Privacy-Preserving Distance-Bounding Proof-of-Knowledge

Ahmad Ahmadi(✉) and Reihaneh Safavi-Naini

University of Calgary, Calgary, Canada
ahmadi@ucalgary.ca

Abstract. In distance-bounding protocols a prover wants to prove that it is located within a distance bound $\mathbb{D}$ from a verifier. Distance-bounding (DB) protocols have numerous applications including authentication and proximity checking. The privacy problem in DB protocols was limited to privacy against *MiM* adversaries. Gambs *et al.* [11] extended this limitation and proposed a protocol that provides *strong* privacy when the verifier is malicious, or *honest-but-curious* registration authority. The protocol however does not provide resistance against terrorist-fraud.

In this paper we consider private DB protocols that provide the strongest level of security against all known DB attacks, in particular terrorist-fraud, and provide anonymity of the prover and unlinkability of its sessions against *malicious* verifiers and assuming an *honest-but-curious* registration authority. We define private distance-bounding as a special ZKPoK in which a prover presents a commitment on its long-term private-key, and later proves in zero-knowledge that; (i) she knows the committed value, (ii) she knows a signature of the authority on the committed value (registration proof), and (iii) she is located within a pre-defined distance to the verifier. The prover stays anonymous and its sessions will be unlinkable. We propose a protocol PDB with these properties that resists against all known attacks including terrorist-fraud. PDB is based on Bussard-Bagga [5] (DBPK-Log). PDB also fixes the vulnerability of the protocol pointed out by Bay *et al.* [2] resulting in a secure public-key DB protocol, hence answering the open question of constructing a secure public-key DB protocol.

1 Introduction

In DB protocols, there are two types of entities; provers and verifiers. In concurrent execution of DB protocols, multiple protocol instances are run at the same time. Provers and verifiers are usually connected to a back-end server, which only takes care of the registration phase and is silent otherwise.

Secure DB protocols provide two functionalities: (1) authentication of a registered prover to a verifier, and (2) bounding the prover's distance to the verifier. The bounding is commonly by measuring the round trip time in a *fast-exchange* phase between the prover and the verifier, during which the verifier presents challenges to the prover and verifies if the prover's responses are correct. The round-trip times of correct challenge and responses are used to estimate the

L.C.K. Hui et al. (Eds.): ICICS 2014, LNCS 8958, pp. 74–88, 2015.
DOI: 10.1007/978-3-319-21966-0_6

distance between prover and verifier, and then compare it against a specified distance bound $\mathbb{D}$.

Privacy is a necessary property in many location-based services. One approach for providing privacy in location-privacy, is to modify the geometric data, for example loose accuracy of location data, to achieve privacy [15,21]. In distance-bounding however, the location accuracy is a requirement. To achieve both privacy and location accuracy, one can unlink the location data from provers' authentication information. This problem has been well studied in the context of RFID systems [13,22]. In these systems the location of provers are known by the verifiers, but provers hide their identity in their interactions. In RFID systems however, verifiers are assumed trusted, and the privacy is only against Man-In-the-Middle (MiM) adversaries. The assumption of trusted verifier is not acceptable in many distance-based location services in which the verifiers can get exploited, and so it is important to consider stronger privacy models, in particular consider untrusted verifiers.

In a symmetric-key DB protocol, there is a shared-key between prover and registration authority and so privacy is not achievable against an adversary who can access the internal state of the authority. One can always achieve prover privacy by using the same key for all provers. This however is unacceptable because of the threat it exposes to the whole system. There are three famous public-key based DB protocols in the literature: (a) The seminal Brands-Chaum protocol [4], which uses commitments and signatures. The protocol is not secure against Terrorist-Fraud Attack (TFA). (b) Bussard-Bagga [5] protocol (DBPK-Log), which uses bit commitment and was designed to provide TFA resistance, but it was recently broken [2]. And, (c) the recent Hermans *et al.* [14] protocol, which uses elliptic curve cryptography and does not provide TFA resistance. It also does not allow the privacy adversary to access the internal state of the registration authority. Therefore, there is no DB protocol, which is both public-key based and is secure against terrorist-fraud adversary.

The following questions have been open in the literature. (i) Design a public-key based DB protocol that is, secure against terrorist-fraud adversary; and (ii) design a secure privacy preserving DB protocol that provides privacy for provers against a privacy adversary who controls the verifier, and has access to the internal state of registration authority.

In this paper we answer to these open problems. Our contributions is three-fold; first we propose a privacy model which we refer to it as *extensive-privacy*, and show that it is stronger than the *wide-privacy* notion of [22], and the privacy model of Gambs *et al.* [11], with respect to the adversary's access to the state of the *non-prover* entities. Second, we fix the security flaw of DBPK-Log protocol, and achieve the first public-key based distance-bounding protocol, which is secure against TFA adversary (called DBPK-log^{+} protocol). Finally, we propose the protocol PDB, as an extension of DBPK-log^{+}, which is secure in the proposed privacy model.

2 Background

In DB systems, since the seminal paper [4], the implicit assumption about all secure DB protocols is the presence of a secure function which generates and distributes the private-keys of the entities. In some cases this function can be executed by a verifier. A distance-bounding protocol allows a registered *prover* to prove that she is within a distance bound to *verifier*, and in possession of the secret-key which is used for user authentication information.

In order to have a *privacy-preserving distance-bounding* system, we need to consider three properties; *Authentication*, *Distance-Bounding* and *Privacy*. By having *authentication* property, the verifiers determines whether an approaching entity is indeed a legitimate prover or not. *Distance-Bounding* property guarantees that the prover is located within a pre-defined distance. And *privacy* property provides assurance to provers that their interactions in the system is not traceable.

Authentication is obtained by a protocol between *prover* and *verifier*. The prover must prove that she knows a secret value which is registered to the system. The protocol must satisfy two properties; *correctness* and *soundness*. The *correctness* holds when the honest verifier accepts, if an honest prover, who knows the secret value is involved. The *soundness* holds when no adversary who doesn't have access to the secret value of a registered prover, can convince the honest verifier to accept in the authentication. Gambs *et al.* [11] consider these two properties, but make two assumptions: (1) the *server* revokes all corrupted provers upon corruption, and (2) dishonest *provers* cannot yield their private-key to the adversary, unless they get un-registered. Implementing these assumptions are major challenges, and so one of the goals of our work is to weaken these assumptions.

Distance-Bounding protocols run a *fast-exchange* phase, which guarantees presence of the owner of the secret-key within distance bound $\mathbb{D}$. By assuming that no prover is willing to disclose her secret-key to others, five attacking scenarios have been studied in *DB* protocols: *Distance-Fraud Attack (DFA)* [4]*;* a dishonest prover $\mathcal{P}^*$, which is not located within distance $\mathbb{D}$ to verifier $\mathcal{V}$, tries to convince $\mathcal{V}$ that she is located within distance $\mathbb{D}$ to $\mathcal{V}$. *Mafia-Fraud Attack (MFA)* [8]*;* an adversary $\mathcal{A}$, which is located within distance $\mathbb{D}$ to $\mathcal{V}$ (between $\mathcal{V}$ and far away honest prover $\mathcal{P}$), convinces $\mathcal{V}$ that $\mathcal{P}$ is close-by. *Terrorist-Fraud Attack (TFA)* [8]*;* an adversary $\mathcal{A}$ (located within distance $\mathbb{D}$ to $\mathcal{V}$) co-operates with a dishonest prover $\mathcal{P}^*$ (far away) to convince $\mathcal{V}$ that $\mathcal{P}^*$ is located within distance $\mathbb{D}$ to $\mathcal{V}$. *Distance-Hijacking* [7]*;* a dishonest prover $\mathcal{P}^*$, which is not located within the distance $\mathbb{D}$ to a verifier $\mathcal{V}$, exploits some honest provers $\mathcal{P}_1, \dots, \mathcal{P}_n$ to mislead $\mathcal{V}$ about the actual distance between $\mathcal{P}^*$ and $\mathcal{V}$. *Impersonation-Attack* [1]*;* a dishonest prover $\mathcal{P}^*$ purports to be another prover in her interaction with $\mathcal{V}$.

Vaudenay *et al.* [23] proposed a general attack model, which captures all these attacks;

- **Distance-Fraud** is defined to capture the classic DFA and *Distance-Hijacking.*
- **MiM** attack captures MFA and *Impersonation-Attack.*
- **Collusion-Fraud** is a different game-based definition about TFA. Based on this definition, if there is any PPT adversary who can win the TFA game with probability γ, then there exist a weaker MiM adversary who can succeed in a specific MiM game with probability γ'.

Authentication and *Distance-Bounding* have been studied in `DB` protocols. Recently, a considerable amount of focus have been on proposing formal definitions and provably secure of symmetric `DB` protocols [3,9,10]. In Dürholz *et al.* [9], the strong simulation-based terrorist-fraud for "single prover, single verifier" setting have been defined. Fischlin-Onete [10] have extended [9] and defined even stronger simulation-based model and proposed a protocol with security proof. On the other hand, Boureanu *et al.* [3] showed that the definition of Dürholz *et al.* is too strong, and proposed a general and practical game-based terrorist-fraud model for "multiple prover, multiple verifier" setting (same model as [23]). Boureanu *et al.* proposed the *SKI* protocol, which is claimed to be secure against the defined *distance-fraud*, *MiM* and *collusion-fraud*. By the way, this protocol is not proven to be secure under the defined *collusion-fraud* adversary, despite their claim. The provided proof is just for deterministic PPT adversaries in the TFA game, rather than any PPT adversary in this game.

Privacy is considered as un-traceablility of different sessions of a single prover. This notion of *privacy* have been well studied in the RFID framework against MiM adversaries, which mostly use symmetric setting and assume trusted verifier and registration authority [13,22]. In Hermans *et al.* [13] model, privacy is defined as a game between an *adversary* and a *challenger*. The adversary has oracle access to the following functionalities; create honest provers (`CreateProver`), launch a session of a pre-defined protocol (`Launch`), ask the *challenger* to choose one of the given two provers and return an anonymous handle (`DrawProver`), send message to an anonymous prover (`SendProver`), free the anonymous handle of a prover (`Free`), send message to the verifier in a protocol session (`SendVerifier`), see output of the verifier in a session of protocol (`Result`), and finally get the non-volatile internal state of an honest prover (`Corrupt`). The adversary wins the privacy game if she can find out the chosen bit of challenger in `DrawProver` oracle. Peeters-Hermans [19] added a new oracle to the above list by which, the adversary can create insider prover and have control on it (`CreateInsider`).

Vaudenay [22] have classified the adversaries, based on their access to the above oracles. A *wide* adversary has access to `Result` oracle, otherwise it will be a *narrow* adversary. In parallel, the adversary can be *weak* (no access to `Corrupt` oracle), *forward* (`Corrupt` queries can only be followed by other `Corrupt` queries), *destructive* (`Corrupt` queries destroys the access to the corrupted prover), and *strong* (unlimited access to `Corrupt` oracle). Paise-Vaudenay [17] showed that *destructive* and *strong* privacy is not achievable in symmetric-key systems.

Gambs *et al.* [11] built of the work of Hermans *et al.* [14] to define public-key DB protocols that are privacy-preserving and constructed the first protocol, which is secure against three separate adversaries: (1) MiM adversary, (2) a MiM adversary who has access to the internal state of the verifier, and (3) an honest-but-curious adversary who knows the internal state of the verifier and the registration server. We extend this model and introduce a stronger adversary who has access to the internal state of verifiers and registration server (*extensive*[1] adversary). Therefore, the following order in the new classification holds; $NARROW \subseteq WIDE \subseteq$ Gambs *et al.* $\subseteq EXTENSIVE$ based on having access to the state of *verifiers* and *registration authority*, as well as $WEAK \subseteq FORWARD \subseteq DESTRUCTIVE \subseteq STRONG$ based on having access to the state of *provers*.

In this paper we propose a new protocol, which provides *authentication*, *distance-bounding* (*distance-fraud*, *MiM* and *terrorist-fraud* resistance) and *privacy* (against *extensive-weak* adversary).

2.1 Distance-Bounding Proof-of-Knowledge (DBPK-Log)

Bussard-Bagga [5] proposed the only public-key DB protocol which was designed to be secure against TFA adversary. This protocol combines a *fast-exchange* DB protocol, Pedersen commitment scheme [18] and *zero-knowledge proof-of-knowledge* [20]. In this protocol, a prover chooses a key pair and registers the public-key with a trusted server. The verifiers are trusted and have access to the public-key of provers. The system parameters are set by a trusted authority, and uses a cyclic group whose order is a strong prime. These parameters are shared and used by all the participants. This protocol was designed to be secure against DFA, MFA and TFA adversaries, while the information leakage about the private-key of provers is minimal.

The protocol combines the bitwise operation, used for *fast-exchange* phase, and modular operations that are required for commitment schemes. This results in some security loss of the secret-keys, while maintaining indistinguishability of the secret-keys.

Bay *et al.* [2] showed TFA and DFA attacks on this protocol, which takes advantage of poor auditing of un-used elements in the `Commitment Opening` phase (i.e. half of the bit commitments won't get opened).

2.2 BBS$^+$ Signature Scheme [6]

This signature scheme uses bilinear mapping and supports signing of committed message block[2] $M = \{m_1, \ldots, m_L\}$, without knowing about the actual message. Two entities are invloved in this scheme; a trusted signer (S) who holds the signing key, and a client (C) who knows a message block.

This signature scheme, follows the standard operations [12] of all signature schemes BBS$^+$ = (`KeyGen`, `Sign`, `Verify`):

[1] Extensive privacy will be defined in Definition 5.

[2] A single message that is represented as string of integers.

– $(sk_S, pk_S = h_0^{sk_S}) \leftarrow \texttt{KeyGen}(1^\lambda)$; S creates a key pair and publishes the public-key.
– $\texttt{Sign}\big[\texttt{C}(M, pk_S; \sigma = sign_S(M)) \leftrightarrow \texttt{S}(sk_S; partial(\sigma), cmt(M))\big]$, s.t. $M = \{m_1, \ldots, m_L\}$, $\sigma = (A, e, s)$, $partial(\sigma) = (A, e)$, $cmt(M) = (\{C_i = g_1^{m_i} g_2^{r_i}\}, C_M = g_1^{s'} g_2^{m_1} \ldots g_{L+1}^{m_L})$, and $A = (g_0 g_1^s g_2^{m_1} \ldots g_{L+1}^{m_L})^{\frac{1}{e+sk_S}}$ for random values of $\{r_i\}, s, s', e$; C and S get involved in a protocol for signing a committed message block (M). In this protocol, first C calculates the commitment $cmt(M) = (\{C_i\}, C_M)$ as mentioned above, and sends it to S, then they run $PoK\{(\{m_i, r_i\}, s') : C_M \bigwedge \{C_i\}\}$ to verify the possession and integrity of the values in the commitment. Then S creates the signature as $A = (g_0 g_1^{s''} C_M)^{\frac{1}{e+sk_S}}$ for random e and s'' and sends $\{A, e, s''\}$ to C. And finally C calculates $s = s' + s''$, checks the validity of $\sigma = \{A, e, s\}$ and keeps σ as the signature of S on M.
– $\texttt{Verify}\big[\texttt{C}(M, \sigma; \emptyset) \leftrightarrow \texttt{E}(pk_S; Out_P, \{C'_i\})\big]$, s.t. $C'_i = g_1^{m_i} g_2^{r'_i}$ for random r'_i; C and any entity(E) with access to the public parameters of system, get involved in a protocol for proving the possession of a signature on a committed message block. First C creates new commitments on each element of the message block $\{C'_i\}$ and sends it to E. Then they run a *signature proof-of-knowledge* $SPK\{(A, e, s, m_1, \ldots, m_L) : A = (g_0 g_1^s g_2^{m_1} \ldots g_{L+1}^{m_L})^{\frac{1}{e+sk_S}}\}$ to prove possession of a valid signature on M, which is committed by $\{C'_i\}$. S returns $Out_P = 1$ if no error happens.

This signature scheme provides two extra properties, beside the standard properties of signature schemes (*authenticity*, *integrity* and *non-repudiation*), as follows:

– *blind-sign*: `Sign` protocol allows a client to obtain the signature of signer on a message, which is committed as $cmt = Commit(M)$ using Pedersen commitment. The client uses *ZKPoK* to prove that the committed values are correctly calculated from message. The protocol perfectly hides the message from the signer. The signature is secure under LRSW assumption [16].
– *blind-verify*: In `Verify` protocol, the client computes a *non-interactive honest-verifier zero-knowledge proof-of-knowledge (SPK)* protocol in order to prove to any verifier that she knows a message block (M) and a signature on it, where the commitment of the message is presented. The proof does not leak any information about the message and the signature.

3 Model

There are three types of entities; a set of untrusted *provers* $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, a set of untrusted *verifiers* $\mathcal{V} = \{\mathcal{V}_1, \ldots, \mathcal{V}_m\}$, and a honest-but-curious *registration authority* $(\mathcal{RA})$ with a key-pair $(pk_{\mathcal{RA}}/sk_{\mathcal{RA}})$. We assume $\mathcal{RA}$ can generate her key pair. Each registered prover $\mathcal{P}_i$ has a secret key sk_i, and a certificate of $\mathcal{RA}$ for it $(\sigma_i = Sign_{\mathcal{RA}}(sk_i))$. The communication channels are public. We assume there exists a public and secure board which keeps the public parameters of the system, including the public-key of *registration authority* and every entity has secure read access to the board.

There are three operational phases in this model; (1) $\mathcal{RA}$ makes a key pair by executing "`KeyGen`" phase and puts the public-key on the public board. (2) In "`Registration`" phase, a new prover ($\mathcal{P}_i$) with a chosen secret-key (sk_i), and $\mathcal{RA}$ interact, resulting in $\mathcal{P}_i$ obtaining a registeration certificate (sk_i, σ_i). (3) In "`DB`" phase, a registered prover $\mathcal{P}_i$ interacts with a verifier $\mathcal{V}_j$, that has read access to the public board. At the end of this phase, $\mathcal{P}_i$ proves that she knows a secret key (sk_i), she has a signature of $\mathcal{RA}$ on sk_i, and is located within a distance bound $\mathbb{D}$ to $\mathcal{V}_j$. The verifier returns a single bit $Out_{\mathcal{V}}$ as the output of protocol ($Out_{\mathcal{V}} = 1$ for accept and $Out_{\mathcal{V}} = 0$ for reject).

Definition 1. Correctness: *for any pair of honest prover and honest verifier, with mutual distance of at most $\mathbb{D}$, the "*`DB`*" protocol should always return $Out_{\mathcal{V}} = 1$.*

The protocol's **soundness** is against two types of adversaries: distance-bounding adversary ($\mathcal{A}_{DB}$) and privacy adversary ($\mathcal{A}_P$). The distance-bounding adversary $\mathcal{A}_{DB}$ can, (1) read the public board, and (2) control the corrupted provers. The aim of $\mathcal{A}_{DB}$ is to convice an honest verifier to return $Out_{\mathcal{V}} = 1$ as the output of `DB` operation. The privacy adversary $\mathcal{A}_P$ can, (1) read the public board, and (2) read the internal state of $\mathcal{RA}$, and (3) control all verifiers and corrupted provers. The aim of $\mathcal{A}_P$ is to distinguish between two honest provers based on their interactions with the system.

We define three general *DB* attacks. The definition of *distance-fraud* and *MiM* attacks are in-line with Vaudenay *et al.*'s [23] approach, but the proposed *terrorist-fraud* attack is following the classic definition. First, a simple two-party attack is considered, where a dishonest prover is far away from the verifier, but wants to convince the verifier that she is within the distance.

Definition 2. α-resistance Distance-Fraud: *For any PPT adversary $\mathcal{A}$, which is not located within the distance $\mathbb{D}$ to an honest verifier $\mathcal{V}_j$, and is able to run a* `Registration` *session with $\mathcal{RA}$, the probability of returning $Out_{\mathcal{V}} = 1$ in an interaction with $\mathcal{V}_j$ is not more than α.*

This definition captures *Distance-Hijacking*, in which a dishonest far-away prover $\mathcal{P}^*$ may mis-use some honest provers to successfully authenticate to $\mathcal{V}_j$. In this attack, the adversary is allowed to mis-behave in the `Registration` session, which results in a more powerful adversary in comparison to the *distance-fraud* adversary of Vaudenay *et al.* [23]. The second `DB` attack is a three-party attack in which an honest *prover* $\mathcal{P}$ is far-away from honest *verifier* ($\mathcal{V}$), but a malicious adversary $\mathcal{A}_2$ which is located within the distance $\mathbb{D}$, wants to convince $\mathcal{V}$ about the distance bound of $\mathcal{P}$.

Definition 3. β-resistance MiM: *For any PPT adversary $\mathcal{A}$, which can*

(i) initiate `Registration` *or* `DB` *session of an honest prover ($\mathcal{P}_i$),*
(ii) listen/block/change the communications of a `Registration` *session between an honest prover ($\mathcal{P}_i$) and $\mathcal{RA}$,*

(iii) listen the communications of polynomially bounded instances of `DB` *sessions between any honest verifier $\mathcal{V}_j$ and $\mathcal{P}_i$, when she* <u>*is located*</u> *within distance $\mathbb{D}$ to $\mathcal{V}_j$ (learning phase),*

(iv) listen/block/change the communications of polynomially bounded instances of `DB` *sessions between any honest verifier $\mathcal{V}_k$ and $\mathcal{P}_i$, when she* <u>*is not located*</u> *within distance $\mathbb{D}$ to $\mathcal{V}_k$,*

(v) run any polynomially bounded instances of algorithms with independent inputs from above,

the probability of returning $Out_{\mathcal{V}} = 1$ in any of `DB` *sessions with $\mathcal{V}_k$ is not more than β.*

Impersonation-Attack is an special case of this attack, in which there is no learning phase and no honest prover. If a protocol is secure against *MiM* adversary, then it is secure against *impersonation-attack* with at least the same probability. With the same argument as before (i.e. more freedom in `Registration` phase), this definition is stronger than the *MiM* adversary of Vaudenay *et al.* [23].

In the third attack, three parties are involved; a dishonest *prover* ($\mathcal{P}^*$) which is far away from the honest *verifier* ($\mathcal{V}_j$), and an *adversary* which is located within the distance to $\mathcal{V}_j$ and helps $\mathcal{P}^*$ to convince $\mathcal{V}_j$ about the distance bound of $\mathcal{P}^*$.

Definition 4. γ-resistance Terrorist-Fraud: *For any pair of PPT adversary $\mathcal{A}^{TF}$ and dishonest prover ($\mathcal{P}_i^*$), which is not located within distance $\mathbb{D}$ to an honest verifier ($\mathcal{V}_j$), when they have the following abilities;*

(i) $\mathcal{P}_i^*$ is able to run a `Registration` *session with $\mathcal{RA}$*

(ii) $\mathcal{P}_i^*$ can communicate with $\mathcal{A}^{TF}$ outside the `DB` *protocol, but is not willing to leak any information about her own secret key,*

(iii) $\mathcal{A}^{TF}$ can listen/block/change the communications of a `DB` *session between any honest verifier $\mathcal{V}_j$ and $\mathcal{P}_i^*$, and*

then the probability of convincing $\mathcal{V}_j$ to return $Out_{\mathcal{V}} = 1$ in the `DB` *session is not more than γ.*

We define **privacy** in terms of the distinguishability advantage of an adversary in a game with a challenger. The adversary chooses two provers $\mathcal{P}_0$ and $\mathcal{P}_1$, and gives them to the challenger. The challenger chooses a random bit $b \in_R \{0, 1\}$ and returns the anonymous handle of $\mathcal{P}_b$, to the adversary. The adversary can have access the orcales listed below, for any polynomial number of times, and then outputs a bit b'. Success probability of adversary is in terms of $\Pr(b = b')$. The formal definition is as follows:

Definition 5. $\rho-$Extensive Privacy: *Consider the following game between a challenger and adversary $\mathcal{A}_P$ who can make query to the following oracles:*

– $(\mathcal{P}_i) \leftarrow$ `CreateProver`()*; creates a prover with a unique identifier $\mathcal{P}_i$. This oracle creates the internal keys and certificates of a new prover and returns $\mathcal{P}_i$.*

- $(\mathcal{P}_i, stt_{\mathcal{P}}) \leftarrow$ `CreateInsider`(); *This oracle is same as* `CreateProver`, *but it also returns the internal state of the prover.*
- $(\pi, m) \leftarrow$ `Launch`(); *this oracle runs a pre-defined protocol on verifier and returns the session identifier* π *and the message sent by the verifier.*
- $(vtag) \leftarrow$ `DrawProver`$(\mathcal{P}_i, \mathcal{P}_j)$; *on input of two provers, this oracle returns a unique virtual fresh identifier to either* $\mathcal{P}_i$ *for* $b = 0$*, or* $\mathcal{P}_j$ *for* $b = 1$*. It first checks if any of them is an insider or already drawn, and terminates if so. Then asks from challenger to choose one bit* $b \in_R \{0, 1\}$*. Based on this bit, creates an anonymous handle vtag for the chosen prover and returns it. Note that in this step, there a private table* $\mathcal{T}$*, which stores the tuple* $(vtag, \mathcal{P}_i, \mathcal{P}_j, b)$*.*
- $(m') \leftarrow$ `SendProver`$(vtag, m)$; *on input of anonymous handle of a prover and a message* m*, this oracle sends the message* m *to the prover, and returns prover's reply message* (m')*.*
- $() \leftarrow$ `Free`$(vtag)$; *on input of anonymous handle of a prover, this oracle removes the handle, which eliminates any access to the prover through this handle. And by using the recorded tuple in the database, the related prover will get reset (i.e. erase volatile memory).*
- $(m') \leftarrow$ `SendVerifier`(π, m); *on input of a protocol session* (π) *and a message* m*, this oracle sends the message* m *to the verifier in the session* π*, and returns verifier's reply message* (m')*.*
- $(stt_{\mathcal{V}}) \leftarrow$ `StateVerifier`(); *this oracle returns the internal state of the verifier. This oracle includes the functionality of the* `Result` *oracle in* [22]*, which just returns the final output of session* π*.*
- $(stt_{\mathcal{RA}}) \leftarrow$ `StateRA`(); *this oracle returns the internal state of* $\mathcal{RA}$*.*

$\mathcal{A}_P$ *wins if she can find the choice of challenger in* `DrawProver` *oracle. A protocol is* ρ*-extensive private, if and only if there is no PPT adversary* $\mathcal{A}_P$ *who can win the game with advantage of more than* ρ*.*

Note 1. Extensive-privacy is stronger than *wide-privacy* [22] and the privacy model of Gambs *et al.* [11], in regards to the view of adversary about all *non-prover* entities. That's because the adversary has access to two more oracles `StateVerifier` and `StateRA` at the same time. This classification is about the view of adversary about all *non-prover* entities. The view of adversary about the provers have been considered independently based on her access to `Corrupt` oracle, which is not in the scope of this privacy model.

Finally, we define the security of `PDB`:

Definition 6. $(\alpha,\beta,\gamma,\rho)-$**secure Privacy-Preserving Distance-Bounding:** *A* `PDB` *protocol is defined by a tuple* $(KeyGen, reg_{\mathcal{P}}, reg_{\mathcal{RA}}, db_{\mathcal{P}}, db_{\mathcal{V}}, \mathbb{D})$*, as follows:*

1. $(pk_{\mathcal{RA}}, sk_{\mathcal{RA}}) \leftarrow$ `KeyGen`(1^{λ})*: A randomized algorithm such that on the input of the security parameter* λ*, returns a key pair to* $\mathcal{RA}$*.*
2. `reg`$_{\mathcal{P}}(pk_{\mathcal{RA}}; sk_i, \sigma_i) \leftrightarrow$ `reg`$_{\mathcal{RA}}(sk_{\mathcal{RA}}; \emptyset)$*: An interactive protocol between two PPT ITMs;* `reg`$_{\mathcal{P}}$ *returns a secret-key and signature of* $\mathcal{RA}$ *on it* $(sk_i, \sigma_i = Sign_{\mathcal{RA}}(sk_i))$ *by taking* $pk_{\mathcal{RA}}$ *as input, while* `reg`$_{\mathcal{RA}}$ *takes* $sk_{\mathcal{RA}}$ *as input.*

3. $\mathtt{db}_{\mathcal{P}}(sk_i, \sigma_i; \emptyset) \leftrightarrow \mathtt{db}_{\mathcal{V}}(pk_{\mathcal{RA}}; Out_{\mathcal{V}})$*: An interactive protocol between two PPT ITMs;* $\mathtt{db}_{\mathcal{V}}$ *returns a single bit* $Out_{\mathcal{V}}$*, by taking* $pk_{\mathcal{RA}}$ *as input, while* $\mathtt{db}_{\mathcal{P}}$ *takes prover's secret-key and* $\mathcal{RA}$*'s signature* $(sk_i, \sigma_i = Sign_{\mathcal{RA}}(sk_i))$ *as input.* $\mathtt{db}_{\mathcal{P}}$ *provides a commitment on* sk_i*, and then provides three proofs about the commitment; (i) proves that she knows the committed value* (sk_i)*, (ii) proves that she knows a valid signature of* $\mathcal{RA}$ *on the committed value* (σ_i)*, and (iii) proves that she (owner of* sk_i*) is located within the distance* $\mathbb{D}$*.* $\mathtt{db}_{\mathcal{V}}$ *returns* $Out_{\mathcal{V}} = 1$ *if the three proofs are correct, otherwise* $Out_{\mathcal{V}} = 0$*.*
4. $\mathbb{D}$ *is an integer indicating the distance bound.*

The protocol is secure, if the following properties hold; **Correctness** *(Definition 1),* α**-distance-fraud** *(Definition 2),* β**-MiM** *(Definition 3),* γ**-terrorist-fraud** (*Definition 4*), ρ**-extensive-privacy** (*Definition 5*).

4 PDB Construction

In this section we introduce our protocol, as an extension of `DBPK-Log` [5], by using *Pedersen commitment* [18], *zero-knowledge proof-of-knowledge protocols* [20] and *signature proof-of-knowledge protocols* [12]. The overview of the protocol is as follows:

1. `Setup`: $\mathcal{RA}$ creates a key-pair for signing secret-key of new *provers*. This operation is an instance of "$\mathtt{BBS}^{+}$`.KeyGen`" function.
2. `Registration`: A new *prover* $(\mathcal{P}_i)$ gets registered by $\mathcal{RA}$. $\mathcal{P}_i$ chooses a random secret-key (sk_i), and gets the signature of $\mathcal{RA}$ for it $(\sigma_i = Sign_{\mathcal{RA}}(sk_i))$ in a blind form. This operation is an instance of "$\mathtt{BBS}^{+}$`.Sign`" interactive protocol.
3. `Distance-Bounding`: A registered *prover* $(\mathcal{P}_i)$ provides distance-bounding proof to a *verifier* $(\mathcal{V}_j)$. $\mathcal{P}_i$ sends a Pedersen commitment $C = commit(sk_i)$ to $\mathcal{V}$, and then provides three proofs about the commitment; (i) proves that she knows the committed value (sk_i), (ii) proves that she knows a valid signature of $\mathcal{RA}$ on the committed value, by using "$\mathtt{BBS}^{+}$`.Verify`" non-interactive protocol, and (iii) proves that she is located within the distance bound $\mathbb{D}$. This proof is based on `fast-exchange` bitwise operation of every single bit of sk_i.

Global Common Parameters. By considering λ as the security parameter, let's define $(\mathbb{G}_1, \mathbb{G}_2)$ as a bilinear group pair with computable isomorphism ψ such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for a λ-bit strong prime p, such that $p = 2q + 1$ for a large prime number q. $\mathbb{G}_p$ is a group of order p, and the bilinearity mapping is $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_p$. $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $H_{evt} : \{0,1\}^* \rightarrow \mathbb{G}_p$ are hash functions. Let g_0, g_1, g_2 be generators of $\mathbb{G}_1$, h_0, h_1, h_2 be generators of $\mathbb{G}_2$ such that $\psi(h_i) = g_i$, and u_0, u_1, u_2 be generators of $\mathbb{G}_p$ such that the discrete logarithm of the generators are unknown. The generation of these parameters can be done by a trusted general manager, which is present just once at the begining. We use $\mathtt{BBS}^{+}$ signature scheme [6] with a block message of size one $(M = \{sk_i\})$, so obviously the parameters of $\mathtt{BBS}^{+}$ is included in these parameters. These values are public and considered as the input of all operations (omitted for simplicity).

The steps of the protocol are as follows:

1. Setup: $(sk_{\mathcal{RA}}, pk_{\mathcal{RA}} = h_0^{sk_{\mathcal{RA}}}) \leftarrow \texttt{BBS}^+.\texttt{KeyGen}(1^\lambda)$
2. Registration: $\mathcal{P}_\mathtt{i}(pk_{\mathcal{RA}}; sk_i, Sign_{\mathcal{RA}}(sk_i)) \leftrightarrow \mathcal{RA}(sk_{\mathcal{RA}}; \emptyset)$. They do the following steps:

– $\mathcal{P}_i$ randomly chooses an odd number $sk_i \in_R \mathbb{Z}_p \setminus \{q\}$.
– They execute $\texttt{BBS}^+.\texttt{Sign}\big[\mathcal{P}_\mathtt{i}(sk_i, pk_{\mathcal{RA}}; \sigma_i = Sign_{\mathcal{RA}}(sk_i)) \leftrightarrow \mathcal{RA}(sk_{\mathcal{RA}}; .)\big]$

3. Distance-Bounding: $\mathcal{P}_\mathtt{i}(sk_i, Sign_{\mathcal{RA}}(sk_i);) \leftrightarrow \mathcal{V}_\mathtt{j}(pk_{\mathcal{RA}}; Out_{\mathcal{V}})$. There are five steps in this phase; (i) `BBS`$^+$`.Verify`, (ii) `Bit Commitments`, (iii) `Fast-Exchange`, (iv) `Commitment Opening` and (v) `Proof-of-Knowledge`. At any step of this phase, $\mathcal{V}$ terminates with $Out_{\mathcal{V}} = 0$, if any failure happens, otherwise if it reaches the end, it returns $Out_{\mathcal{V}} = 1$ as the output.

(i) $\texttt{BBS}^+.\texttt{Verify}\big[\mathcal{P}_\mathtt{i}(sk_i, \sigma_i = Sign_{\mathcal{RA}}(sk_i); .) \leftrightarrow \mathcal{V}_\mathtt{j}(pk_{\mathcal{RA}}; Out, C = g_1^{sk_i}.g_2^r)\big]$. If $Out = 1$, then $\mathcal{V}_j$ continues to the next step and keeps the value of C.

(ii) In `Bit Commitment` step, the process of Fig. 1 gets executed. At the end of this phase, $\mathcal{V}_j$ is able to compute:

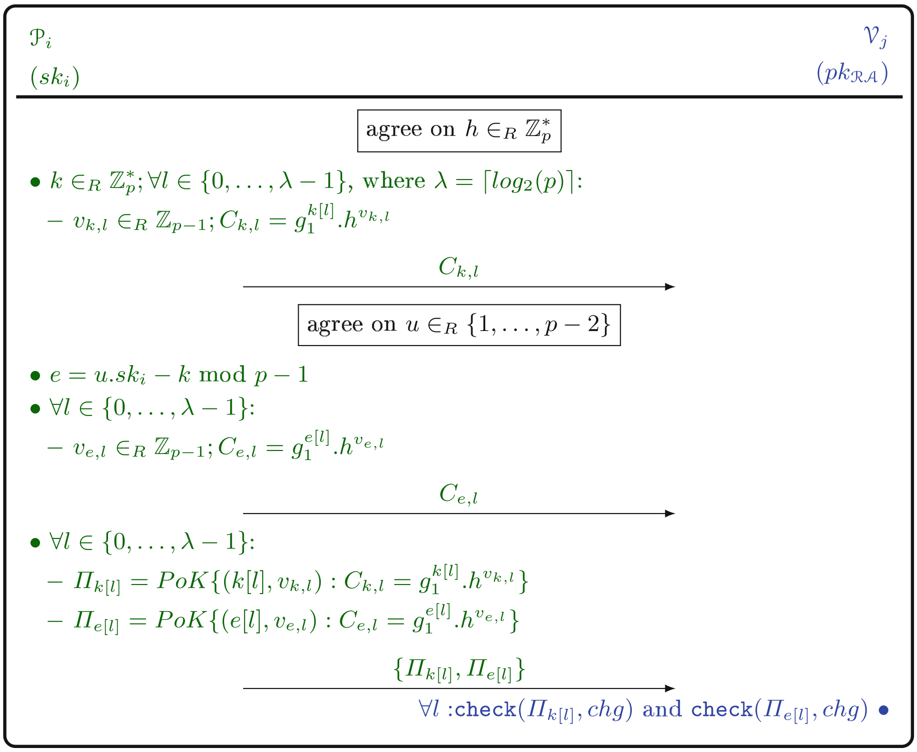
$\mathcal{P}_i$
(sk_i)
$\mathcal{V}_j$
$(pk_{\mathcal{RA}})$
agree on $h \in_R \mathbb{Z}_p^*$
• $k \in_R \mathbb{Z}_p^*; \forall l \in \{0,\dots,\lambda-1\}$, where $\lambda = \lceil log_2(p) \rceil$:
– $v_{k,l} \in_R \mathbb{Z}_{p-1}; C_{k,l} = g_1^{k[l]}.h^{v_{k,l}}$
$\xrightarrow{C_{k,l}}$
agree on $u \in_R \{1,\dots,p-2\}$
• $e = u.sk_i - k \bmod p-1$
• $\forall l \in \{0,\dots,\lambda-1\}$:
– $v_{e,l} \in_R \mathbb{Z}_{p-1}; C_{e,l} = g_1^{e[l]}.h^{v_{e,l}}$
$\xrightarrow{C_{e,l}}$
• $\forall l \in \{0,\dots,\lambda-1\}$:
– $\Pi_{k[l]} = PoK\{(k[l], v_{k,l}) : C_{k,l} = g_1^{k[l]}.h^{v_{k,l}}\}$
– $\Pi_{e[l]} = PoK\{(e[l], v_{e,l}) : C_{e,l} = g_1^{e[l]}.h^{v_{e,l}}\}$
$\xrightarrow{\{\Pi_{k[l]}, \Pi_{e[l]}\}}$
$\forall l$:check$(\Pi_{k[l]}, chg)$ and check$(\Pi_{e[l]}, chg)$ •

Fig. 1. `PDB`: `Bit Commitment` step

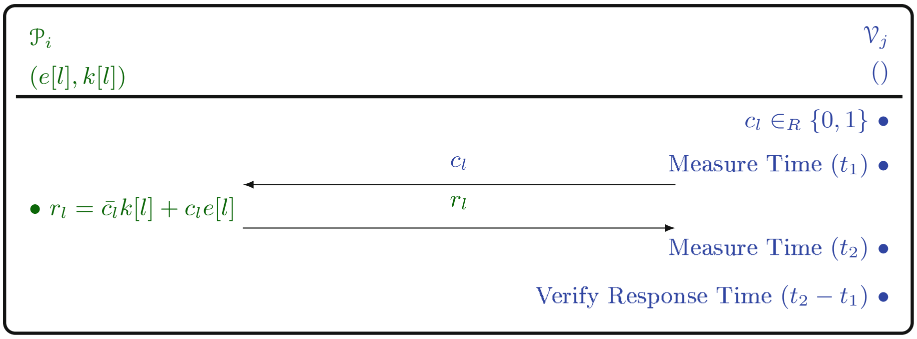

Fig. 2. `PDB`: `Fast-Exchange` step

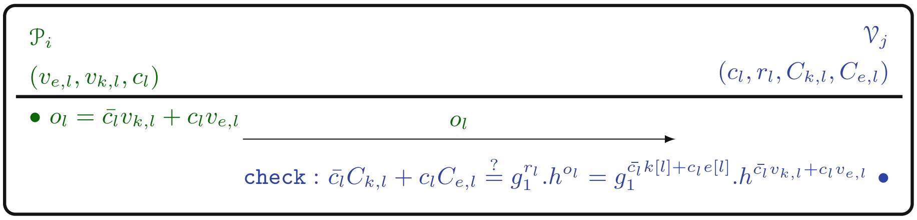
$\mathcal{P}_i$
$(v_{e,l}, v_{k,l}, c_l)$
$\mathcal{V}_j$
$(c_l, r_l, C_{k,l}, C_{e,l})$
• $o_l = \bar{c}_l v_{k,l} + c_l v_{e,l}$
o_l
check : $\bar{c}_l C_{k,l} + c_l C_{e,l} \overset{?}{=} g_1^{r_l}.h^{o_l} = g_1^{\bar{c}_l k[l] + c_l e[l]}.h^{\bar{c}_l v_{k,l} + c_l v_{e,l}}$ •

Fig. 3. `PDB`: `Commitment Opening` step

$$z = \prod_{l=0}^{\lambda-1}(C_{k,l}C_{e,l})^{2^l} = g_1^{\sum_{l=0}^{\lambda-1}(2^l.k[l]+2^l.k[l])}.h^{\sum_{l=0}^{\lambda-1}(2^l.(v_{k,l}+v_{e,l}))} = g_1^{k+e}.h^v = g_1^{u.sk_i}.h^v \bmod p$$

such that: $k = \sum_{l=0}^{\lambda-1}(2^l.k[l]) \ mod \ p-1$, $e = \sum_{l=0}^{\lambda-1}(2^l.e[l]) \ mod \ p-1$, $e = \sum_{l=0}^{\lambda-1}(2^l.e[l]) \ mod \ p-1$, $v = \sum_{l=0}^{\lambda-1}(2^l.(v_{k,l}+v_{e,l})) \ mod \ p-1$.

(iii) In `Fast-Exchange` step, process of Fig. 2 runs for $\forall l \in \{0, \ldots, \lambda-1\}$.

(iv) In `Commitment Opening` step, the process of Fig. 3 runs for $\forall l \in \{0, \ \ldots, \lambda-1\}$.

(v) Finally in `Proof-of-Knowledge` step, and interactive instance of $PoK[(sk_i, v, r) : z = g_1^{u.sk_i}.h^v \wedge C = g_1^{sk_i}.g_2^r]$ takes place to make sure that the summation of the secret values k and e is equal to randomized form of the committed secret-key $(u.sk_i)$. One possible way of doing the `PoK` is repeating the process of Fig. 4 for t times. The process continues, unless occurance of failure.

If all t times verifications succeed, then $\mathcal{V}_j$ returns $Out_{\mathcal{V}} = 1$. Note that if we replace the secret-key commitments $(C = g_1^{sk_i}.g_2^r)$ with public-keys of the prover $(pk_i = g_1^{sk_i})$, and remove the `BBS`$^+$ signature scheme, then we would have a secure public-key distance-bounding protocol (`DBPoK-log`$^+$), which fixes the vulnerabilities of `DBPK-Log` [5]. Now we provide our claim about the security of `PDB` protocol.

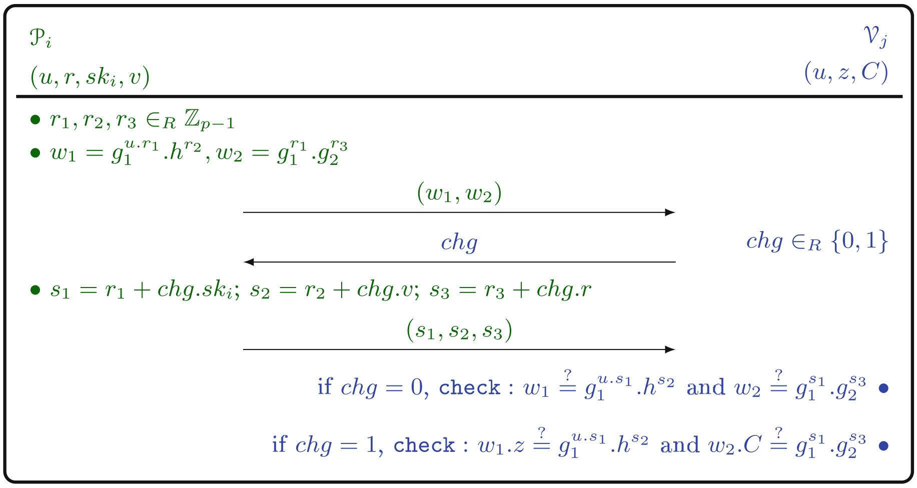
$\mathcal{P}_i$
(u, r, sk_i, v)
$\mathcal{V}_j$
(u, z, C)
• $r_1, r_2, r_3 \in_R \mathbb{Z}_{p-1}$
• $w_1 = g_1^{u.r_1}.h^{r_2}, w_2 = g_1^{r_1}.g_2^{r_3}$
(w_1, w_2)
chg
$chg \in_R \{0, 1\}$
• $s_1 = r_1 + chg.sk_i$; $s_2 = r_2 + chg.v$; $s_3 = r_3 + chg.r$
(s_1, s_2, s_3)
if $chg = 0$, check : $w_1 \overset{?}{=} g_1^{u.s_1}.h^{s_2}$ and $w_2 \overset{?}{=} g_1^{s_1}.g_2^{s_3}$ •
if $chg = 1$, check : $w_1.z \overset{?}{=} g_1^{u.s_1}.h^{s_2}$ and $w_2.C \overset{?}{=} g_1^{s_1}.g_2^{s_3}$ •

Fig. 4. `PDB`: `Proof-of-Knowledge` step

Theorem 1. `PDB` *protocol is* $(negl(\lambda), negl(\lambda), negl(\lambda), negl(\lambda))$*-secure, under Definition* 6*.*

The security proof of this theorem will be provided in the full version paper.

5 Conclusion

In this paper we solved the open problem of having a public-key `DB` protocol, which is secure against all `DB` adversaries by proposing a new protocol (`DBPoK-log`$^+$). This protocol is based on `DBPoK-Log` protocol which has shown to be vulnerable against DFA and TFA. We achieved security by adding some `PoK` operations. The computational cost of this achievement is equivalent to about $4.\lambda$ exponentiations in a prime order cyclic group per each `DB` instance.

Moreover we proposed a new privacy model for the provers against *dishonest* verifiers and *honest-but-curious* registration authority. And finally extended `DBPoK-log`$^+$ protocol to build a privacy-preserving `DB` protocol (`PDB`) in the new privacy model. This protocol, inherits *distance-bounding* properties from `DBPoK-log`$^+$ protocol. We replaced the public-key setting of provers, with Pedersen commitments and adopted `BBS`$^+$ signature scheme to provide *privacy* and *authentication* at the same time. As a result, `PDB` provides all three properties together; *distance-bounding*, *privacy* and *authentication*. The computational cost of this achievement, is about 25 extra exponentiations per each `DB` instance, in comparison with `DBPoK-log`$^+$.

There are still two open problems in this field; (1) having a `DB` protocol secure against all `DB` adversaries, which supports *extensive* privacy against an adversary who has access to `corrupt` oracle. And (2) having the same `DB` protocol in the presence of dishonest registration authority.

References

1. Avoine, G., Bingöl, M.A., Kardaş, S., Lauradoux, C., Martin, B.: A framework for analyzing RFID distance bounding protocols. J. Comput. Secur. **19**, 289–317 (2011)
2. Bay, A., Boureanu, I., Mitrokotsa, A., Spulber, I., Vaudenay, S.: The bussard-bagga and other distance-bounding protocols under attacks. In: Kutyłowski, M., Yung, M. (eds.) Inscrypt 2012. LNCS, vol. 7763, pp. 371–391. Springer, Heidelberg (2013)
3. Boureanu, I., Mitrokotsa, A., Vaudenay, S.: Secure & lightweight distance-bounding. In: Second International Workshop on Lightweight Cryptography for Security and Privacy (2013)
4. Brands, S., Chaum, D.: Distance bounding protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
5. Bussard, L., Bagga, W.: Distance-bounding proof of knowledge protocols to avoid terrorist fraud attacks. Technical report, Institut Eurecom, France (2004)
6. Camenisch, J.L., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
7. Cremers, C., Rasmussen, K.B., Schmidt, B., Capkun, S.: Distance hijacking attacks on distance bounding protocols. In: Security and Privacy, pp. 113–127 (2012)
8. Desmedt, Y.: Major security problems with the "unforgeable" (feige-)fiat-shamir proofs of identity and how to overcome them. In: Congress on Computer and Communication Security and Protection Securicom 1988, pp. 147–159 (1988)
9. Dürholz, U., Fischlin, M., Kasper, M., Onete, C.: A formal approach to distance-bounding rfid protocols. In: Lai, X., Zhou, J., Li, H. (eds.) ISC 2011. LNCS, vol. 7001, pp. 47–62. Springer, Heidelberg (2011)
10. Fischlin, M., Onete, C.: Terrorism in distance bounding: modeling terrorist-fraud resistance. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 414–431. Springer, Heidelberg (2013)
11. Gambs, S., Onete, C., Robert, J.M.: Prover anonymous and deniable distance-bounding authentication. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, pp. 501–506 (2014)
12. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**, 281–308 (1988)
13. Hermans, J., Pashalidis, A., Vercauteren, F., Preneel, B.: A new RFID privacy model. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 568–587. Springer, Heidelberg (2011)
14. Hermans, J., Peeters, R., Onete, C.: Efficient, secure, private distance bounding without key updates. In: Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks, pp. 207–218. ACM (2013)
15. Krumm, J.: A survey of computational location privacy. Pers. Ubiquitous Comput. **13**, 391–399 (2009)
16. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems (extended abstract). In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, p. 184. Springer, Heidelberg (2000)
17. Paise, R.I., Vaudenay, S.: Mutual authentication in RFID: security and privacy. In: Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security. pp. 292–299 (2008)
18. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)

19. Peeters, R., Hermans, J.: Wide strong private RFID identification based on zero-knowledge. IACR Cryptol. ePrint Arch. **2012**, 389 (2012)
20. Pieprzyk, J., Hardjono, T., Seberry, J.: Fundamentals of Computer Security. Springer, Heidelberg (2003)
21. Shokri, R., Theodorakopoulos, G., Le Boudec, J.Y., Hubaux, J.P.: Quantifying location privacy. In: 2011 IEEE Symposium on Security and Privacy (SP), pp. 247–262 (2011)
22. Vaudenay, S.: On privacy models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)
23. Vaudenay, S., Boureanu, I., Mitrokotsa, A., et al.: Practical & provably secure distance-bounding. In: The 16th Information Security Conference (2013)