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

Gildas Avoine
INSA/IRISA Rennes
gildas.avoine@irisa.fr

Xavier Bultel
LIMOS, U. Clermont Auvergne
xavier.bultel@uca.fr

Sébastien Gambs
UQAM, Montréal
gambs.sebastien@uqam.ca

David Gérault
LIMOS, U. Clermont Auvergne
david.gerault@uca.fr

Pascal Lafourcade
LIMOS, U. Clermont Auvergne
pascal.lafourcade@uca.fr

Cristina Onete
INSA/IRISA Rennes
cristina.onete@gmail.com

Jean-Marc Robert
ETS, Montréal
jean-marc.robert@etsmtl.ca

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 complex. The classical solutions 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, if he is able to authenticate.

We propose a novel approach to obtain *provable* terrorist-fraud resistant protocols that does not rely on an accomplice being able to extract any long-term key. Instead, we simply assume that he can replay the information received from the prover. Thus, rational provers should refuse to cooperate with third parties if they can impersonate them freely afterwards. We introduce a generic construction for provably secure distance-bounding protocols, and give three instances of this construction: (1) an efficient symmetric-key protocol, (2) a public-key protocol protecting the identities of provers against external eavesdroppers, and finally (3) a fully anonymous protocol protecting the identities of provers even against malicious verifiers that try to profile them.

*This research was supported by the FEDER program of 2014-2020, the region council of Auvergne, the Digital Trust Chair of the University of Auvergne, NSERC Discovery Grants and the Cost Action IC1403 in the EU Framework Programme Horizon 2020.

AsiaCCS '17, April 4–6, 2017, Abu Dhabi, United Arab Emirates.

DOI: http://dx.doi.org/10.1145/3052973.3053000

1. INTRODUCTION

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

A crucial concern for contactless communications are *relay* attacks, in which an adversary forwards the communications between a prover and a verifier to authenticate [15, 4]. These attacks cannot be prevented by cryptographic tools and 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 (or *rounds*) 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 transportation system in which users authenticate to access buses or subway stations through their NFC-enabled smartphones. The transportation company will deploy controls to prevent misuses of its system but a legitimate user might be tempted to help a friend to use his credentials illegally for a single trip, which is known as a *terrorist fraud (TF)*. Nevertheless, this user might not accept that his friend uses them afterwards as the original user may get caught and be accountable. Note that this attack targets the transportation company. Another threat against DB protocols, known as a *mafia fraud (MF)*, is a fraudster using the presence of a legitimate user to authenticate. This attack targets the transportation company as well as the end user as he may have to pay for this extra fare. Both types of attacks are typical relay attacks against contactless authentication protocols. Another crucial aspect for such a system is the protection of user privacy. Indeed, most users would not accept that their whereabouts can be tracked down by other users or by the transportation company due the wealth of personal information that can be inferred from such data.

Another simple scenario could be the access to a restricted building. In this case, third parties may want to enter (MF attacks), or legitimate workers may want to help friends to access the building (TF attacks). However, the verifier is not directly a threat against the privacy of the workers.

In this paper, we propose a new approach for developing provably secure DB protocols resisting to all classical threats against such protocols. Its novelty relies on the fact that a prover can 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 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). In particular, 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, which expose the privacy and the anonymity of provers.

Note that the TF-resistance property is a concept that is difficult to formalize and numerous attempts have been made [16, 7, 17, 26]. Far from claiming that our approach is the only viable alternative to achieving TF-resistance, it expands the fundamental understanding of the 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. Our main contribution is to propose a new approach for provable TF resistance in which the prover selects unilaterally the binary responses used during the time-critical challenge-response phases. If a malicious prover gives this information to his accomplice, the accomplice can then *adapt and replay* successfully the information received during a new session. Since a *rational* prover is not willing to allow an accomplice to impersonate him at will, he will not attempt any TF attack in the first place. As a consequence, we obtain an intuitive TF resistance proof without relying on any artificial extraction mechanism. Surprisingly, this idea has not been considered in the literature before. As shown in this paper, it can be used to design protocols achieving the simulation-based TF resistance notion [16], 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. Intuitively, relaxing the freshness of the information and allowing the replay of some authenticated data may introduce a way for an attacker to impersonate a legitimate prover. In our context, such an attack is typically referred to as a MF attack. Fortunately, this attack would be successful only if the attacker could be able to guess half of his missing responses to the verifier's challenges. This explains why the MF-resistance of our solution is not as strong as the best available solutions.

Generic construction. Our second contribution is the protocol 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 many ways including a more efficient symmetric-key protocol, a public-key protocol protecting the privacy of provers in the presence of external eavesdroppers, and a protocol based on group signatures protecting the anonymity of provers even against *malicious* verifiers trying to profile provers. The latter one can be used for instance in the public transportation scenario, whilst the first two are more adapted to the scenario of the restricted-access building.

Extension of DFKO. The final contribution is to extend the DFKO framework [16] to deal with *distance-hijacking (DH)* attacks [14], in which a malicious prover tries to fool a verifier, by taking advantage of nearby honest provers. This provides a framework to deal with all the potential attacks against DB protocols and the security of TREAD is proven in this extended framework.

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

Table 1: Summary of existing solutions. TF denotes the terrorist-fraud resistance. The probabilities of successful mafia-fraud (MF) and distance-fraud (DF) 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, while R indicates if a user can be revoked easily.

Protocol	TF	MF	DF	P	A	R
Not formally proven						
Swiss Knife [21]	✓	$\left(\frac{1}{2}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✗	✓
Provably secure						
SKI [6]	✓	$\left(\frac{3}{4}\right)^n$	$\left(\frac{2}{3}\right)^n$	✗	✗	✓
FO [17]	✓	$\left(\frac{3}{4}\right)^n$	$\left(\frac{3}{4}\right)^n$	✗	✗	✓
Provably-secure and privacy-preserving						
privDB [25]	✗	$\left(\frac{1}{2}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✗	✓
GOR [19]	✗	$\left(\frac{1}{2}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✓	✓
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	✓	$\left(\frac{3}{4}\right)^n$	$\left(\frac{3}{4}\right)^n$	✓	✓	✓

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 [19], 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 imposes to double the number of time-critical rounds to achieve the same security level). In fact, it has the best security properties of any fully anonymous protocol *not relying on* any artificial and inefficient extraction mechanism. It almost matches the TF, MF and distance-fraud (DF) resistance of the best proven solutions [6, 17] 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, the latter has not been formally proven. In fact, a minor attack has been presented against it [5].

Related Work. Since the introduction of DB protocols in 1993 by Brands 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 helping an accomplice to authenticate (*Terrorist Fraud*), which is the most difficult attack to counter.

The classical countermeasures against TF rely on the assumption that a malicious prover does not trust his accomplice enough 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 [17] 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. In SPADE, 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 these properties. An important drawback of this approach is that it does not easily support 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, another drawback of this solution is that a *malicious* verifier is able to replay the received information and impersonate a given prover, which constitutes a major threat against the latter.

In their SKI protocols [7], Boureanu, Mitrokotsa and Vaudenay used a *leakage scheme* allowing an adversary to retrieve the long-term secret key used several times by a prover. This 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, which consists in sharing the prover's long-term secret using a (n, k)-threshold cryptographic scheme. Upon reception of a challenge, a prover 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. Furthermore, the scheme neither considers distance fraud, nor addresses the issue of privacy. Finally, Vaudenay [26] relies 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. All these solutions depend on computationally-expensive primitives. Overall, TREAD has a simpler analysis than any of these protocols with the same security properties. In addition, as these solutions rely explicitly on long-term secret keys, they present serious challenges for developing strong privacy properties.

While a lot of effort has gone in proposing 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 [18], 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 have been introduced recently [20]. Gambs, Onete and Robert [19] extended this notion to deal with *honest-but-curious* and *malicious* verifiers, which try to profile legitimate provers by linking their authentication sessions. They proposed an extension of the HPO protocol [20] in which the provers are managed as a group. Though they addressed the classical MF, DF and IF, they did not consider TF. Recently, Vaudenay [25] proposed a generic solution to add privacy to DB protocols with respect to external eavesdroppers, which relies on an authenticated key-exchange build on top of a one-time secure DB protocol. Unfortunately, it does not provide neither TF -esistance nor anonymity against honest-but-curious or malicious verifiers.

Finally, Ahmadi and Safavi-Naini [1] gave a TF-resistant protocol PDB, which protects the anonymity of the prover by fixing weaknesses of the DBPK-log protocol [13]. 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 allow to revoke the credential of a prover without adding too much complexity and damaging the robustness of the scheme. Furthermore, since the authentication is 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 can be done by separate authorities).

Overall, more than forty DB protocols have appeared since 1993. Unfortunately, based on a recent survey [11] only few of them have not been broken yet.

Outline. In the next section, we describe our generic construction providing TF-resistance and three of its possible instantiations. Afterwards, in Section 3, we introduce the security models and prove the main security properties of our solutions 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 fraction 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 a symmetric-key or public-key scheme) and an EUF-CMA-secure signature scheme S. The given 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 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 both by the verifier and the prover. Unfortunately, this is not sufficient to prevent provers from influencing the values $\alpha||\beta$ [5, 11]. Indeed as mentioned earlier, there is a potential attack against the Swiss-Knife protocol [21] based on the use of a weak PRF [5].

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 a fundamental shift compared to previous approaches existing in the distance-bounding literature. 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* successfully this information in future sessions. Thus, rational provers will refuse to cooperate in the first place. The difficulty is to ensure that *only* TF accomplices benefit from this strategy, and not regular *Man-in-the-Middle* (MiM) adversaries.

In our construction, anyone knowing proper responses corresponding to a given initial message e 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 bits of the strings α and β corresponding to an authenticated commitment e. He may then replay e and the responses known to him. However, this adversary must still guess on average $\frac{n}{2}$ values.

The counter-intuitive second design choice has important 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 a probability very close to 1.

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.

DEFINITION 1 (TREAD). *The construction is composed of five algorithms and parametrized by an* IND-CCA2*-secure encryption scheme* E*, an* EUF-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 rounds* n*, which is a function of* λ*.*

$\mathsf{DB.prover}(\mathsf{ek}, \mathsf{sk})$ *is the algorithm executed by the prover in Figure 2. The prover draws a random value* $\alpha||\beta$ *from the uniform distribution on* $\{0,1\}^{2n}$*. Then, he computes a signature* σ_p *on it with* $\mathsf{S.sig_{sk}}(\alpha||\beta||\mathsf{idprv}(P))$*. Afterwards, he gets* $e = \mathsf{E.enc_{ek}}(\alpha||\beta||\mathsf{idprv}(P)||\sigma_p)$ *and sends* $e||\mathsf{idpub}(P)$*. Finally, during the* n *time-critical rounds, 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 (*$\mathsf{idpub}(P)$*) or 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 then expects to receive an initial message* e *and deciphers it as* $(\alpha||\beta||\mathsf{idprv}(P)||\sigma_p) = \mathsf{E.dec_{dk}}(e)$*. If* σ_p *is invalid (*i.e.*,* $\mathsf{S.ver_{vk}}(\sigma_p, \alpha||\beta||\mathsf{idprv}(P)) = 0$*), the verifier aborts. Otherwise, he returns a random* n*-bit string* m*.*

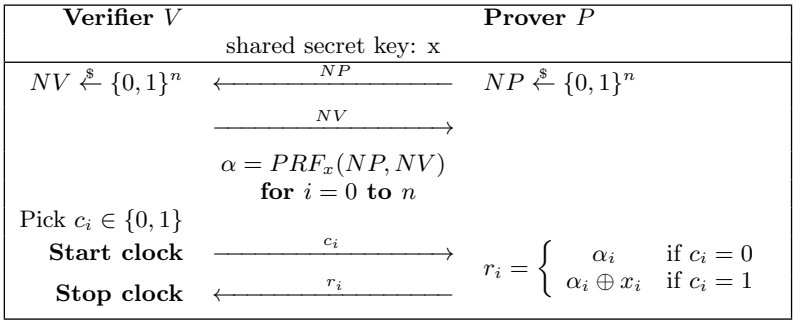

Figure 1: The classical countermeasure against terrorist fraud: if the prover gives both possible responses, such as for instance α_i and $\alpha_i \oplus x_i$ to his accomplice for a given c_i, he leaks one bit of his long-term authentication secret x. Note that PRF is a pseudorandom function keyed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

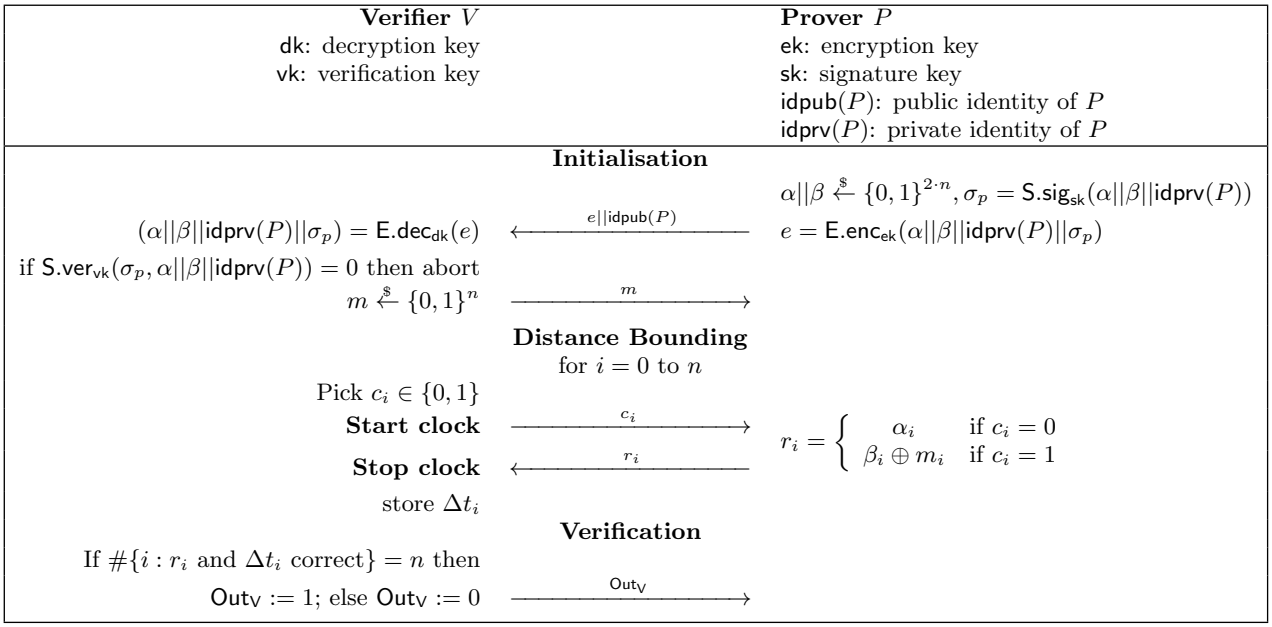

Figure 2: Our generic and provably secure DB construction TREAD built from an IND-CCA2-secure encryption scheme E and an EUF-CMA-secure signature scheme S. The symbol || denotes the concatenation operation.

Afterwards, during the n time-critical rounds, he generates random bits c_i with a uniform distribution, starts his clock, sends c_i, gets back r_i, stops his clock and stores the corresponding elapsed 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 this holds, he sends an accepting bit $\mathsf{Out_V} = 1$, otherwise he sends $\mathsf{Out_V} = 0$.

DB.join(ID, UL) *is the algorithm to register a new prover with identifier* ID *in the list* UL. *It returns the keys* (ek, dk) *for* E *and* (sk, 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.

DB.revoke(ID, UL, RL) *is the algorithm to revoke a prover with identifier* ID *and move him to the revoke list* RL.

These last two algorithms depend on the instance of the protocol and are described in the following section. TREAD adopts the *sign-then-encrypt* paradigm instead of the more usual *encrypt-then-sign*. If the latter were used, an eavesdropper would be able to infer the identity of any prover, by 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

Three instances of our construction are presented here.

Efficient symmetric-key scheme. Computational efficiency is critical for the design of DB protocols as they are usually run in resource-limited devices.

Our most efficient construction is based on an IND-CCA2 symmetric-key encryption scheme SKE and an EUF-CMA message authentication code scheme MAC. The public identity idpub(P) is the identity of the prover and the private identity idprv(P) is set to *null*. Since SKE and MAC are symmetric, we have ek = dk and sk = vk. Thus, the prover and the verifier have the same symmetric key k = (ek, 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. It is also responsible to distribute securely these lists to the legitimate verifiers.

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

TREAD can be instantiated with an IND-CCA2 public-key encryption PKE and an EUF-CMA digital signature scheme S-SIG, in which 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 the public and the private keys of the verifier, and sk and vk are the (private) signature and the (public) verification keys of the prover.

With such a protocol, two sessions by the same user are unlinkable for an external eavesdropper as the only information sent about the prover's identity is encrypted with the verifier's public key. 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 is in charge of adding provers in the public list UL or in the revokation public list RL.

Prover anonymity and group signature. Finally, TREAD can provide full prover-anonymity with respect to a malicious verifier. As profiling users is now a common threat, it is crucial to develop privacy-preserving DB protocols.

Both the prover anonymity and the revocability properties can be achieved by instantiating TREAD with an IND-CCA2 public-key encryption scheme PKE and a revocable group signature scheme G-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. Many 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 the public and private keys of the verifier, sk is the prover's signing key and vk is the group verification key.

Group signature schemes allow a user to anonymously sign on behalf of his group. 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 scheme G-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], a second one by Dürholz, Fischlin, Kasper and Onete [16] (DFKO) and a third one by Boureanu, Mitrokotsa and Vaudenay [7]. In this paper, we use the DFKO model and its strong TF-resistance notion (SimTF). The DFKO model is also extended to address DH attacks [14]. Finally, we use the privacy and anonymity models derived from the work of Gambs, Onete and Robert [19], which are compatible with the proposed extension of the DFKO model.

Distance-bounding protocols. DB protocols are interactive protocols running between two participants. The objective of the *prover* P is to convince the *verifier* V that he is legitimate and located at a distance at most d_{max} from him.

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 response r_i. These back-and-forth rounds are referred to as *time-critical* rounds. 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. Non-critical phases are simply called *slow phases*. After measuring the elapsed time at the end of each time-critical round, the verifier compares this value to a threshold t_{max} associated with the maximal allowed distance d_{max}. If 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, if it is possible, he may try to lift the anonymity of a prover or 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 the DFKO model, an adversary can interact with provers and verifiers in three kinds of sessions. Each session is associated with a unique identifier sid.

- *Prover-verifier* sessions to observe an honest execution of the protocol between a prover and a verifier.
- *Prover-adversary* sessions to interact with a honest prover as a verifier.
- *Adversary-verifier* sessions to interact with a legitimate verifier as a prover.

The adversaries are defined in terms of their computational resources (*i.e.*, time) 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 $\mathsf{marker} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, such that $\mathsf{marker}(\cdot, \cdot)$ is strictly increasing. It can be used to define *tainted* time-critical rounds. This notion is used to rule out some illegitimate actions of attackers, 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 [24] and later re-used to define prover-anonymity [20], 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 constructions: (1) TF resistance, (2) MF resistance, (3) DH resistance (implying DF resistance), (4) prover privacy and finally (5) prover anonymity. In our context, the last property is the strongest one as it protects the privacy of the provers against the verifiers themselves. The proofs of the TF resistance and prover anonymity properties are among the main contributions of this paper.

The *slow-phase* impersonation threat is discarded in our analysis [16]. This property refers exclusively to how much impersonation resistance the slow phases of the protocol adds to the impersonation protections provided in the time-critical phases of the MF countermeasures. The notion of slow-phase impersonation security was introduced especially for resource-limited provers, which cannot handle a high number of time-critical rounds. However, such a restriction is no longer a problem for contactless devices, which have become faster and more efficient in their interactions. As a consequence, we choose to rely only time-critical phases to achieve impersonation resistance, rather than adding that property in slow phases of our 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.

Thus, the challenger may have to simulate these oracles:

$\mathsf{Verifier}(\cdot)$ *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 arrival of a corrupted prover U_i by running* $\mathsf{DB.join}(i, \mathsf{UL})$ *and returning the secret keys of this prover.*

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

3.2 Terrorist-Fraud Resistance

Dürholz, Fischlin, Kasper and Onete defined the notion of simulation-based TF-resistance SimTF [16]. In this model, a far-away malicious prover P wants to use an accomplice $\mathcal{A}$ close to the verifier to authenticate. If the prover P is rational, $\mathcal{A}$ should not receive during the attack enough information allowing him to impersonate P later on in any MF or IF attacks. This is formalized as a two-phase game. During the first phase, $\mathcal{A}$ tries to authenticate with the help of P, in which $p_{\mathcal{A}}$ denotes his success probability. During the second phase, a simulator $Sim_{\mathsf{TF}}(e, \mathsf{IK})$ takes the internal view IK of $\mathcal{A}$ and a valid initial commitment e of P, and tries to authenticate without any further interaction with another legitimate prover (let p_{TF} denotes his success probability). The TF attack conducted by the malicious pair $(P, \mathcal{A})$ is said to be *successful*, if the help of P during the attack does make any difference with respect to its success probability (*i.e.*, if $p_{\mathcal{A}} > p_{\mathsf{TF}}$).

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 round taints the session, which can be formalized by the following definition:

Definition 2 (Tainted Session (TF) [17]). *A time-critical round $\Pi_{\mathsf{sid}}[k, k+1] = (m_k, m_{k+1})$, for some $k \geq 1$ and m_k sent by the verifier, of an adversary-verifier session* sid *is* tainted *if there is an adversary-prover session* sid' *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. As the malicious prover is not allowed to have any feedback from his accomplice during the time-critical rounds of the protocol, this makes the prover's strategy non-adaptive to the challenges sent by the verifier. This also simplifies the construction of a simulator that can match the adversary's winning probability.

The strength of the adversary is quantified in terms of the probability with which he can win this game. In the SimTF game, a TF attack is considered successful as long as the generic simulator is not able to duplicate the adversary's success with the same probability. This captures a wide range of attacks, which are considered as being trivial by weaker models, such as GameTF resistance [17]. Note that some alternative TF definitions also attempt to detect which prover has helped the attacker to be able to punish his attempt. However, these approaches are not necessarily stronger, in the sense of the easiness with which the adversary wins.

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

Definition 3 (SimTF Resistance [17]). *For a DB authentication scheme DB, a $(t, q_{\mathsf{v}}, q_{\mathsf{p}}, q_{\mathsf{obs}})$-terrorist-fraud adversary pair $(P, \mathcal{A})$ and a simulator $Sim_{\mathsf{TF}}(\cdot)$ 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, which has not been* tainted*, with probability $p_{\mathcal{A}}$, and if $Sim_{\mathsf{TF}}(\cdot)$ authenticates in one of q_{v} sessions with the view of $\mathcal{A}$ with probability p_{TF}, then $p_{\mathcal{A}} \leq p_{\mathsf{TF}}$.*

As stated in Table 1, TF resistance is a binary property. Indeed, the accomplice/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 or not.

First, we prove that the generic TREAD construction is SimTF-resistant without relying on any extraction mechanism. This simply means that if the prover provides some information to his accomplice to succeed in the first phase of the TF attack, his accomplice can succeed similarly later without any further help of the prover.

THEOREM 1. *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 $Sim_{\mathsf{TF}}(\cdot)$ that can succeed independently at least as well as $\mathcal{A}$ by using him as a black-box.

Let $p_{\mathcal{A}}$ 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 $Sim_{\mathsf{TF}}(\cdot)$ 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 $Sim_{\mathsf{TF}}(\cdot)$, 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, $Sim_{\mathsf{TF}}(\cdot)$ sends m to $\mathcal{A}$, even though V has sent a different m' to $Sim_{\mathsf{TF}}(\cdot)$. If P sent any additional message to $\mathcal{A}$ in sid before the beginning of the time-critical phases, $Sim_{\mathsf{TF}}(\cdot)$ relays it to $\mathcal{A}$. Hence, from $\mathcal{A}$'s point of view, this is the same as in sid.

Next, the simulator $Sim_{\mathsf{TF}}(\cdot)$ simply forwards the challenges c_i from V to $\mathcal{A}$. If $c_i = 0$, $Sim_{\mathsf{TF}}(\cdot)$ sends the response r_i of $\mathcal{A}$ to V. Otherwise, if $c_i = 1$, $Sim_{\mathsf{TF}}(\cdot)$ needs to adapt the response to m' and then sends $r'_i = r_i \oplus m_i \oplus m'_i$.

Using this strategy, it is clear that $Sim_{\mathsf{TF}}(\cdot)$ can respond to any challenge with a probability at least equal to that the success probability of $\mathcal{A}$. Hence, $Sim_{\mathsf{TF}}(\cdot)$ can authenticate in session sid$'$ with a probability p_{TF}, such that $p_{\mathsf{TF}} \geq p_{\mathcal{A}}$. □

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 TF-resistance result, a stronger result can be obtained. In fact, a more elaborate simulator can amplify any non-negligible advantage of $\mathcal{A}$ until it becomes overwhelming after a polynomial number of sessions with the verifier oracle, without requiring any 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 a successful impersonation attack by the accomplice with an overwhelming probability.

THEOREM 2. *For any adversary $\mathcal{A}$ authenticating with the help of a prover with a non-negligible probability, there is a simulator* amplify *using the internal view of $\mathcal{A}$ and oracle access to a verifier, such that after a polynomial number of steps,* Pr[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. According to the SimTF model, $\mathcal{A}$ has access to the prover only before the beginning of the time-critical phase. Hence, he starts this phase with an initial knowledge IK given by P, and should succeed to authenticate with a probability $p_{\mathcal{A}}$. This information IK can be described as one of these two possibilities:

- The prover sends beforehand 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 a randomized algorithm A to generate these vectors.

In the first case, the prover controls precisely the amount of information given to his accomplice. Let assume for simplicity that the internal view IK of the prover P is only related to the strings α and β. Hence, the vector $\mathbf{c}^1$ would have to be adapted according to the string m provided by the verifier for each new authentication. Let now consider the different ways that P can use to build the vectors $\mathbf{c}^0$ and $\mathbf{c}^1$. One possibility is to provide the perfect information about the two bits $\mathbf{c}^0_i = \alpha_i$ and $\mathbf{c}^1_i = \beta_i$. Another possibility is to only give partial information about one bit, say $\mathbf{c}^0_i = \alpha_i$, and complement the information with one bogus bit $\mathbf{c}^1_i = \overline{\beta_i}$, or $\perp$ that $\mathcal{A}$ would have to guess. In the non-perfect case, the accomplice would succeed to respond to the given verifier's challenge only with probability $\mathsf{q}_{suc} = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0 = \frac{1}{2}$ or $\mathsf{q}_{suc} = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$, respectively, if the verifier uses perfect independent random query bits. Therefore, if the information is perfectly correct in only $n - r$ time-critical rounds, the probability that $\mathcal{A}$ succeeds to authenticate is given by $p_{\mathcal{A}} = 1^{n-r} \cdot \mathsf{q}^r_{suc}$.

In the second case, let us assume that the random process A generates strings with distributions closed to the distributions used to generate the strings α and β. Thus, A can guess α_i and β_i respectively with probabilities $\mathsf{p}_{\alpha,i}$ and $\mathsf{p}_{\beta,i}$. For simplicity but without loss of generality, these probabilities are supposed to be independent. In this case, the probability that $\mathcal{A}$ succeeds to authenticate is given by $p_{\mathcal{A}} = \prod_i \left(\frac{1}{2} \cdot \mathsf{p}_{\alpha,i} + \frac{1}{2} \cdot \mathsf{p}_{\beta,i}\right)$. Obviously, if the original strings have been generated by a perfectly random process, this probability would be equal to 2^{-n} and A would then be totally useless. Note that the first case corresponds simply to $n - r$ time-critical rounds for which $p_{\alpha,i} = p_{\beta,i} = 1$ and r rounds for which $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot p_{\beta,i} \neq 1$, since $p_{\beta,i}$ is either equal to 0 or $\frac{1}{2}$.

Consider now the simulator $Sim_{\mathsf{TF}}(e, \mathsf{IK})$ that tries to impersonate P to the verifier with no further interaction with P. As seen in Theorem 1, TREAD is SimTF-resistant. Therefore, the simulator can authenticate with the probability p_{TF}, which is at least as great as the probability that $\mathcal{A}$ can succeed with the help of P. The next step is to show that if the success probability p_{TF} is non-negligible (*i.e.*, $\exists c, \forall n_c, \exists n > n_c, \mathsf{p}_{\mathsf{TF}} \geq \frac{1}{n^c}$), it can be amplified arbitrarily close to one. Let us define a simulator amplify(e, IK) using $Sim_{\mathsf{TF}}(e, \mathsf{IK})$ internally. This simulator can try $k \cdot n \cdot n^c$ independent authentication experiments with the verifier, for any constant $k > 1$. In such a case, amplify(e, IK) should win at least n experiments with an overwhelming probability, as stated in the following lemma:

LEMMA 1. *For a valid view (e, IK) of an accomplice $\mathcal{A}$, the probability that the simulator* amplify(e, IK) *wins less than n of the $k \cdot n \cdot n^c$ experiments is less than $e^{-\frac{kn}{2}\left(\frac{k-1}{k}\right)^2}$, for any $k > 1$.*

The lemma is derived from the Chernoff bound. If n is large enough, the average number of wins μ is $(k \cdot n \cdot n^c) \cdot \frac{1}{n^c} = k \cdot n$. On the other hand, if $1 - \delta = \frac{1}{k}$ (*i.e.*, $\delta = \frac{k-1}{k}$), $(1 - \delta) \cdot \mu$ is simply n. The lemma follows directly and, as a corollary, if $k \geq 4$, the probability is smaller than $\frac{1}{e^{1.125n}} < \frac{1}{2^n}$.

Assume now that amplify(e, IK) has won n independent experiments. This should happen at least with probability $1 - 2^{-n}$. Their independence follows from the independence of the challenges chosen by the legitimate verifier. At the end of each successful experiment, the simulator amplify would

obtain the explicit values of either α_i or β_i, for all $1 \leq i \leq n$, associated to the commitment value e. At the end of these successful simulations, consider that some bits α_i or β_i have not been recovered. A bit, say α_i, has not been recovered only if the verifier has always asked for the opposite bit β_i in the successful experiments – let $\mathcal{E}_i$ be such an event. This happens only with probability $\Pr[\mathcal{E}_i] = 2^{-n}$, since the verifier's challenges are truly random and are independent.

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

LEMMA 2. *Assume that the simulator* $\mathsf{amplify}(e, \mathsf{IK})$ *has won n experiments. Thus, it should have recovered all the n bits of* α_i and βi *with an overwhelming probability. In fact, the probability that some bits are still unknown is simply* $\Pr[\cup_i \mathcal{E}_i] \leq \sum_i \Pr[\mathcal{E}_i] = \frac{n}{2^n}$.

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

LEMMA 3. *Assume that the simulator* $\mathsf{amplify}(e, \mathsf{IK})$ *has done* $4 \cdot n \cdot n^c$ *authentication experiments. After these experiments, it should have recovered* α *and* β *and be able to impersonate P without his help with an overwhelming probability. Thus,*

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

This concludes the proof of the theorem. □

3.3 Mafia Fraud

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

DEFINITION 4 (TAINTED SESSION (MF) [16]). *A time-critical round* $\Pi_{\mathsf{sid}}[k, k+1] = (m_k, m_{k+1})$, *for some* $k \geq 1$ *and* m_k *sent by the verifier, of an adversary-verifier session* sid *is* tainted *by the phase* $\Pi_{\mathsf{sid}'}[k, k+1] = (m'_k, m'_{k+1})$ *of a prover-adversary session* sid′ *if*

$$(m_k, m_{k+1}) = (m'_k, m'_{k+1}),$$
$$\mathsf{marker}(\mathsf{sid}, k) < \mathsf{marker}(\mathsf{sid}', k),$$
$$\textit{and} \qquad \mathsf{marker}(\mathsf{sid}, k+1) > \mathsf{marker}(\mathsf{sid}', k+1).$$

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

DEFINITION 5 (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* $\mathbf{Adv}^{\mathsf{MF}}_{DB}(\mathcal{A})$ *that* $\mathcal{A}$ *wins this game.*

We now prove that TREAD is MF-resistant.

THEOREM 3. *If the challenges are drawn randomly from a uniform distribution by the verifier,* E *is an* IND-CCA2*-secure encryption scheme and* S *is* EUF-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 for previous ones. It can be reduced to the security analysis of a simpler version of the protocol, using the game-hopping technique formalized by Shoup in [23]. 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 the reuse of nonces 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 5 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 $q_p^2/2^{2n}$. Thus, using Shoup's difference lemma, $|\Pr[\Gamma_0] - \Pr[\Gamma_1]| \leq q_p^2/2^{2n}$, 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 EUF-CMA resistance of S. The reduction simply consists in starting EUF-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 EUF-CMA experiment. From the Shoup's 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 IND-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 IND-CCA2 experiment on E is negligible.

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

Prover simulation: $\mathcal{B}$ generates two challenge messages: $m_0 = (\delta||\mathsf{idprv}(P)||\mathsf{S.sig}_{\mathsf{sk}}(\delta||\mathsf{idprv}(P)))$ and $m_1 = (\alpha||\beta||\ \mathsf{S.sig}_{\mathsf{sk}}(\alpha||\beta||\mathsf{idprv}(P)))$, in which $\alpha||\beta$ and δ are random binary strings of length $2n$. 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, 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, 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 m_0 or 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 difference in the probabilities of the two games can be transformed into an $\mathsf{IND\text{-}CCA2}$ advantage. Thus, from the difference lemma, we have $|Pr[\Gamma_2] - Pr[\Gamma_3]| \leq \mathsf{Adv}_{\mathsf{E}}^{\mathsf{IND\text{-}CCA2}}(\lambda)$, which is negligible by hypothesis.

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 makes a wrong guess for 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 at authenticating from a distance greater than d_{max}. In DH attacks, the adversary attempts to do the same, but he uses the unintentional help of a legitimate prover located close to the verifier. The remote adversary may initiate the DB protocol and let a nearby prover complete the DB phase.

Although the DH attacks are generally real threats against most the DB protocols, they do not represent a realistic threat against 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 legitimate group $\mathcal{X}$ 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, consider an adversary (here a malicious prover) able to use the help of an honest prover in the verifier's proximity. In the DB phase, he commits to a response in advance (before the challenge of that round) and sends this commitment. These commitments do not refer to cryptographic commitments (with the hiding and binding properties), but rather to 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 nearby 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 $\mathsf{sid_{Commit}}$ for committed responses and $\mathsf{sid_{Prompt}}$ 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. The formal definition of tainted phases is as follows.

Definition 6 (Tainted Session (DH)). *A time-critical round $\Pi_{\mathsf{sid}}[k, k+1] = (m_k, m_{k+1})$, for some $k \geq 1$ and m_k sent by the verifier, of an adversary-verifier session* sid *is* tainted *if one of the following conditions holds.*

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

This definition rules out some potential actions of the adversary. Afterwards, the game-based definition of DH resistance notion can be stated as follows.

Definition 7 (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 without any critical round being tainted. Thus, the DH resistance is defined as the probability $\mathbf{Adv}^{\mathsf{DH}}_{DB}(\mathcal{A})$ 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 4. *If the challenges are drawn from 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 provided in Appendix B.1.

3.5 Privacy

We now establish that the public-key instance of our protocol preserves the privacy of the provers against external eavesdroppers. In particular, 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 [19] has been formalized as follows:

DEFINITION 8 (PRIVACY PROTECTION). *Let DB be a DB scheme. The privacy experiment* $\mathsf{Exp}^{Priv}_{\mathcal{A},DB}(\lambda)$ *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, the adversary* $\mathcal{A}$ *has access to the following oracles:*

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

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

$\overline{DB}.\mathsf{Prover}_b$ *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*, which means that the guess of* $\mathcal{A}$ *is correct, while otherwise he outputs* 0.

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

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

and the advantage on the privacy experiment as

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

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

THEOREM 5. *If* PKE *is an* IND-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*

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

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

3.6 Prover Anonymity

Finally, we show that the anonymous version of our protocol preserves the anonymity of the provers even against malicious verifiers. These verifiers may try to profile legitimate provers by linking their authentication sessions, thus threatening their privacy. For instance, this threat is particularly relevant for the public transportation system scenario described in the introduction.

The only information on a prover identity that a verifier can get during the protocol is the signatures produced by the prover. Therefore, if a secure group signature scheme is used, the protocol would 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 G-SIG. In such protocols, $\mathsf{idprv}(P)$ should only identify the corresponding group identity. Thus, the verifier should not obtain any information on a prover identity. To formalize this property, we generalize the model of [19] drawing on the anonymity model of revocable group signature [22].

DEFINITION 9 (PROVER ANONYMITY). *Let DB be a DB scheme. The anonymity experiment* $\mathsf{Exp}^{Anon}_{\mathcal{A},DB}(\lambda)$ *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:*

$\overline{\mathsf{DB}}.\mathsf{Join}^h(\cdot)$: *On input* i*, it creates a new legitimate 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* $b \xleftarrow{\$} \{0,1\}$*.* $\mathcal{A}$ *then accesses* $\overline{DB}.\mathsf{Revoke}(\cdot)$ *and* $\overline{DB}.\mathsf{Corrupt}(\cdot)$ *on* i_0 *and* i_1 *(the oracles return* $\perp$ *if* $\mathcal{A}$ *uses these inputs). Hereafter,* $\mathcal{A}$ *has now access to the following challenge oracle:*

$\overline{DB}.\mathsf{Prover}_b$ *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*, which means that the guess of* $\mathcal{A}$ *is correct, while otherwise he outputs* 0.

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

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

and the advantage on the PA experiment as

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

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

THEOREM 6. *If* G-SIG *is an anonymous revocable group signature scheme [22] 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*

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

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

4. CONCLUSION

In this paper, we have introduced 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 succeed a new authentication session with the same probability. However, this comes at the cost of a slightly lower MF and MF resistance.

We have also reinforced 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 impersonate this prover in further sessions to an overwhelming probability. We have also presented three instances of our protocol. 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. Finally, 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.

5. REFERENCES

[1] A. Ahmadi and R. Safavi-Naini. Privacy-preserving distance-bounding proof-of-knowledge. In *Proc. of Information Security and Cryptology*, pages 74–88. Springer-Verlag, 2014.

[2] G. Avoine, M. A. Bingol, S. Karda, C. Lauradoux, and B. Martin. A formal framework for analyzing RFID distance bounding protocols. *Journal of Computer Security - Special Issue on RFID System Security, 2010*, pages 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. On the pseudorandom function assumption in (secure) distance-bounding protocols: PRF-ness alone does not stop the frauds! In *Proc. of LATINCRYPT*, pages 100–120. Springer-Verlag, 2012.

[6] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical & provably secure distance-bounding. Cryptology ePrint Archive, Report 2013/465, 2013. http://eprint.iacr.org/2013/465.

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

[8] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Towards secure distance bounding. In *Proc. of FSE*, pages 55–67. Springer-Verlag, 2014.

[9] I. Boureanu and S. Vaudenay. Optimal proximity proofs. In *Proc. of Inscrypt*, pages 170–190. Springer-Verlag, 2014.

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

[11] A. Brelurut, D. Gérault, and P. Lafourcade. Survey of distance bounding protocols and threats. In *Proc. of the Foundations and Practice of Security*, volume 9482, pages 29–49. Springer-Verlag, 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*, pages 121–133. 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*, pages 222–238. Springer-Verlag, 2005.

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

[15] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In *Proc. of CRYPTO*, LNCS, pages 21–39. Springer-Verlag, 1988.

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

[17] M. Fischlin and C. Onete. Terrorism in distance bounding: Modeling terrorist fraud resistance. In *Proc. of Applied Cryptography and Network Security*, pages 414–431. Springer-Verlag, 2013.

[18] 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.

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

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

[21] 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*, pages 98–115. Springer-Verlag, 2008.

[22] 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 (PKC)*, pages 463–480. Springer-Verlag, 2009.

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

[24] S. Vaudenay. On privacy models for RFID. In *Proc. of ASIACRYPT*, pages 68–87. Springer-Verlag, 2007.

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

[26] S. Vaudenay. Sound proof of proximity of knowledge. In *Proc. of ProvSec*, pages 105–126. Springer-Verlag, 2015.

APPENDIX

A. DEFINITIONS

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

DEFINITION 10. *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 a global security parameter* λ*.*

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

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

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

DEFINITION 11. *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 a global security parameter* λ*.*

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

$\mathsf{PKE.dec}_{\mathsf{sk}}(c)$**:** *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 the equality* $\mathsf{PKE.dec}_{\mathsf{sk}}(\mathsf{PKE.enc}_{\mathsf{pk}}(m)) = m$ *holds for any message* m *and any key pair* $(\mathsf{pk}, \mathsf{sk})$ *generated by* $\mathsf{PKE.gen}$*.*

DEFINITION 12. *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 advantage probability* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{A},\mathsf{SKE}}(1^\lambda)$ *is negligible:*

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

in which the oracles $\overline{\mathsf{SKE}}.\mathsf{enc}_{\mathsf{k}}(\mathsf{LR}_b), \overline{\mathsf{SKE}}.\mathsf{dec}_{\mathsf{k}}$ *are defined as:*

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

$\overline{\mathsf{SKE}}.\mathsf{dec}_{\mathsf{k}}(c)$**:** *if* c *has been generated by* $\overline{\mathsf{SKE}}.\mathsf{enc}_{\mathsf{k}}(\mathsf{LR}_b)$ *returns* $\perp$*, while otherwise it returns* $\mathsf{SKE.dec}_{\mathsf{k}}(c)$*.*

DEFINITION 13. *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 advantage probability* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{A},\mathsf{PKE}}(1^\lambda)$ *is negligible:*

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

in which the oracles $\overline{\mathsf{PKE}}.\mathsf{enc}_{\mathsf{pk}}(\mathsf{LR}_\mathsf{b}), \overline{\mathsf{PKE}}.\mathsf{dec}_{\mathsf{sk}}$ *are defined as:*

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

$\overline{\mathsf{PKE}}.\mathsf{dec}_{\mathsf{sk}}(c)$**:** *if* c *has been generated by* $\overline{\mathsf{PKE}}.\mathsf{enc}_{\mathsf{pk}}(\mathsf{LR}_\mathsf{b})$ *returns* $\perp$*, while otherwise it returns* $\mathsf{PKE.dec}_{\mathsf{sk}}(c)$*.*

DEFINITION 14. *A* message authentication code *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 a global 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 the equality* $\mathsf{MAC.ver}_{\mathsf{sk}}(m, \mathsf{MAC.sig}_{\mathsf{sk}}(m)) = 1$ *holds for any message* m *and any key* sk *generated by* $\mathsf{MAC.gen}$*.*

DEFINITION 15. *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 a global 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 the equality* $\mathsf{SIG.ver}_{\mathsf{pk}}(m, \mathsf{SIG.sig}_{\mathsf{sk}}(m)) = 1$ *holds for any message* m *and any key pair* $(\mathsf{sk}, \mathsf{vk})$ *generated by* $\mathsf{SIG.gen}$*.*

DEFINITION 16. *Let* $\mathsf{MAC} : (\mathsf{MAC.gen}, \mathsf{MAC.sig}, \mathsf{MAC.ver})$ *be a message authentication code.* MAC *is said to be* unforgeable against chosen message attack *(*$\mathsf{EUF\text{-}CMA}$*) when for any adversary* $\mathcal{A}$*, the advantage probability* $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MAC}}(1^\lambda)$ *is negligible:*

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

in which 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* $\perp$*, while otherwise it returns* $\mathsf{MAC.ver}_{\mathsf{k}}(m, s)$*.*

DEFINITION 17. *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 message attack *when for any adversary* $\mathcal{A}$*, the advantage probability* $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{SIG}}(1^\lambda)$ *is negligible:*

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

in which 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* $\perp$*, while otherwise it 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 18. *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 global 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_i *(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* $\mathsf{reg}_i, \mathsf{gpk}$ *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, while otherwise it returns* 0*.*

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

DEFINITION 19. *Let* G-SIG *be a group signature scheme. The anonymity experiment* $\mathsf{Exp}^{\mathit{Anon}}_{\mathcal{A},\mathsf{G\text{-}SIG}}(\lambda)$ *for the adversary* $\mathcal{A}$ *on* G-SIG *is defined as follows.* $\mathcal{A}$ *interacts 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* i*, creates* P_i *with* $\mathsf{G.join}_{\mathsf{msk}}(i, \mathsf{gpk}, \mathsf{UL})$*.*

$\overline{G}.\mathsf{Join}^c(\cdot)$: *on* 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* i*, revokes* P_i *with* $\mathsf{G.rev}_{\mathsf{msk}}(i, \mathsf{RL}, \mathsf{UL}, \mathsf{gpk})$*, updates RL and returns it.*

$\overline{G}.\mathsf{Corrupt}(\cdot)$: *on* 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* 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* 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* m*, returns* $\mathsf{G.sig}_{\mathsf{ssk}_{i_b}}(m)$*.*

Finally, $\mathcal{A}$ *returns* b'*. If* $b = b'$*, the challenger returns* 1*, which means that the guess of* $\mathcal{A}$ *is correct, while otherwise he outputs* 0*.*

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

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

and the advantage on the experiment as

$$\mathsf{Adv}^{\mathit{Anon}}_{\mathsf{G\text{-}SIG}}(\lambda) = \max_{\mathcal{A} \in \mathsf{Poly}(\lambda)} \{\mathsf{Adv}^{\mathit{Anon}}_{\mathcal{A},\mathsf{G\text{-}SIG}}(\lambda)\}.$$

A group signature G-SIG *is anonymous when if* $\mathsf{Adv}^{\mathit{Anon}}_{\mathsf{G\text{-}SIG}}(\lambda)$ *is negligible.*

B. SECURITY PROOFS

B.1 Distance-hijacking Resistance

PROOF PROOF OF THEOREM 4. Note that if $\mathcal{A}$ uses the message Prompt as 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 a 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 $\mathcal{A}$ 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 $\mathbf{Adv}^{\mathsf{DH}}_{\mathsf{DB}}(\mathcal{A})$ is at most $\left(\frac{3}{4}\right)^n$, which is negligible. □

B.2 Privacy Property

PROOF PROOF OF THEOREM 5. Assume that there exists a polynomial-time adversary $\mathcal{A}$ s. t. $\mathsf{Adv}^{\mathsf{Priv}}_{\mathcal{A},\mathsf{TREAD}^{\mathsf{Pub}}}(\lambda)$ is non-negligible. We show how to build an adversary $\mathcal{B}$ s. .t. $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{B},\mathsf{PKE}}(\lambda)$ 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 generate 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 i, $\mathcal{B}$ returns a public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ of the 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 retrieves the message $(\alpha||\beta||\mathsf{idprv}(P_i)||\sigma_p) = \mathsf{PKE.dec}_{\mathsf{sk}_v}(e)$ 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 $\{0, \ldots, n\}$.

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 $\mathcal{A}$, while otherwise he returns 0.

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

Initialization phase $\mathcal{B}$ picks $\alpha||\beta \xleftarrow{\$} \{0,1\}^{2n}$ and computes the two 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 then 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 $\mathsf{SKE.enc_k}(\mathsf{LR}_b(\cdot,\cdot))$ 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, $\overline{\mathsf{DB}}.\mathsf{Join}^c(\cdot)$ and $\overline{\mathsf{DB}}.\mathsf{Prover}(\cdot)$ are simulated by $\mathcal{B}$ as in the first phase of the experiment. Hence, $\overline{\mathsf{DB}}.\mathsf{Verifier}$ can be simulated as follows:

Initialization phase $\mathcal{B}$ receives e from $\mathcal{A}$. If there is no $0 \leq d \leq l$ s. t. $\text{List}_d = (\alpha, \beta, e)$, $\mathcal{B}$ 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 for the response r_j. He 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. □

B.3 Anonymity Property

Proof Proof of Theorem 6. Assume that there exists a polynomial-time adversary $\mathcal{A}$ s. t. $\mathsf{Adv}^{\mathsf{Anon}}_{\mathcal{A},\mathsf{TREAD}^{ANO}}(\lambda)$ is non-negligible. We show how to construct an adversary $\mathcal{B}$ s. t. $\mathsf{Adv}^{\mathsf{Anon}}_{\mathcal{B},\mathsf{G\text{-}SIG}}(\lambda)$ 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 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 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 i, revokes P_i with $\overline{G}.\mathsf{Revoke}(\cdot)$, which updates RL and returns it.

$\overline{\mathsf{DB}}.\mathsf{Corrupt}(\cdot)$: on 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 $\alpha||\beta \xleftarrow{\$} \{0,1\}^{2n}$ and uses his oracle $\overline{G}.\mathsf{Sign}(\cdot,\cdot)$ to get the corresponding 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 retrieves the message $(\alpha||\beta||\sigma_p) = \mathsf{PKE.dec}_{\mathsf{sk}_v}(e)$. 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 $\mathcal{A}$, while otherwise 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 $\mathcal{A}$, This implies that $\mathcal{B}$ wins his experiment with the same probability that $\mathcal{A}$ wins his experiment. 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 that an adversary should have a negligible advantage on $\mathsf{G\text{-}SIG}$. □