $\mathsf{DB.revoke}_{\mathsf{MK}}(i, \mathsf{RL}, \mathsf{UL})$ to revoke the prover P_i.

$\overline{\mathsf{DB}}.\mathsf{Corrupt}(\cdot)$: On input i, it simulates the corruption of P_i by returning his secret key psk_i, and adds P_i to CU.

$\overline{\mathsf{DB}}.\mathsf{Prover}(\cdot)$: On input i, it simulates a session by the honest prover P_i using psk_i.

$\overline{\mathsf{DB}}.\mathsf{Verifier}$ simulates a session by the verifier V using sk_v.

First, $\mathcal{A}$ sends the pair of provers (i_0, i_1) to the challenger. If i_0 or i_1 is in CU, the challenger aborts the experiment. Otherwise, he picks $b \xleftarrow{\$} \{0,1\}$. Then, $\mathcal{A}$ loses access to $\overline{\mathsf{DB}}.\mathsf{Revoke}(\cdot)$ and $\overline{\mathsf{DB}}.\mathsf{Corrupt}(\cdot)$ on i_0 and i_1 (the oracles return $\perp$ if $\mathcal{A}$ uses these inputs). Thereafter, $\mathcal{A}$ has now access to the following challenge oracle:

$\overline{\mathsf{DB}}.\mathsf{Prover}_b$ simulates a session by the prover P_{i_b} using psk_{i_b}.

Finally, $\mathcal{A}$ returns b'. If $b = b'$, the challenger returns 1 (*i.e.*, the guess of $\mathcal{A}$ is correct), while otherwise he returns 0.

We define $\mathcal{A}$'s advantage on this experiment as

$$\mathsf{Adv}^{\mathsf{Anon}}_{\mathcal{A},\mathsf{DB}}(\lambda) = \left| \Pr[\mathsf{Exp}^{\mathsf{Anon}}_{\mathcal{A},\mathsf{DB}}(\lambda) = 1] - \frac{1}{2} \right|$$

and the advantage on the PA experiment as

$$\mathsf{Adv}^{\mathsf{Anon}}_{\mathsf{DB}}(\lambda) = \max_{\mathcal{A} \in \mathsf{Poly}(\lambda)} \{\mathsf{Adv}^{\mathsf{Anon}}_{\mathcal{A},\mathsf{DB}}(\lambda)\}.$$

DB is *prover anonymous* if $\mathsf{Adv}^{\mathsf{Anon}}_{\mathsf{DB}}(\lambda)$ is negligible.

Theorem 19. *If* $\mathsf{G\text{-}SIG}$ *is an anonymous revokable group signature scheme [21] and if for any prover P values* $\mathsf{idpub}(P)$ *and* $\mathsf{idprv}(P)$ *are either set to null or the group identity, then* TREAD^{ANO} *is prover-anonymous and*

$$\mathsf{Adv}^{\mathit{Anon}}_{\mathsf{TREAD}^{ANO}}(\lambda) \leq \mathsf{Adv}^{\mathit{Anon}}_{\mathsf{G\text{-}SIG}}(\lambda).$$

The proof of this theorem is given in Appendix B.3.

4 Conclusion

In this paper, we introduce a novel approach for provable TF resistance. More precisely, instead of relying on extraction mechanisms to make sure that a TF accomplice can impersonate the malicious prover helping him, we build a generic yet simple construction relying on replay. In this construction, an adversary helped by a malicious prover is given the ability to directly adapt the authentication information he learnt to perform a new authentication with the same probability. However, this comes at the cost of a slightly lower mafia-fraud and distance-fraud resistance.

We have also reinforce the already strong notion of SimTF and prove that if an adversary successfully authenticates with the help of a malicious prover with a non-negligible success probability, he can amplify his winning probability to an overwhelming probability. Three instances of the protocol have been presented. The first one is a symmetric-key lightweight DB protocol with no privacy, the second one is a public-key protocol private against external eavesdroppers, while the last one provides full prover anonymity with respect to malicious verifiers. Our design is generic and may be used to extend existing DB protocols.

References

[1] A. Ahmadi and R. Safavi-Naini. Privacy-preserving distance-bounding proof-of-knowledge. In *Proc. of the 16th Int. Conf. on Information and Communications Security - Revised Selected Papers*, LNCS, pages 74–88. Springer, 2014.

[2] G. Avoine, M. A. Bingöl, S. Kardaş, C. Lauradoux, and B. Martin. A formal framework for analyzing RFID distance bounding protocols. *Journal of Computer Security - Special Issue on RFID System Security*, 19(2):289–317, 2010.

[3] G. Avoine, C. Lauradoux, and B. Martin. How secret-sharing can defeat terrorist fraud. In *Proc. of WiSec*, pages 145–156. ACM, 2011.

[4] S. Bengio, G. Brassard, Y. G. Desmedt, C. Goutier, and J.-J. Quisquater. Secure implementation of identification systems. *Journal of Cryptology*, 4(3):175–183, 1991.

[5] I. Boureanu, A. Mitrokotsa, and S.Vaudenay. Secure and lightweight distance-bounding. In *Proc. of LightSec*, LNCS, pages 97–113. Springer, 2013.

[6] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. On the pseudorandom function assumption in (secure) distance-bounding protocols: PRF-ness alone does not stop the frauds! In *Proc. of the 2nd Int. Conf. on Cryptology and Information Security in Latin America*, LNCS, pages 100–120. Springer, 2012.

[7] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical & provably secure distance-bounding. Cryptology ePrint Archive, Report 2013/465, 2013.

[8] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Towards secure distance bounding. In *Proc. of Fast Software Encryption*, LNCS, pages 55–67. Springer, 2014.

[9] I. Boureanu and S. Vaudenay. Optimal proximity proofs. In *Proc. 10th Int. Conf. Inscrypt 2014*, LNCS, pages 170–190. Springer, 2014.

[10] S. Brands and D. Chaum. Distance-bounding protocols. In *Proc. of Advances in Cryptology – EUROCRYPT*, LNCS, pages 344–359. Springer, 1993.

[11] A. Brelurut, D. Gérault, and P. Lafourcade. Survey of distance bounding protocols and threats. In *Proc. of 8th Int. Symp. on Foundations and Practice of Security*, LNCS, pages 29–49. Springer, 2015.

[12] X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *Proc. of WiSec*. ACM, 2016.

[13] L. Bussard and W. Bagga. Distance-bounding proof of knowledge to avoid real-time attacks. In *Proc. of Security and Privacy in the Age of Ubiquitous Computing*, IFIP International Federation for Information Processing, pages 222–238. Springer, 2005.

[14] C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *Proc. of IEEE Symp. on Security and Privacy*, pages 113–127. IEEE, 2012.

[15] U. Dürholz, M. Fischlin, M. Kasper, and C. Onete. A formal approach to distance-bounding RFID protocols. In *Proc. of ISC*, LNCS, pages 47–62. Springer, 2011.

[16] M. Fischlin and C. Onete. Terrorism in distance bounding: Modeling terrorist fraud resistance. In *Proc. of ACNS*, LNCS, pages 414–431. Springer, 2013.

[17] S. Gambs, M. Killijian, and M. N. del Prado Cortez. Show me how you move and I will tell you who you are. *Trans. Data Privacy*, 4(2):103–126, 2011.

[18] S. Gambs, C. Onete, and J.-M. Robert. Prover anonymous and deniable distance-bounding authentication. In *Proc. of AsiaCCS*, pages 501–506. ACM, 2014.

[19] J. Hermans, R. Peeters, and C. Onete. Efficient, secure, private distance bounding without key updates. In *Proc. of WiSec*, pages 207–218. ACM, 2013.

[20] C. H. Kim, G. Avoine, F. Koeune, F. Standaert, and O. Pereira. The Swiss-Knife RFID distance bounding protocol. In *Proc. of Information Security and Cryptology*, LNCS, pages 98–115. Springer, 2008.

[21] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *Proc. of Public Key Cryptography*, LNCS, pages 463–480. Springer, 2009.

[22] V. Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004. URL: `http://eprint.iacr.org/2004/332`.

[23] S. Vaudenay. On privacy models for RFID. In *Proc. of Advances in Cryptology – Asiacrypt*, LNCS, pages 68–87. Springer, 2007.

[24] S. Vaudenay. Private and secure public-key distance bounding: Application to NFC payment. In *Proc. of Financial Cryptography*, LNCS, pages 207–216. Springer, 2015.

[25] S. Vaudenay. Sound proof of proximity of knowledge. In *Proc. of 9th Int. Conf. ProvSec*, LNCS, pages 105–126. Springer, 2015.

A Definitions

In this section, we present the classical definitions used implicitly in our formal proofs.

Definition 20. A *symmetric key encryption* scheme SKE is a triplet of algorithms $(\mathsf{SKE.gen}, \mathsf{SKE.enc}, \mathsf{SKE.dec})$ s.t.:

$\mathsf{SKE.gen}(1^\lambda)$**:** returns a secret key sk from the security parameter λ.

$\mathsf{SKE.enc_{sk}}(m)$**:** returns a ciphertext c from the message m and the key sk.

$\mathsf{SKE.dec_{sk}}(c)$**:** returns a plaintext m from the ciphertext c and the key sk.

A symmetric key encryption scheme is said *correct* if and only if $\mathsf{SKE.dec_{sk}}(\mathsf{SKE.enc_{sk}}(m)) = m$ for any message m and any secret key sk generated by $\mathsf{SKE.gen}$.

Definition 21. A *public key encryption* scheme PKE is a triplet of algorithms $(\mathsf{PKE.gen}, \mathsf{PKE.enc}, \mathsf{PKE.dec})$ s.t.:

$\mathsf{PKE.gen}(1^\lambda)$**:** returns a public/private key pair $(\mathsf{pk}, \mathsf{sk})$ from the security parameter λ.

$\mathsf{PKE.enc_{pk}}(m)$**:** returns a ciphertext c from the message m and the public key pk.

$\mathsf{PKE.dec_{sk}}(c)$**:** returns a plaintext m from the ciphertext c and the private key sk.

A public key encryption scheme is said *correct* if and only if $\mathsf{PKE.dec_{sk}}(\mathsf{PKE.enc_{pk}}(m)) = m$ for any message m and any key pair $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{PKE.gen}$.

Definition 22. Let $\mathsf{SKE} = (\mathsf{SKE.gen}, \mathsf{SKE.enc}, \mathsf{SKE.dec})$ be a symmetric key encryption scheme. SKE is said to be *indistinguishable against adaptive chosen ciphertext attack* ($\mathsf{IND\text{-}CCA2}$) when for any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, the following advantage probability $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{A},\mathsf{SKE}}(1^\lambda)$ is negligible:

$$\left| \Pr\left[\begin{array}{l} \mathsf{k} \leftarrow \mathsf{SKE.gen}(1^\lambda), b \xleftarrow{\$} \{0,1\} \\ b' \leftarrow \mathcal{A}_0^{\overline{\mathsf{SKE}}.\mathsf{enc_k}(\mathsf{LR}_b), \overline{\mathsf{SKE}}.\mathsf{dec_k}}(\lambda) \end{array} : b = b' \right] - \frac{1}{2} \right|$$

where the oracles $\overline{\mathsf{SKE}}.\mathsf{enc_k}(\mathsf{LR}_b), \overline{\mathsf{SKE}}.\mathsf{dec_k}$ are defined as:

$\overline{\mathsf{SKE}}.\mathsf{enc_k}(\mathsf{LR}_b(m_0, m_1))$**:** returns $\mathsf{SKE.enc_k}(m_b)$ on the message pair (m_0, m_1), for a random bit b.

$\overline{\mathsf{SKE}}.\mathsf{dec_k}(c)$**:** if c has been generated by $\overline{\mathsf{SKE}}.\mathsf{enc_k}(\mathsf{LR}_b)$ returns $\perp$, else returns $\mathsf{SKE.dec_k}(c)$.

Definition 23. Let $\mathsf{PKE} = (\mathsf{PKE.gen}, \mathsf{PKE.enc}, \mathsf{PKE.dec})$ be a public key encryption scheme. PKE is said to be *indistinguishable against adaptive chosen ciphertext attack* when for any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, the following advantage probability $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{A},\mathsf{PKE}}(1^\lambda)$ defined is negligible:

$$\left| \Pr\left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.gen}(1^\lambda), b \xleftarrow{\$} \{0,1\} \\ b' \leftarrow \mathcal{A}^{\overline{\mathsf{PKE}}.\mathsf{enc_{pk}}(\mathsf{LR_b})}, \overline{\mathsf{PKE}}.\mathsf{dec_{sk}}(\mathsf{pk}, \lambda) \end{array} : b = b' \right] - \frac{1}{2} \right|$$

where the oracles $\overline{\mathsf{PKE}}.\mathsf{enc_{pk}}(\mathsf{LR_b}), \overline{\mathsf{PKE}}.\mathsf{dec_{sk}}$ are defined as:

$\overline{\mathsf{PKE}}.\mathsf{enc_{pk}}(\mathsf{LR}_b(m_0, m_1)$**:** returns $\mathsf{PKE.enc_{pk}}(m_b)$ on the message pair (m_0, m_1), for a random bit b.

$\overline{\mathsf{PKE}}.\mathsf{dec_{sk}}(c)$**:** if c has been generated by $\overline{\mathsf{PKE}}.\mathsf{enc_{pk}}(\mathsf{LR_b})$ returns $\perp$, else returns $\mathsf{PKE.dec_{sk}}(c)$.

Definition 24. A *message authentication* scheme MAC is a triplet of algorithms $(\mathsf{MAC.gen}, \mathsf{MAC.sig}, \mathsf{MAC.ver})$ s.t.:

$\mathsf{MAC.gen}(1^\lambda)$**:** returns a secret key sk from the security parameter λ.

$\mathsf{MAC.sig}_{\mathsf{sk}}(m)$**:** returns a tag s from the message m and the key sk.

$\mathsf{MAC.ver}_{\mathsf{sk}}(s, m)$**:** returns a verification bit v from the tag s and the key sk.

A message authentication scheme is said *correct* if and only if $\mathsf{MAC.ver}_{\mathsf{sk}}(m, \mathsf{MAC.sig}_{\mathsf{sk}}(m)) = 1$ for any message m and any key sk generated by $\mathsf{MAC.gen}$.

Definition 25. A *digital signature* scheme SIG is a triplet of algorithms $(\mathsf{SIG.gen}, \mathsf{SIG.sig}, \mathsf{SIG.ver})$ s.t.:

$\mathsf{SIG.gen}(1^\lambda)$**:** returns a key pair $(\mathsf{sk}, \mathsf{vk})$ from the security parameter λ.

$\mathsf{SIG.sig}_{\mathsf{sk}}(m)$**:** returns a signature s from the message m and the signing key sk.

$\mathsf{SIG.ver}_{\mathsf{vk}}(s, m)$**:** returns a verification bit v from the signature s and the verification key vk.

A digital signature scheme is said *correct* if and only if $\mathsf{SIG.ver}_{\mathsf{pk}}(m, \mathsf{SIG.sig}_{\mathsf{sk}}(m)) = 1$ for any message m and any key pair $(\mathsf{sk}, \mathsf{vk})$ generated by $\mathsf{SIG.gen}$.

Definition 26. Let $\mathsf{MAC} = (\mathsf{MAC.gen}, \mathsf{MAC.sig}, \mathsf{MAC.ver})$ be a message authentication scheme. MAC is said to be *unforgeable against chosen massage attack* ($\mathsf{EUF\text{-}CMA}$) when for any adversary $\mathcal{A}$, the following advantage probability $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MAC}}(1^\lambda)$ is negligible:

$$\Pr\left[\begin{array}{l}\mathsf{k} \leftarrow \mathsf{MAC.gen}(1^\lambda)\\ (s, m) \leftarrow \mathcal{A}^{\overline{\mathsf{MAC}}.\mathsf{sign}_{\mathsf{k}}, \overline{\mathsf{MAC}}.\mathsf{ver}_{\mathsf{k}}}(\lambda)\end{array} : \mathsf{MAC.ver}_{\mathsf{k}}(s, m) = 1\right]$$

where the oracles $\overline{\mathsf{MAC}}.\mathsf{sign}_{\mathsf{k}}, \overline{\mathsf{MAC}}.\mathsf{ver}_{\mathsf{k}}$ are defined as:

$\overline{\mathsf{MAC}}.\mathsf{sign}_{\mathsf{k}}(m)$**:** returns $(m, \mathsf{MAC.sig}_{\mathsf{k}}(m))$ on input m.

$\overline{\mathsf{MAC}}.\mathsf{ver}_{\mathsf{k}}(s, m)$**:** if s has been generated by $\overline{\mathsf{MAC}}.\mathsf{sign}_{\mathsf{k}}(m)$ returns $\perp$, else returns $\mathsf{MAC.ver}_{\mathsf{k}}(m, s)$.

Definition 27. Let $\mathsf{SIG} = (\mathsf{SIG.gen}, \mathsf{SIG.sig}, \mathsf{SIG.ver})$ be a digital signature scheme. SIG is said to be *unforgeable against chosen massage attack* when for any adversary $\mathcal{A}$, the following advantage probability $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{SIG}}(1^\lambda)$ is negligible:

$$\Pr\left[\begin{array}{l}\mathsf{k} \leftarrow \mathsf{SIG.gen}(1^\lambda)\\ (s, m) \leftarrow \mathcal{A}^{\overline{\mathsf{SIG}}.\mathsf{sign}_{\mathsf{sk}}, \overline{\mathsf{SIG}}.\mathsf{ver}_{\mathsf{vk}}}(\mathsf{vk}, \lambda)\end{array} : \mathsf{SIG.ver}_{\mathsf{vk}}(s, m) = 1\right]$$

where the oracles $\overline{\mathsf{SIG}}.\mathsf{sign}_{\mathsf{sk}}, \overline{\mathsf{SIG}}.\mathsf{ver}_{\mathsf{vk}}$ are defined as:

$\overline{\mathsf{SIG}}.\mathsf{sign}_{\mathsf{sk}}(m)$**:** returns $(m, \mathsf{SIG.sig}_{\mathsf{sk}}(m))$ on message m.

$\overline{\mathsf{SIG}}.\mathsf{ver}_{\mathsf{vk}}(s, m)$**:** if s has been generated by $\overline{\mathsf{SIG}}.\mathsf{sign}_{\mathsf{sk}}(m)$ returns $\perp$, else returns $\mathsf{SIG.ver}_{\mathsf{vk}}(s, m)$.

In this case, the verification oracle is optional since the adversary knows the verification key and can simulate it.

Definition 28. A *revocable group signature scheme* $\mathsf{G\text{-}SIG}$ is defined by six algorithms:

$\mathsf{G.gen}(1^\lambda)$**:** According to a security parameter k, returns a group/master key pair $(\mathsf{gpk}, \mathsf{msk})$ and two empty lists: the user list UL and the revoked user list RL.

$\mathsf{G.join}_{\mathsf{msk}}(i, \mathsf{gpk}, \mathsf{UL})$**:** is a protocol between a user U_is (using gpk) and a group manager GM (using msk and gpk). U_i interacts with GM to obtain a group signing key ssk_i. Finally, GM outputs a value reg_i and adds U_i to UL.

$\mathsf{G.rev}_{\mathsf{msk}}(i, \mathsf{UL}, \mathsf{RL}, \mathsf{gpk})$**:** computes revocation logs rev_i for U_i, using reg_i, gpk and msk, and moves U_i from UL to RL.

$\mathsf{G.sig}_{\mathsf{ssk}_i}(m)$**:** returns a group signature σ for the message m.

$\mathsf{G.ver}_{\mathsf{gpk}}(\sigma, m, \mathsf{RL})$**:** returns 1 if σ is valid for the message m and the signing key ssk_i of a non-revoked user, and 0 otherwise.

$\mathsf{G.ope}_{\mathsf{msk}}(\sigma, m, \mathsf{UL}, \mathsf{gpk})$**:** outputs the identity of U_i who generated the signature σ.

Definition 29. Let $\mathsf{G\text{-}SIG}$ be a group signature scheme. The anonymity experiment $\mathsf{Exp}^{\mathsf{Anon}}_{\mathcal{A},\mathsf{G\text{-}SIG}}(\lambda)$ for the adversary $\mathcal{A}$ on $\mathsf{G\text{-}SIG}$ is defined as follows. $\mathcal{A}$ interact with a challenger who creates $(\mathsf{UL}, \mathsf{RL}, \mathsf{msk}, \mathsf{gpk})$ using $\mathsf{G.gen}(1^\lambda)$, gives gpk to $\mathcal{A}$, and sets the lists CU and Σ. During this phase $\mathcal{A}$ has access to G-oracles:

$\overline{G}.\mathsf{Join}^h(\cdot)$**:** On input i, creates P_i with $\mathsf{G.join}_{\mathsf{msk}}(i, \mathsf{gpk}, \mathsf{UL})$.

$\overline{G}.\mathsf{Join}^c(\cdot)$**:** On input i, creates P_i with $\mathsf{G.join}_{\mathsf{msk}}(i, \mathsf{gpk}, \mathsf{UL})$ with $\mathcal{A}$, and adds him to CU.

$\overline{G}.\mathsf{Revoke}(\cdot)$**:** On input i, revokes P_i with $\mathsf{G.rev}_{\mathsf{msk}}(i, \mathsf{RL}, \mathsf{UL}, \mathsf{gpk})$.

$\overline{G}.\mathsf{Corrupt}(\cdot)$**:** On input i, returns the secret information of an existing P_i. If $P_i \in \mathsf{UL}$, it sends ssk_i to $\mathcal{A}$ and adds P_i to CU.

$\overline{G}.\mathsf{Sign}(\cdot, \cdot)$**:** On input i, returns a signature σ_p on behalf of P_i, using $\mathsf{G.sig}_{\mathsf{ssk}_i}(m)$, and adds the pair (m, σ_p) to Σ.

$\overline{G}.\mathsf{Open}(\cdot, \cdot)$**:** On input i, opens a signature σ on m and returns P_i to $\mathcal{A}$, using the algorithm $\mathsf{G.ope}_{\mathsf{msk}}(\sigma, m, \mathsf{UL}, \mathsf{gpk})$. This oracle rejects all signatures produced by $\overline{G}.\mathsf{Sign}_b(\cdot, \cdot)$.

$\mathcal{A}$ outputs (i_0, i_1) to the challenger. If i_0 and $i_1 \in \mathsf{CU}$, the challenger stops. Otherwise, he picks $b \xleftarrow{\$} \{0, 1\}$ and sends it to $\mathcal{A}$. $\mathcal{A}$ cannot henceforth use $\overline{G}.\mathsf{Corrupt}(\cdot)$ and $\overline{G}.\mathsf{Revoke}(\cdot)$ on i_0 or i_1. Moreover, $\mathcal{A}$ has access to the G-oracle:

$\overline{G}.\mathsf{Sign}_b(\cdot, \cdot)$**:** On input m, returns $\mathsf{G.sig}_{\mathsf{ssk}_{i_b}}(m)$.

Finally, $\mathcal{A}$ outputs b'. If $(b = b')$ the challenger outputs 1, else he outputs 0.

Define $\mathsf{Adv}^{\mathsf{Anon}}_{\mathsf{G\text{-}SIG}}(\lambda)$ as in Definition 18. A group signature $\mathsf{G\text{-}SIG}$ is anonymous when $\mathsf{Adv}^{\mathsf{Anon}}_{\mathsf{G\text{-}SIG}}(\lambda)$ is negligible.

B Security Proofs

B.1 Distance-Hijacking resistance

Proof of Thm 15. First note that if $\mathcal{A}$ uses Prompt for the initial message, *i.e.*, he lets an honest prover send it and then his authentication automatically fails, as $\mathsf{idpub}(P)$ and/or $\mathsf{idprv}(P)$ do not correspond to the identity of $\mathcal{A}$.

Hence, consider the case in which $\mathcal{A}$ initiated the protocol with a message e^* (associated with α^*, β^*). Let e (and $\alpha||\beta$) denote the values picked by the nearby honest prover P. For each challenge c_i, either $\mathcal{A}$ uses Prompt to let P respond or he uses Commit to respond himself before receiving c_i.

- If he uses Prompt, his response is valid with probability $\frac{1}{2}$. This is the probability to have $\alpha_i = \alpha_i^*$ (or $\beta_i = \beta_i^*$).
- If he uses Commit, either $\alpha_i^* = \beta_i^* \oplus m_i$, and he can commit to a correct response with probability 1, or $\alpha_i^* \neq \beta_i^* \oplus m_i$, and he must guess the challenge to commit to the correct response. Since m is uniformly distributed and unknown to $\mathcal{A}$ at the time when he picks $\alpha||\beta$, $Pr[\alpha_i^* = \beta_i^* \oplus m_i] = \frac{1}{2}$. Hence, the probability to commit to the valid response is $Pr[\alpha_i^* = \beta_i^* \oplus m_i] \cdot 1 + Pr[\alpha_i^* \neq \beta_i^* \oplus m_i] \cdot \frac{1}{2} = \frac{3}{4}$.

It follows that the best strategy for $\mathcal{A}$ is to respond by himself, as in a classical DF, using Commit. For n challenges, his advantage $\mathbf{Adv}_{\mathsf{DB}}^{\mathsf{DH}}(\mathcal{A})$ is at most $\left(\frac{3}{4}\right)^n$, which is negligible. □

B.2 Privacy preserving property

Proof of Thm 17. Assume that there is a polynomial-time adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{Priv}}_{\mathcal{A},\mathsf{TREAD}^{\mathsf{Pub}}}(\lambda)$ is non-negligible. We show how to build an adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{B},\mathsf{PKE}}(\lambda)$ is also non-negligible.

Initially, the challenger sends a key pk_v to $\mathcal{B}$. Then, $\mathcal{B}$ runs $\mathsf{DB.gen}(1^\lambda)$ to generates the setup parameters of the scheme and sends to $\mathcal{A}$ the public set-ups and pk_v. Having access to PKE-oracles from his challenger, $\mathcal{B}$ can simulate the DB-oracles for $\mathcal{A}$ as follows.

$\overline{\mathsf{DB}}.\mathsf{Join}^c(\cdot)$: On input i, $\mathcal{B}$ returns the public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ of a new prover P_i using $\overline{\mathsf{DB}}.\mathsf{join}(\lambda)$.

$\overline{\mathsf{DB}}.\mathsf{Prover}(\cdot)$: $\mathcal{B}$ simulates P_i for $\mathcal{A}$ using sk_i and pk_v.

$\overline{\mathsf{DB}}.\mathsf{Verifier}$: $\mathcal{B}$ simulates V for $\mathcal{A}$ as follows:

Initialization phase $\mathcal{B}$ receives e from $\mathcal{A}$ and computes $(\alpha||\beta||\mathsf{idprv}(P_i)||\sigma_p) = \mathsf{PKE.dec}_{\mathsf{sk}_v}(e)$ using his oracle. If $\mathsf{S.ver}_{\mathsf{vk}_i}(\sigma_p, \alpha||\beta||\mathsf{idprv}(P_i)) = 0$ (vk_i is the verification key of P_i), $\mathcal{B}$ returns $\perp$ and aborts this simulation. Finally, he picks $m \xleftarrow{\$} \{0,1\}^n$ and returns it.

Distance-bounding phase $\mathcal{B}$ picks $c_j \in \{0,1\}$, sends it to $\mathcal{A}$ and waits for the response r_j. He repeats this protocol for all j in $\{0, \dots, n\}$.

Verification phase If, for all j in $\{0, \dots, n\}$, $r_j = \alpha_j$ when $c_j = 0$ and $r_j = \beta_j \oplus m_j$ when $c_j = 1$ then $\mathcal{B}$ returns 1 to $\mathcal{A}$, otherwise he returns 0.

$\mathcal{A}$ sends (i_0, i_1) to $\mathcal{B}$. Then, $\mathcal{B}$ instantiates a counter $l := 0$ and simulates the challenge oracle $\overline{\mathsf{DB}}.\mathsf{Prover}_b$ as follows.

Initialization phase $\mathcal{B}$ picks $\alpha||\beta \xleftarrow{\$} \{0,1\}^{2\cdot n}$ and computes the signatures $\sigma_p^0 = \mathsf{S.sig}_{\mathsf{sk}_{i_0}}(\alpha||\beta||\mathsf{idprv}(P_{i_0}))$ and $\sigma_p^1 = \mathsf{S.sig}_{\mathsf{sk}_{i_1}}(\alpha||\beta||\mathsf{idprv}(P_{i_1}))$. He sends the messages $m_0 = (\alpha||\beta||\mathsf{idprv}(P_{i_0})||\sigma_p^0)$ and $m_1 = (\alpha||\beta||\mathsf{idprv}(P_{i_1})||\sigma_p^1)$ to his challenge encryption oracle $\mathsf{SKE.enc_k}(\mathsf{LR}_b(\cdot,\cdot))$ in order to obtain e. Afterwards, he sets $\text{List}_l = (\alpha, \beta, e)$ and increments the counter l by one. Finally, he returns e and receives m.

Distance-bounding phase $\mathcal{B}$ uses α, β and m to correctly respond to the challenges c_i sent by $\mathcal{A}$.

Verification phase $\mathcal{B}$ receives Out_V from $\mathcal{A}$.

After the challenge phase, the oracles $\overline{\mathsf{DB}}.\mathsf{Join}^c(\cdot)$ and $\overline{\mathsf{DB}}.\mathsf{Prover}(\cdot)$ are simulated by $\mathcal{B}$ as in the first phase of the experiment. $\overline{\mathsf{DB}}.\mathsf{Verifier}$ is simulated as follows:

Initialization phase $\mathcal{B}$ receives e from $\mathcal{A}$. If there is no $0 \le d \le l$ such that $\text{List}_d = (\alpha, \beta, e)$, $\mathcal{B}$ simulates the oracle as in the first phase. Otherwise, $\mathcal{B}$ picks $m \xleftarrow{\$} \{0,1\}^n$ and returns it.

Distance-bounding phase $\mathcal{B}$ picks $c_j \in \{0,1\}$, sends it to $\mathcal{A}$ and waits the response r_j. It repeats this protocol for all j in $\{0,\ldots,n\}$.

Verification phase Using $\text{List}_d = (\alpha, \beta, e)$, if for all $j \in \{0,\ldots,n\}$, $r_j = \alpha_j$ when $c_j = 0$ and $r_j = \beta_j \oplus m_j$ when $c_j = 1$, $\mathcal{B}$ returns 1 to $\mathcal{A}$. Otherwise, he simply returns 0.

Finally, $\mathcal{A}$ returns b' to $\mathcal{B}$ who returns it to the challenger.

The experiment is perfectly simulated for $\mathcal{A}$, and in consequence $\mathcal{B}$ wins his experiment with the same probability that $\mathcal{A}$ wins his. Thus, $\mathsf{Adv}^{\mathsf{Priv}}_{\mathcal{A},\mathsf{TREAD}^{\mathsf{Pub}}}(\lambda) = \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{B},\mathsf{PKE}}(\lambda)$, contradicting the assumption on PKE. $\square$

B.3 Anonymity preserving property

Proof of Thm 19. Assume that there is a polynomial-time adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{Anon}}_{\mathcal{A},\mathsf{TREAD}^{ANO}}(\lambda)$ is non-negligible. We show how to construct an adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\mathsf{Anon}}_{\mathcal{B},\mathsf{G\text{-}SIG}}(\lambda)$ is also non-negligible.

Initially, the challenger sends a key gpk and a revoked list RL to $\mathcal{B}$. Then, $\mathcal{B}$ generates a public/private key pair $\mathsf{pk}_v, \mathsf{sk}_v$ for the verifier using $\mathsf{PKE.gen}(1^\lambda)$. Thus, $\mathcal{B}$ sends $(\mathsf{pk}_v, \mathsf{gpk}, \mathsf{RL})$ to $\mathcal{A}$ and creates the empty list CU. Having access to $\mathsf{G\text{-}SIG}$-oracles from his challenger, $\mathcal{B}$ can simulate the DB-oracles for $\mathcal{A}$ as follows:

$\overline{\mathsf{DB}}.\mathsf{Join}^h(\cdot)$: On input i, creates P_i with $\overline{G}.\mathsf{Join}^h(\cdot)$, and adds P_i to UL.

$\overline{\mathsf{DB}}.\mathsf{Join}^c(\cdot)$: On input i, creates a corrupted P_i with $\overline{G}.\mathsf{Join}^c(\cdot)$, adds P_i to UL and CU, and returns ssk_i.

$\overline{\mathsf{DB}}.\mathsf{Revoke}(\cdot)$: On input i, revokes P_i with $\overline{G}.\mathsf{Revoke}(\cdot)$, which updates RL and returns it.

$\overline{\mathsf{DB}}.\mathsf{Corrupt}(\cdot)$: On input i, corrupts P_i with $\overline{G}.\mathsf{Corrupt}(\cdot)$ and gets ssk_i. $\mathcal{B}$ adds P_i to CU, and returns ssk_i.

$\overline{\mathsf{DB}}.\mathsf{Prover}(\cdot)$: $\mathcal{B}$ simulates P_i for $\mathcal{A}$ as follows.

Initialization phase $\mathcal{B}$ picks $\alpha||\beta \xleftarrow{\$} \{0,1\}^{2\cdot n}$ and uses his oracle $\overline{G}.\mathsf{Sign}(\cdot,\cdot)$ to get the signature $\sigma_p = \mathsf{G.sig}_{\mathsf{ssk}_i}(\alpha||\beta)$. He computes $e = \mathsf{PKE.enc}_{\mathsf{pk}_v}(\alpha||\beta||\sigma_p)$ and returns it. He then gets m.

Distance-bounding phase $\mathcal{B}$ uses α, β and m to correctly respond to the challenges c_i sent by $\mathcal{A}$.

Verification phase $\mathcal{B}$ receives Out_V from $\mathcal{A}$.

$\overline{\mathsf{DB}}.\mathsf{Verifier}$: $\mathcal{B}$ simulates V for $\mathcal{A}$ as follows:

Initialization phase $\mathcal{B}$ receives e from $\mathcal{A}$ and computes $(\alpha||\beta||\sigma_p) = \mathsf{PKE.dec}_{\mathsf{sk}_v}(e)$. If the verification $\mathsf{G.ver}_{\mathsf{gpk}}(\sigma_p, \alpha||\beta, \mathsf{RL}) = 0$ then $\mathcal{B}$ returns $\perp$ and aborts this oracle simulation. Finally, he picks $m \xleftarrow{\$} \{0,1\}^n$ and returns it.

Distance-bounding phase $\mathcal{B}$ picks $c_j \in \{0,1\}$, sends it to $\mathcal{A}$ and waits the response r_j. He repeats this protocol for all j in $\{0,\ldots,n\}$.

Verification phase If, for all j in $\{0,\ldots,n\}$, $r_j = \alpha_j$ when $c_j = 0$ and $r_j = \beta_j \oplus m_j$ when $c_j = 1$ then $\mathcal{B}$ returns 1 to $\mathcal{A}$, else he returns 0.

$\mathcal{A}$ sends (i_0, i_1) to $\mathcal{B}$. If i_0 or $i_1 \in \mathsf{CU}$, $\mathcal{B}$ aborts the experiment. Otherwise, $\mathcal{B}$ sends (i_0, i_1) to the challenger. Then, $\mathcal{B}$ returns $\perp$ when he simulates $\mathsf{Corrupt}(\cdot)$ and $\mathsf{Revoke}(\cdot)$ on inputs i_0 and i_1 . Afterward, $\mathcal{B}$ simulates the challenge oracle $\overline{\mathsf{DB}}.\mathsf{Prover}_b$ for P_{i_b} as follows:

Initialization phase $\mathcal{B}$ picks $\alpha||\beta \xleftarrow{\$} \{0,1\}^{2\cdot n}$, uses his oracle $\overline{G}.\mathsf{Sign}_b(\cdot,\cdot)$ to get the signature $\sigma_p = \mathsf{G.sig}_{\mathsf{ssk}_i}(\alpha||\beta)$, and returns $e = \mathsf{PKE.enc}_{\mathsf{pk}_v}(\alpha||\beta||\sigma_p)$. He then gets m.

Distance-bounding phase $\mathcal{B}$ uses α, β and m to correctly respond to the challenges c_i sent by $\mathcal{A}$.

Verification phase $\mathcal{B}$ receives Out_V from $\mathcal{A}$.

Finally, $\mathcal{A}$ returns b' to $\mathcal{B}$ who returns it to the challenger.

The experiment is perfectly simulated for $\mathcal{A}$ and in consequence, $\mathcal{B}$ wins his experiment with the same probability that $\mathcal{A}$ wins his. Thus, $\mathsf{Adv}^{\mathsf{Anon}}_{\mathcal{B},\mathsf{G\text{-}SIG}}(\lambda) = \mathsf{Adv}^{\mathsf{Anon}}_{\mathcal{A},\mathsf{TREAD}^{\mathsf{Ano}}}(\lambda)$, contradicting the assumption on $\mathsf{G\text{-}SIG}$. □