Anonymous Distance-Bounding Identification

Ahmad Ahmadi, Reihaneh Safavi-Naini, and Mamunur Akand

University of Calgary, Canada
{ahmadi, rei, mdmamunurrashid.akan}@ucalgary.ca

Abstract. Anonymous Distance-Bounding (DB) protocols allow a prover to convince a verifier that they are within a distance bound from them, without revealing their identity. This is an attractive property that enables the prover to enjoy proximity based services, while their privacy is maintained. Combination of anonymity and distance-bounding however introduces new security challenges. We consider two new realistic attacks: a physical layer attack that uses *directional antenna*, and a *collusion attack* that involves multiple users. We show all existing anonymous DB protocols become insecure against at least one of these attacks, and then propose a new security model that captures these new attacks, and finally construct two protocols with provable security in this model. Our protocols are the only known anonymous DB protocols with provable security against known attacks.

1 Introduction

Distance upper bounding (DB) protocols were first proposed in [20] to provide security against Man-in- the-middle (MiM) attack in authentication protocols, and later found wide applications in location and proximity based services [12,29,22,17]. Early DB protocols are symmetric key protocols where the prover and the verifier share a secret key.

More recently public key DB protocols have also been proposed [32] to alleviate the traceability of the prover by the verifier. In this paper we consider this latter type of protocols. In these protocols there are three types of *participants*: *provers* who are *registered user* of the system and have secret keys, an honest *verifier* who knows the public keys of the provers, and *actors* who are not unregistered participants of the system, but would like to be accepted by the verifier, individually, or by helping a dishonest prover. A secure DB protocol estimates the distance between prover and verifier by measuring the round trip time of challenge and response bits that are exchanged between the two during the *fast challenge-response rounds*. The prover responds to a challenge immediately after it is received. The correct response to challenges in a protocol execution is stored in the *challenge-response table* that is calculated at the start of the protocol, and depends on the provers' secret key as well as nonces that are exchanged during the initialization phase.

In a DB protocol with distance bound $\mathcal{D}$, participants who are closer than $\mathcal{D}$ to the verifier are called $\mathcal{D}$ as *close-by* participants, and those who are farther than $\mathcal{D}$, are called as *far-away* participants. Widely considered attacks against public key DB protocols are;

(A1) *Distance-Fraud [13];* where a dishonest far-away prover tries to be accepted in the protocol. *Distance-Hijacking [17]* is a special case of this attack, where a far-away prover takes advantage of the communication of honest close-by provers to succeed in the protocol.
(A2) *Mafia-Fraud (MF) [20];* a close-by actor tries to use the communications of a far-away honest prover to succeed in the protocol. An special case of this attack, where the far-away prover is not active, is impersonation attack [7].
(A3) *Terrorist-Fraud (TF) [20];* a dishonest far-away prover colludes with a close-by actor, in order to succeed in the protocol. In *original TF-resistance* definition, it's assumed that the prover does not leak their secret key to the actor. In *recent TF-resistance* [31] however, the key leakage is allowed, but success of the TF attack requires negligible improvement in future impersonation attacks by the actor.

Security of public DB protocols has been formalized and protocols with provable properties [21,11] have been constructed.

Distance-bounding protocols leak the identity of prover to verifier: in symmetric key DB, prover and verifier share a secret key, and in public-key DB, prover's response is compared against the public-key of a specific user. Anonymous DB can be used to prove that the distance of a registered user is less than a prescribed bound, without revealing their exact identity. Security of anonymous DB protocols has also been formalized [3,14,8] against DF, MF and TF. In these model that we refer to as *single-user* model, attack involves at most a single corrupted user, possibly helped by an actor.

Our contributions. We introduce two new type of attacks that although applicable to all DB protocols, become particularly effective against anonymous DB protocols.

Directional antennas. The use of directional antennas in consumer devices has grown tremendously in recent years [1]. We consider the effect of employing these antennas by a malicious prover, on the security of anonymous DB. Note that verifiers need to use omni-directional antennas because they want estimate the distance of the prover without knowing their location. However malicious provers (or actors) may use directional antennas with a narrow beam to target messages to the verifier. In Section 3.1 we show that using directional antennas by malicious provers can break all existing anonymous DB protocols.

Collusion attacks. In a collusion attack multiple users, each with a secret key, participate in the attack that can be on DF, MF or TF form as shown in Figure 1, Figure 2 and Figure 3. These attacks had not been considered before and need not to be considered as long as the secret keys of two users are independently generated, and so (without anonymity) a protocol transcript can be linked to a user through their key information and so cannot be combined with other transcripts to form a new forged transcript. In anonymous DB protocols however, the verifier should not be able to link the transcript of a protocol to a single user and so combining protocol transcripts can give advantage to colluders. In Section 3.2 we show that collusion TF attack can be used to subvert traceability property of anonymous DB. This functionality is necessary in all anony-

mous DB protocols to ensure user accountability by allowing a third party that holds a master key, to "open" a transcript and identify the user, when required.

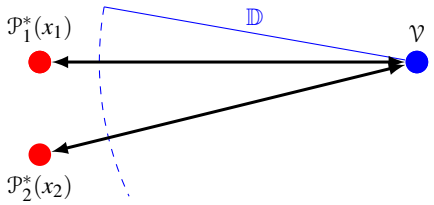

Fig. 1: Collusion DF

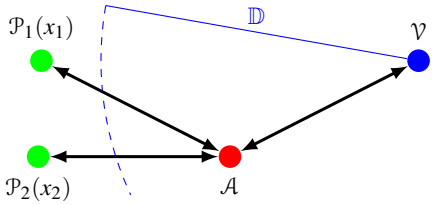

Fig. 2: Collusion MF

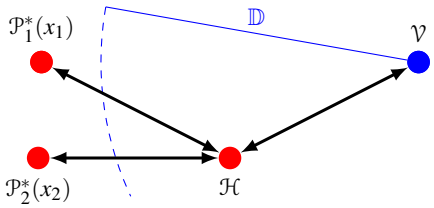

Fig. 3: Collusion TF

The above two classes of attacks are realistic. Directional antennas are widely used in modern communication systems [1] and there are strong incentive to launch collusion attacks, hiding behind anonymity that is offered by anonymous DB protocols. None of the existing anonymous DB protocols provide security against these attacks, and as shown later, there are concrete attacks against all existing protocols.

We will then show how collusion TF attack succeeds on a modified version of SPADE that is secure against a single-user non-directional TF attack. In this attack, a close-by user can interact with the verifier and get accepted, while credentials of a far-away user

is used. Thus the close-by user can be authenticated, and later during the opening phase a far-away user be identified. The system fails to provide security because the far-away user can present alibi that they have not been the protocol participant.

Model: We propose a formal model that captures the above two new classes of attacks. Our formalization uses a cryptographic approach and models an anonymous DB protocol as a cryptographic identification protocol [19] where the prover, in addition to proving their cryptographic credentials, prove that they are within a distance bound from the verifier. This builds on the model of public-key DB [2], by including directional antennas and collusion DF, MF and TF.

We formalize anonymity in terms of indistinguishability of candidate provers, given the protocol transcript. The challenge is to include sufficient information about the user in the transcript to allow a third party that holds the master key be able to open the transcript and identify the user.

Construction. We construct two anonymous DB protocol and prove their security in our proposed model. Our constructions can be seen as modular construction that adds anonymity and security in the new model by introducing an additional layer of group identification to a public-key DB protocol with provable security in a single-user model. The proposed protocols consist of a phase in which the prover commits to a temporary public-key, followed by a public-key DB. These protocols are designed for two different cryptosystems; Goldwasser-Micali cryptosystem [24] and Pedersen commitment [28].

The underlying public key DB protocols in our construction are `ProProx` [32], whose security in single-user `DBID` model was proven in [2], and `POXY` [4]. We reduce security of the new protocols in our proposed model (including directional antenna and collusion) to the security of the single-user model, and prove its anonymity and correctness of traceability.

This paper is the full version of conference paper [5].

Paper organization Section 2 includes preliminaries. Section 3.1 proposes new directional TF attack that breaks all anonymous DB protocols and Section 3.2 proposes collusion DB attacks, generalizing traditional DB attacks. Section 4 presents our model, Section 5 Section 6 give the constructions and security proofs. Section 7 gives a summary of related works, and Section 8 concludes the paper.

2 Preliminaries

In this section we introduce the primitives that we will later use in our model and constructions.

A **Σ-protocol** is a 3-message cryptographic protocol between a prover P and a verifier V, that allows P to prove validity of a statement to V. The two parties have a common input y, and P has a private input x for which the relation $\mathcal{R}(x,y)$ holds. `Σ-protocol` is used in many important cryptographic systems [18,23,27,26,30].

Definition 21 (Σ-protocol). *Prover P and verifier V execute three algorithms* (Commit, Response, Check) *using inputs* (x,y) *and* (y)*, respectively. x is private and y is public.*

Let $\mathbb{C}$*,* $\mathbb{H}$ *and* $\mathbb{R}$ *denote three sets:* $\mathbb{C}$ *is the set of possible input that is chosen by the prover;* $\mathbb{H}$ *is the set of possible challenges chosen by the verifier; and* $\mathbb{R}$ *is the set of possible responses of the prover. The steps of the protocol are as follows:*

1. *P randomly chooses* $a \in \mathbb{C}$ *and computes the commitment* $A = \texttt{Commit}(a)$*. P sends A to V.*
2. *Challenge/Response is a pair of messages:*
   (a) *V randomly chooses a challenge* $c \in \mathbb{H}$ *and sends it to P,*
   (b) *P computes* $r = \texttt{Response}(x,a,c,\hat{c}) \in \mathbb{R}$*, where* $\hat{c}$ *is the list of all challenges before c, and sends it to V,*

   Steps 2-(a) and 2-(b) may be repeated a number of times.
3. *V calculates* $ret = \texttt{Check}(y,[c],[r],A)$*, where* $ret \in \{accept, reject\}$ *and* $[c]$ *and* $[r]$ *are lists of all challenges and responses, respectively.*

At the end of the protocol, V outputs $Out_V = 1$ *if* $ret = accept$*, and* $Out_V = 0$ *if* $ret = reject$*.*

Here we define a more general form of Σ-protocols, called Σ^*-protocols, in which the verifier consecutively sends multiple challenges, each (except the first one) after receiving the response to the previous challenges.

Definition 22 (Σ^*-protocol). *A prover P and verifier V run the following*

Let $\mathbb{C}$*,* $\mathbb{H}$ *and* $\mathbb{R}$ *denote three sets defined as follows.* $\mathbb{C}$ *is the set of possible input that is chosen by the prover;* $\mathbb{H}$ *is the set of possible challenges chosen by the verifier; and* $\mathbb{R}$ *is the set of possible responses of the prover. The steps of the protocol are as follows:*

1. *P randomly chooses* $a \in \mathbb{C}$*, computes the commitment* $A = \texttt{Commit}(a)$*, and sends A to V.*
2. *Challenge and Response messages that are defined as follows:*
   (a) *V randomly chooses a challenge* $c \in \mathbb{H}$ *and sends it to P,*
   (b) *P computes* $r = \texttt{Response}(x,a,c,\bar{c}) \in \mathbb{R}$*, where* $\bar{c}$ *is the list of previous challenges before c, and sends it to V,*

   Steps 2-(a) and 2-(b) may be repeated a number of times.
3. *V calculates* $ret = \texttt{Check}(y,[c],[r],A)$*, where* $ret \in \{accept, reject\}$ *and* $[c]$ *and* $[r]$ *are lists of all challenges and responses, respectively.*

At the end of the protocol, V outputs $Out_V = 1$ *if* $ret = accept$*, and* $Out_V = 0$ *otherwise.*

In a cryptographic *identification scheme* (ID), a prover $\mathcal{P}$ convinces a verifier $\mathcal{V}$ that they know a witness x related to a public value y. Witness is a value that is in a certain relation with the public value, *i.e.*, $\mathcal{R}(x,y)$ holds for a defined relation $\mathcal{R}$.

The scheme is specified by the tuple $\texttt{ID=}(\texttt{KeyGen}, \Pi)$. The key generation algorithm $(x,y) \leftarrow \texttt{KetGen}(1^\lambda)$ is a PPT algorithm that takes the security parameter λ and generates a key pair (x,y). Π is a PPT protocol between the prover and the verifier. The prover $P(x,y)$ and the verifier $V(y)$ take the values of (x,y) and y respectively, as input. At the end of the protocol, the verifier returns either *accept* or *reject*. The protocol $\Pi = (\texttt{Commit}, \texttt{Response}, \texttt{Check})$ consists of two PPT algorithms `Commit` and `Response`, and a function `Check`, as defined in Definition 21.

An `ID` scheme is correct if the `Check` function outputs *accept* if $\mathcal{R}(x,y)$ holds, and *reject* otherwise. An `ID` scheme is secure if an adversary with access to a set of valid transcripts $\mathcal{T} = \{(A,[c],[r])\}$, cannot generate a valid transcript $(A',[c'],[r'])$ for a c' that has not appeared in $\mathcal{T}$. Note that a transcript $(A,[c],[r])$ is valid according to public-key y, if the function $\texttt{Check}(y,[c],[r],A)$ returns *accept*.

2.1 DBID

DBID model [2] is a security model for public-key DB protocols, that is based on cryptographic identification schemes.

Definition 23 (DBID). *Let $\lambda \in \mathbb{N}$ denote the security parameter. A distance-bounding identification (DBID) is a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathbb{P}, \mathcal{D}, p_{noise}, \texttt{Init}, \texttt{KeyGen}, \Pi, \texttt{Revoke})$, where;*

(I) $\mathbb{X}$ and $\mathbb{Y}$ are sets of possible master and public keys of the system, chosen based on the security parameter λ. The system master key $msk \in \mathbb{X}$, and public key $gpk \in \mathbb{Y}$ are generated using $(msk, gpk) \leftarrow \texttt{Init}(1^\lambda)$ algorithm;

(II) $\mathbb{S}$ and $\mathbb{P}$ are sets of possible (private, public) key pairs of users, with their size chosen according to the security parameter λ. A (private, public) key pair is generated using $(sk, pk) \leftarrow \texttt{KeyGen}(1^\lambda, msk, gpk)$ algorithm;

(III) Π is a Σ^*-protocol (Definition 22) between a prover $P(sk, pk)$ and the verifier $V(pk)$, that convinces the verifier that the prover is located within the distance bound $\mathcal{D} \in \mathbb{R}$ to the verifier.

(IV) The transmitted bits of a fast challenge-response round in Π protocol are affected by noise where $p_{noise} \in [0,1]$ is the probability of a bit flip on each fast challenge-response message.

(V) $\texttt{Revoke}(msk, gpk, i)$ is an algorithm that removes the corresponding user u_i from the system and updates the group public key accordingly.

At the end of the protocol Π, V outputs $Out_V = 1$ if they accept, or 0 if they reject.

In this model, the initialization (`Init`) and key generation (`KeyGen`) are run by a trusted party. The *distance bounding protocol* of a DBID scheme is denoted by $\texttt{DBID}.\Pi$, and in each run involves a single active user that is represented by multiple provers, sharing the same secret key. For honest users, only a single prover is active in a run. For corrupted users, no restriction on the number of active provers exists.

In our construction we consider DBID schemes for which public and private key of users are generated using Goldwasser-Micali (probabilistic) [24] encryption system and Pedersen [28] commitment scheme. We refer to DBID schemes with the property as these classes of DBID schemes as $\texttt{DBID}^{\texttt{GM}}$ and $\texttt{DBID}^{\texttt{P}}$, respectively. An example of $\texttt{DBID}^{\texttt{GM}}$ scheme is ProProx [32] and an example of $\texttt{DBID}^{\texttt{P}}$ scheme is POXY [4].

Security properties of DBID schemes are:

- **Completeness**: in the absence of an adversary, the verifier accepts an execution of Π with high probability when the prover is within the distance bound.
- **Soundness**: the success chance of a close-by adversary who is trying to take advantage of sessions of a far-away honest prover or a close-by inactive prover, is negligible.
- **DF resistance**: the verifier rejects an execution of Π with high probability if there is no close-by prover.
- **TF resistance**: if a dishonest far-away prover and a close-by helper succeeds in an execution of Π, then the helper can impersonate the prover in future Π executions with high probability.

We omit the formal definition of these properties (that appear in [2]) and present them in an expanded form in the formalization of anonymous DB in Section 4.

2.2 Camenisch-Lysyanskaya Signature Scheme

Camenisch and Lysyanskaya [16] proposed a special signature scheme (CLSig), that allows the users to commit to a message and then prove the knowledge of a signature on the message without leaking information about the message. This scheme is based on the standard definition of signature schemes due to *Goldwasser, Micali, and Rivest* [25].

Definition 24 **(CLSig).** *Let λ denote the security parameter.* CLSig *is a tuple* $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathbb{M}, \texttt{KeyGen}, \texttt{Sign}, \texttt{Verify}, \texttt{BSign}, \texttt{SPK})$*, where;*

(I) $\mathbb{X}$ *and* $\mathbb{Y}$ *are sets of possible private and public keys of the signature scheme, chosen based on the security parameter λ and the input parameter L. The private key $sk \in \mathbb{X}$, and public key $pk \in \mathbb{Y}$ are generated using* $(sk, pk) \leftarrow \texttt{KeyGen}(1^{\lambda}, L)$ *algorithm;*

(II) $\mathbb{M}$ *is the set of possible messages. A message is a string that represented as a string of integers $M = \{m_1, \ldots, m_L\}$, where $m_i \in \mathbb{P}$ can be chosen arbitrarily. The set $\mathbb{P}$ is chosen based on the security parameter λ. i.e, $\mathbb{M} = \mathbb{P}^L$;*

(III) $\mathbb{S}$ *is the set of possible signatures that is chosen based on the security parameter λ. A signature $\sigma \in \mathbb{S}$ is generated using* $\sigma \leftarrow \texttt{Sign}(1^{\lambda}, sk, M)$ *algorithm that is run by the signature authority who owns sk;*

(IV) $accept/reject \leftarrow \texttt{Verify}(M,\sigma,pk)$ *is a function that validates the generated signature* σ *according to the message M and the public key pk;*

(V) `BSign` *is a blind signature protocol between a prover* $P(M,pk)$ *and the signature authority* $A(sk)$*. The prover first commits to the message M, and then interact with the signature authority to generate a valid signature* σ *on the message M, that satisfies* $accept \leftarrow \texttt{Verify}(M,\sigma,pk)$*. At the end of the protocol* `BSign`*, both the prover and the signature authority output a valid signature* σ *on the message M, while the signature authority does not learn any information about M.*

(VI) `SPK` *is a protocol between a prover* $P(M,\sigma)$ *and the verifier* $V(pk)$*. The prover first commits to a message M and then convinces the verifier that the prover is in possession of the message M and a signature* σ *that satisfies* $accept \leftarrow \texttt{Verify}(M,\sigma,pk)$*, without leaking information about M or* σ*. At the end of the protocol* `SPK`*, V outputs the commitment of the message M and* $Out_V = 1$ *if they accept, or* $Out_V = 0$ *if they reject, while P outputs the randomness of the commitment.*

Security properties of `CLSig` can be informally stated as follows:

- **Completeness**; if the prover and the signature authority are honest in the protocol `BSign`, and then the prover and the verifier are honest in the protocol `SPK`, then the verifier outputs $Out_V = 1$ at the end of `SPK`.
- **Unforgeability**; even if an adversary has oracle access to the `Sign` algorithm to generate valid signature on any message that adversary chooses, the adversary cannot generate a valid signature on a message that is not explicitly queried.
- **Soundness**; if an adversary doesn't have access to any valid pair of message and signature (M,σ), then the verifier rejects execution of `SPK` with high probability.
- **Zero-Knowledge**; if the prover is honest in the `BSign` protocol then the signature authority learns no information about the message M, and if the prover is honest in the `SPK` protocol then the verifier learns no information about the pair (M,σ).

We formally define *zero-knowledge* property for a general two party protocol, but omit the formal definition of the other properties.

Definition 25 (Zero-Knowledge Protocol [32]). *A protocol between a pair of ITMs* $(P(\alpha),V(z))$ *is* ζ*-zero-knowledge for* $P(\alpha)$*, if for any PPT interactive machine* $V^*(z,aux)$ *there is a PPT simulator* $S(z,aux)$ *such that for any PPT distinguisher* $\mathcal{A}$*, any* $(\alpha : z) \in L$*, and any* $aux \in \{0,1\}^*$*, the advantage of* $\mathcal{A}$ *between the final view of* V^* *in the interaction* $P(\alpha) \leftrightarrow V^*(z,aux)$*, and output of the simulator* $S(z,aux)$ *is bounded by* ζ*.*

In this paper we use the term *zero-knowledge* as the short form of ζ-*zero-knowledge* for negligible value of ζ.

`BBS`$^+$ [16] is a `CLSig` signature scheme that uses Pedersen commitment in the non-interactive `SPK` protocol, and provides *completeness*, *unforgeability*, *soundness* and *zero-knowledge* [6].

3 New attacks

We present two new classes of attacks and show their effectiveness on anonymous DB protocols.

3.1 Directional TF Attack on Anonymous DB

DB protocols consist of - *slow phases* that are used during protocol initialization, and the final verification, and a *fast challenge-response phase* that is used for time (and so distance) measurement. Using a directional antenna a malicious prover can target the messages of the two phases such that the initialization messages are only received by the verifier and not the helper. This strategy allows the prover to send the whole challenge and response table of a particular protocol run to the helper, and so take advantage of the location of the helper, without leaking their long term key and so succeed in TF attack. Note that the prover is not leaking its identity to the helper. Figure 4 shows a directional TF, where the helper $\mathcal{H}$ does not receive slow phase messages sent by a malicious prover P^* to V using a directional antenna (orange ribbon in Figure 4). Before the start of the fast-phase, P^* sends the fast challenge-response table to $\mathcal{H}$, making $\mathcal{H}$ in-charge of responding to the fast-phase challenges.

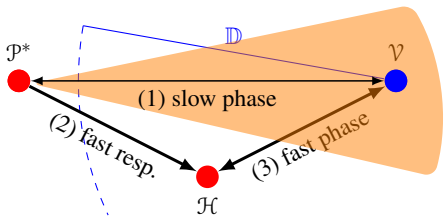

Fig. 4: Directional TF

In the following we describe how this setting helps a malicious prover to succeed in terrorist-fraud against all known anonymous DB protocols: PDB [3], SPADE [14] and TREAD [8].

Directional TF on PDB The presented model of [3] follows the original definition of TF (A3), and so our attack can be seen as outside their model. However, we showed that the original definition of TF is not suitable for anonymous distance-bounding protocols. In this section we present a TF attack against PDB, using the more recent definition of TF (A3).

The Π protocol in PDB scheme consists of the following five sub-protocols between the verifier (V) and the prover (P). At the end of each sub-protocol, the state of each of the two parties become the starting state of the next sub-protocol.

Step (i) SPK is a signature proof-of-knowledge protocol, in which the prover uses the secret key sk_i and a secret membership certificate (σ_i), and the verifier uses group public-key (gpk), as inputs. At the end of the protocol, the verifier will be convinced that the prover has a valid secret-key sk_i and membership certificate σ_i, in zero-knowledge (*i.e.*, the verifier doesn't learn anything about the prover's secrets). The verifier also learns the value of a commitment $C = g_1^{sk_i}.g_2^r$, that will be used in the last step. This protocol uses the BBS^+ signature scheme [16] for generating the membership certificate σ_i and the SPK protocol.

Step (ii) Bit Commitment is a commitment protocol, in which the prover uses the secret key sk_i, and the verifier uses the group public-key (gpk), as input. In this protocol, the prover decides on the "fast challenge-response table" and commits to each bit in the table. The verifier learns the committed values of every single bit of the fast challenge-response table. This table consists of two rows: $\{r_b[l]\}_{l=\{1,...,\lambda\},b\in\{0,1\}}$, where $r_b[l]$ is the response in the i^{th} fast challenge-response round. The corresponding committed values are $\{C_b[l]\}_{l=\{1,...,\lambda\},b=\{0,1\}}$, and the corresponding randomness of commitments are indicated by $\{v_b[l]\}_{l=\{1,...,\lambda\},b=\{0,1\}}$, where $v_b[l] \in \mathbb{Z}_p^*$. The commitment function is as follows: $C_b[l] = g_1^{r_b[l]}.h^{v_b[l]}$ for $b \in \{0,1\}$, $l = \{1...\lambda\}$, and $g_1, h \in \mathbb{Z}_p$. The committed table and the randomness is kept secret at the prover, while the commitments are sent to the verifier. Figure 5 shows the details of this step. The parts that are shown in a box, are sub-protocols whose details are omitted.

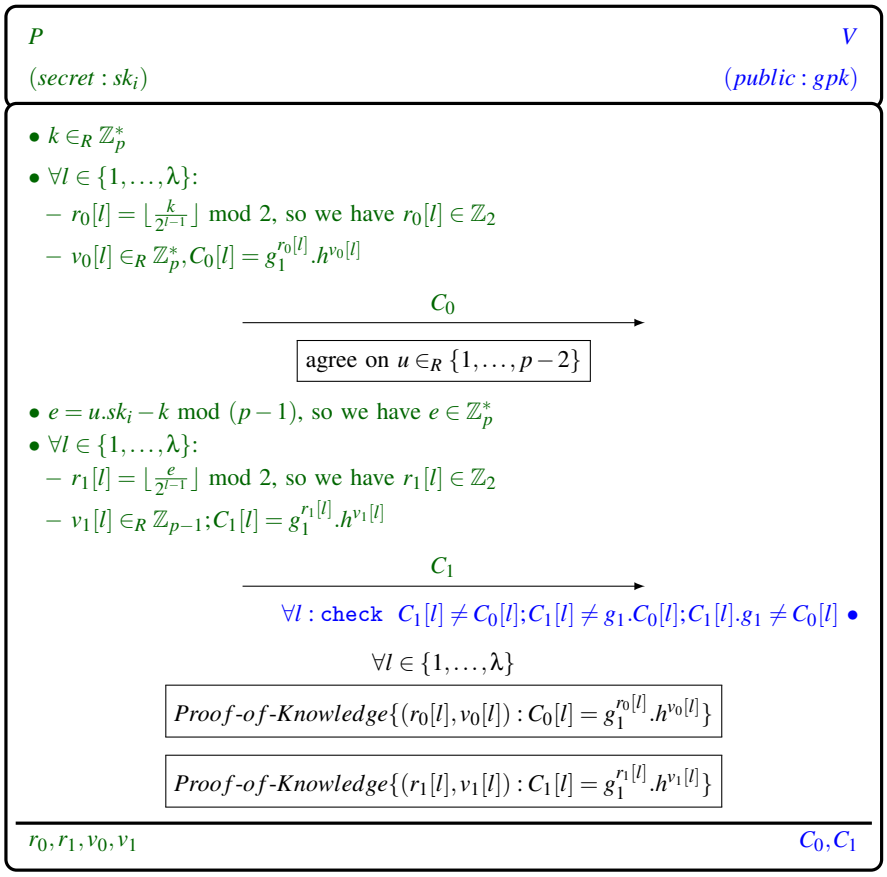

Fig. 5: PDB Step (ii): `Bit Commitment`

Step (iii) `Fast Challenge/Response` is the distance-bounding protocol, in which the prover uses the calculated "fast challenge-response table" $\{r_b[l] : l = \{1...\lambda\}, b = \{0,1\}\}$, generated in step (ii), as input. They run the protocol in Figure 6.

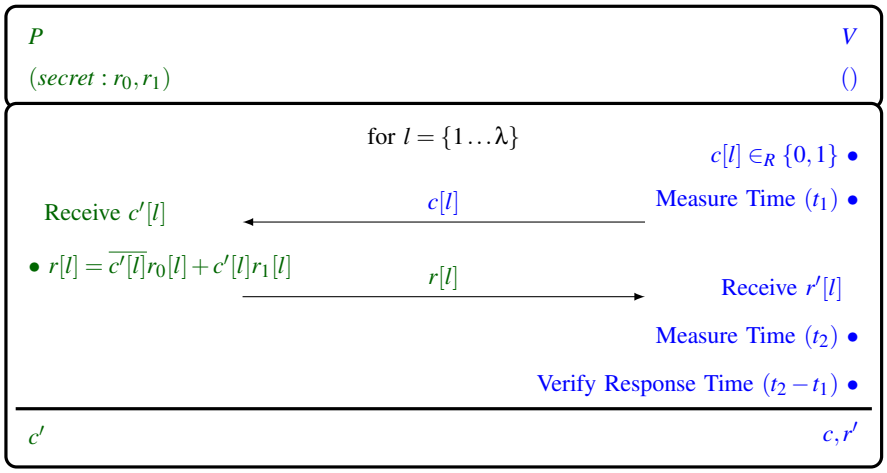

Fig. 6: PDB Step (iii): Fast Challenge/Response

Step (iv) Commitment Opening is used to open half of the commitments, that correspond to the challenge bits sent by the verifier in step (iii). In this step, the prover uses the secret commitment randomness (*i.e.*, $\{v_b[l] : l = \{1...\lambda\}, b = \{0,1\}\}$) and the challenge values of step (iii) (*i.e.*, c'), and the verifier uses the committed values (*i.e.*, $\{C_b[l] : l = \{1...\lambda\}, b = \{0,1\}\}$) and the challenge and response values of step (iii) (*i.e.*, c and r') as input. This protocol is shown in Figure 7, which improves the original PDB protocol [3] by adding noise resistance to the protocol. This step succeeds, if the noise counter is less than the threshold (*i.e.*, $count_{noise} < \tau$).

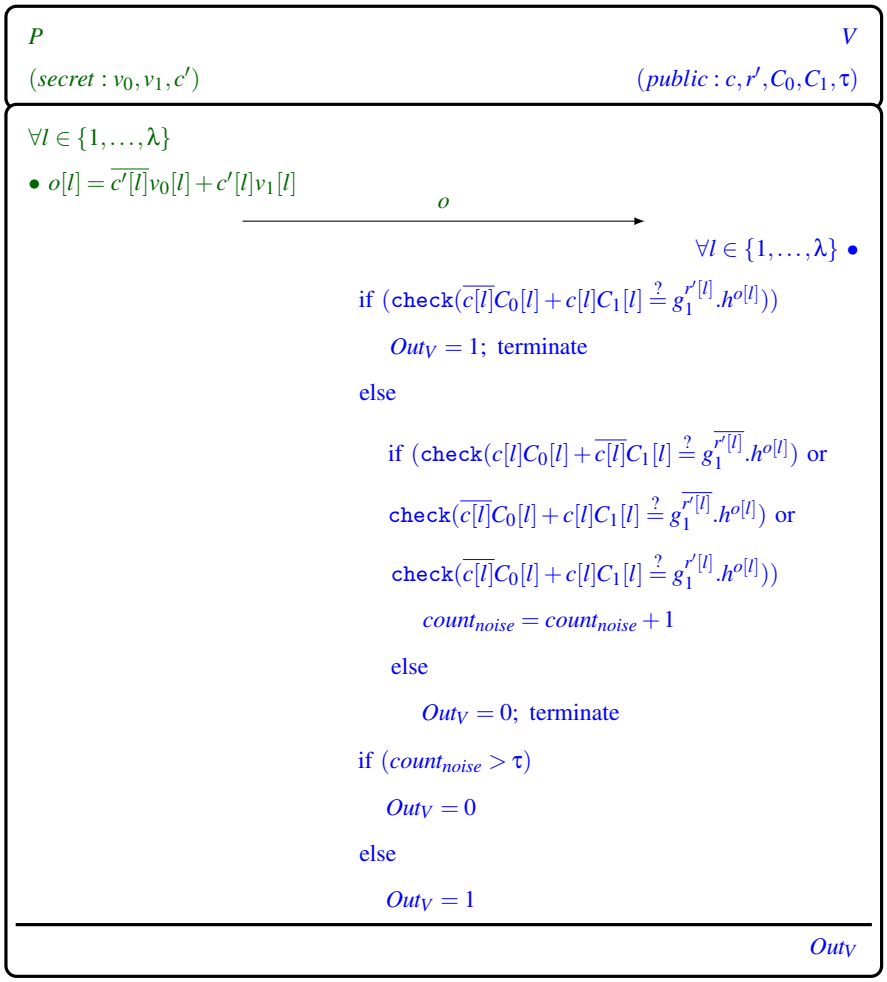

Fig. 7: `Commitment Opening` step of `PDB` protocol

Step (v) `Proof-of-Knowledge` is a protocol for zero-knowledge proof of equality that shows the secret key of step (i) and the committed secret key of step (ii) are the same. In this protocol, the prover uses the secret key sk_i and the commitment randomness of step (i) and step (ii), and the verifier uses the committed values of step (i) and step (ii), as input. C is the committed value of step (i), r is the commitment randomness of step (i), z is the accumulation of the committed values of step (ii) as $z = \prod_{l=1}^{\lambda} (C_0[l]C_1[l])^{2^{l-1}} \bmod p$, and v is the accumulation of the commitment randomness of step (ii) as $v = \sum_{l=1}^{\lambda} 2^l.(v_0[l] + v_1[l]) \bmod (p-1)$.

This protocol runs t iterations of ZK proof where z and C satisfy the following relation: $PoK[(sk_i, v, r) : z = g^{u.sk_i}.h^v \wedge C = g^{sk_i}.g_2^r]$.

If all five steps terminate successfully, then the verifier outputs $Out_V = 1$.

Lemma 1 *In the* Π *protocol of* PDB *scheme, the fast challenge-response table does not leak information about the membership certificate* σ_i *of the prover, unless negligible probability.*

Proof 1 *We know that by having the fast challenge-response table, we can calculate the secret-key of the prover, i.e.,* $sk = \frac{k+e}{u}$ *mod* $(p-1)$ *where* k *is fresh randomness. Note that the fast challenge-response table is the output of random function that takes sk as input. So it cannot leak any information about other independent secrets of the prover.*

A valid membership certificate σ *is the signature of registration authority on the secret-key sk. If there is an adversary* $\mathcal{A}$ *that can calculate the membership certificate from the secret-key sk, then* $\mathcal{A}$ *is a successful forgery adversary of the signature scheme. Therefore, since we assume the signature scheme is forgery resistant, then the success chance of* $\mathcal{A}$ *is negligible.* ■

Attack P sends to V the slow phase messages of step (i) and step (ii), using directional antenna. Before the start of the fast phase, P sends the fast challenge-response table, $\forall i \in \{1,...,\lambda\} : (r_0[i], r_1[i])$, to $\mathcal{H}$, allowing $\mathcal{H}$ to respond to V's challenges. P sends to V the slow phase messages of step (iv) and step (v), using directional antenna. In this way, the helper can respond in time and correctly to the challenges of the verifier during the fast challenge-response rounds. This attack makes the verifier to accept the protocol. Note that the fast challenge-response table does not leak the membership certificate σ_i, according to Lemma 1.

Since the helper has no information about σ_i, it cannot succeed in step (i) of future impersonation attacks. Therefore, it cannot impersonate the prove in future, which is required for a successful TF attack (See Property 44).

Directional TF on SPADE SPADE [14] is an anonymous DB system that use a group signature $GSign_{sk_p}()$ to register users in an authorized group. A registered user can use their credentials to participate in the protocol without leaking their identity, hence ensuring anonymity. Figure 8 presents the Π distance bounding protocol of SPADE scheme.

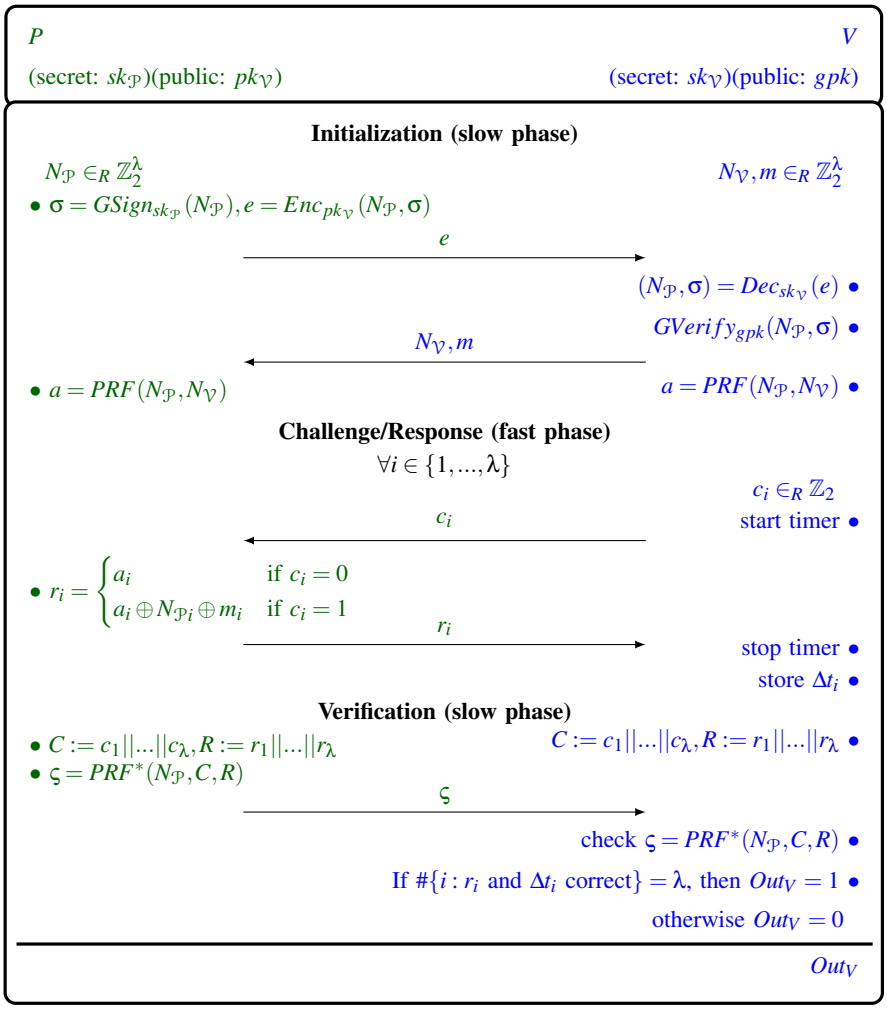

Fig. 8: Π protocol of SPADE scheme. $(GSign_{sk}, GVerify_{pk})$ is a group signature scheme. (Enc_{pk}, Dec_{sk}) is a secure public-key encryption scheme. $PRF : \mathbb{Z}_2^\lambda \times \mathbb{Z}_2^\lambda \to \mathbb{Z}_2^\lambda$ is a pseudo-random function. λ is the security parameter. $N_{\mathcal{P}}$ and $(N_{\mathcal{V}}, m)$ are nonce values of prover and verifier, (c_i, r_i) is a pair of challenge and response.

Lemma 2 *In the* Π *protocol of* SPADE *scheme, the fast challenge-response table does not leak information about the secret of prover.*

Proof 2 *The fast challenge-response table is* $r_i = \begin{cases} a_i \\ a_i \oplus N_{\mathcal{P}i} \oplus m_i \end{cases}$ *for* $i = \{1, ..., \lambda\}$. *In each instance of* Π *protocol,* $N_{\mathcal{P}}$ *and* m *are fresh and chosen randomly, and* $a =$

$PRF(N_{\mathcal{P}}, N_{\mathcal{V}})$ is the output of a pseudo-random function, fed by two fresh random values. Therefore, this table is independent from the secret value $sk_{\mathcal{P}}$. ■

Attack P sends to V the slow phase message e to V using directional antenna. Before the start of the fast phase, P sends the fast challenge-response table, $\forall i \in \{1,...,\lambda\}$: $(a_i, a_i \oplus N_{\mathcal{P}i} \oplus m_i)$, to $\mathcal{H}$, allowing $\mathcal{H}$ to respond to V's challenges. The collusion of P and $\mathcal{H}$ makes V to accept (*i.e.*, $Out_V = 1$) and this is without P sending to $\mathcal{H}$ any information that is dependent on the secret key of P (*i.e.*, $sk_{\mathcal{P}}$). Note that the secret key of P is required for generation of the message e which will not be known by $\mathcal{H}$.

According to Lemma 2, the fast challenge-response table does not leak any information about the prover's long-term secret $sk_{\mathcal{P}}$. Since the helper has no information about $sk_{\mathcal{P}}$, it cannot generate a valid message e in future, as the secret of prover is required to generate it. So its' success chance in a future impersonation attack will not be improved. This completes a successful TF attack (See Property 44).

Directional TF on `TREAD` `TREAD` [8] is an anonymous DB system that use a group signature $GSign_{sk}()$ to register users in an authorized group. The verifier needs to be registered and have a key-pair of their own. The structure of `TREAD` is very similar to `SPADE`. A registered user can use their credentials to participate in the protocol without leaking their identity, hence ensuring anonymity. Figure 9 presents the Π distance bounding protocol of `TREAD` scheme.

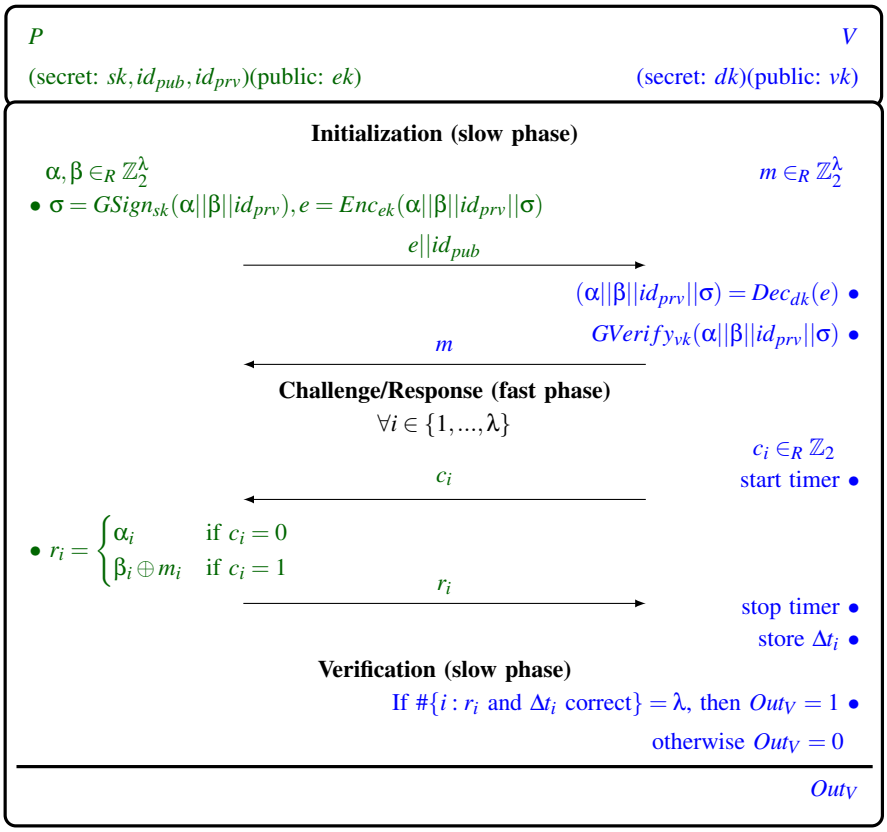

Fig. 9: Π protocol of TREAD scheme. $(GSign, GVerify)$ is a group signature scheme. (Enc, Dec) is a secure encryption scheme.

Lemma 3 *In the* Π *protocol of* TREAD *scheme, the fast challenge-response table leaks no information about the secret of prover.*

Proof 3 *The fast challenge-response table is* $r_i = \begin{cases} \alpha_i & \text{if } c_i = 0 \\ \beta_i \oplus m_i & \text{if } c_i = 1 \end{cases}$ *for* $i = \{1,...,\lambda\}$. *In each instance of* Π *protocol,* β, α *and* m *are fresh and chosen randomly. Therefore, this table is independent from the secret value* $sk_{\mathcal{P}}$. ■

Attack P sends to V the slow phase message $e||if_{pub}$ to V using directional antenna. Before the start of the fast phase, P sends the fast challenge-response table, $\forall i \in \{1,...,\lambda\}$: $(\alpha_i, \beta_i \oplus m_i)$, to $\mathcal{H}$, allowing $\mathcal{H}$ to respond to V's challenges. The collusion of P and $\mathcal{H}$ makes V to accept (*i.e.*, $Out_V = 1$) and this is without P sending to $\mathcal{H}$ any information that is dependent on the secret key of P (*i.e.*, sk). Note that the secret key of P is required for generation of the message e which will not be known by $\mathcal{H}$.

According to Lemma 3, the fast challenge-response table does not leak any information about the prover's long-term secret *sk*. Since the helper has no information about *sk*, its' success chance in a future impersonation attack will not be improved, as the secret of prover is required to generate a valid message *e*. So its' success chance in a future impersonation attack will not be improved. This completes a successful TF attack (See Property 44).

Concluding Remarks on Directional TF In all existing anonymous DB protocols, the fast challenge-response table does not determine the prover's credential with overwhelming probability. Directional TF attack allows the prover to limit the view of helper to the fast challenge-response table and so TF succeeds because the leaked information to the helper, does not allow the helper to succeed in a future attack individually, as required by the definition of TF attacks (see Property 44).

3.2 Collusion TF on Anonymous DB

In a traditional DB protocol attacks, collusion of at most a single registered user and an actor (non-registered user) is considered. We show that in anonymous DB protocols collusion of multiple registered users must be considered also.

We consider two types of *collusion TF* shown in Figure 10a and Figure 10b. In *collusion TF* type 1 attack, both colluding users are outside the bound and use a helper that is inside the bound. In *collusion TF* type 2, the helper can be a prover of a user, that tries to help the far-away provers of another user. Note that in type 2 attack there is a close-by prover $\mathcal{P}_2^*$ who can succeed in the protocol by themselves. However by colluding with $\mathcal{P}_1^*$, can succeed without being traced! (This attack also works in public-key DB protocols such as [15], where users choose their own private-keys, and so can collude and choose related keys that leads to the success of the above attack.)

(a) Collusion TF Type 1 (b) Collusion TF Type 2

Fig. 10: Collusion TF Attacks

Both collusions can be used to increase the success chance of the attacker. Here we show how Collusion TF Type 2 (Figure 10b) can break a protocol that is secure against TF in a single-user security model. As noted in Section 3.1, all existing anonymous DB protocols are vulnerable to single-user TF attack (directional TF) and so to show that

protection against single-user TF attack does not imply security against collusion TF attack, we first modify SPADE protocol to make it (intuitively) secure against single-user TF attacks (given in Section 3.1), and then describe how a multi-user collusion TF attack succeeds against the modified protocol.

SPADE* *(modified* SPADE*).* We modify the challenge-response table of the SPADE protocol to the following: $r_i = \begin{cases} a_i & \text{if } c_i = 0 \\ a_i \oplus x_i & \text{if } c_i = 1 \end{cases}$, where x is part of the prover secret-key that is chosen independent of $sk_{\mathcal{P}}$, and $|x| = \lambda$. The verification phase will also be revised to accommodate this change and allow the verifier can check if the correct parameters are used in the challenge-response table. The challenge-response table of SPADE* contains the secret-key of the prover, which makes the protocol intuitively secure against single-user TF attacks (let's assume that). This if the whole table is leaked to the helper, the helper can learn the secret key of the (malicious) prover by XOR-ing the two response bits of each challenge. Now we propose a collusion TF Type 2 (Figure 10b) against SPADE*;

Collusion TF Type 2 Attack: First, $\mathcal{P}_1^*(x_1)$ runs the "Initialization" phase of SPADE* with the verifier from outside the distance bound, and sends a to $\mathcal{P}_2^*(x_2)$. Then $\mathcal{P}_2^*(x_2)$ runs the challenge-response and verification phase with the verifier from inside the distance bound with its own credentials (x_2).

The intuition for the attack is that the challenge-response table is not linked to the long-term secret key of the user (group signature key). The verifier sees σ which is the group signature of the far-away prover $\mathcal{P}_1^*$, but runs the distance bounding phase using a key that is not related to group signature key. Thus the tracing authority will link the session to x_1, which is a violation of TF-resistance (Property 44).

4 Model

Firstly we define the settings of the system, *i.e.*, entities and how they communicate, protocol and view of an entity, adversary and their capability. Then we provide a definition of anonymous distance-bounding (AnonDB) and also describe AnonDB experiment, which captures an AnonDB scheme in execution. Finally, we formalize six security properties (*Completeness*, *MiM-resistance*, *DF-resistance*, *Soundness*, *Traceability*, *Anonymity*) of anonymous distance-bounding systems based on a game (AnonDB game), which is an AnonDB experiment played between a challenger and an adversary.

Entities. There are m users in the system $\mathcal{U} = \{u_1, \ldots, u_m\}$. Each user in the system can have multiple provers, which captures the practical scenario of a single person having multiple devices. We denote the list of provers for a user u_i as $\mathcal{P}^i$. Thus, we have m list of provers forming the prover set $\mathcal{P} = \{\mathcal{P}^1, \ldots, \mathcal{P}^m\}$.

A trusted group manger generates the public parameters of the system, registers users and issues a unique group membership certificate to each user. A user u_i ($1 \leq i \leq m$) is identifiable by their certificate. The certificate, that must be kept secret, forms the secret

input of the user in proving their membership in the group. The certificate of user u_i is shared by all provers of the list $\mathcal{P}^i$.

We define three types of *participants* in the system: provers ($\mathcal{P}$), verifiers ($\mathcal{V}$) (a singleton set), and actors ($\mathcal{T}$), called *helpers in TF attack*. $\mathcal{V}$ and $\mathcal{T}$ have access to only the public parameters of the system. Each participant has a location $loc = (x,y) \in \mathbb{R} \times \mathbb{R}$, that is an element of a metric space equipped with Euclidean distance, and is fixed during the protocol. The distance function $d(loc_1, loc_2)$ returns the distance between any two locations. Message travel time between locations loc_1 and loc_2 is $\frac{d(loc_1,loc_2)}{\mathcal{L}}$, where $\mathcal{L}$ is the speed of light. A bit sent over the channel may flip with probability p_{noise} ($0 \leq p_{noise} \leq 1$). Participants, if located within a predefined distance bound $\mathcal{D}$ from $\mathcal{V}$, are called *close-by* participants, otherwise they are called *far-away* participants.

Communication Structure. Each participant is equipped with a directional and an omni-directional antennas. Having directional antennas enables them to choose the angle of the transmission beam such that only the intended participants receive the message.

View. The view of an entity at any point (in time) of a protocol consists of: all the inputs of the entity (including random coin tosses) and the set of messages received by that entity up to that point. Any instance of receiving message is called an *event*. $View_x^{\Gamma}(e)$ is a random variable that denotes the view of an entity x right after the event e in protocol Γ. $View_x^{\Gamma}$ denotes the view of x at the end of the protocol Γ, *i.e.*, $View_x^{\Gamma} = View_x^{\Gamma}(e_{last})$ where e_{last} is the last event in Γ.

Adversary. An adversary can corrupt any subset of participants $\mathcal{X}^* \subset \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$. Corrupting one prover from a prover subset (*e.g.*, $x \in \mathcal{P}^j$) effectively corrupts the whole subset, since all members of that subset share the same certificate (of user u_j). Provers of uncorrupted subset follow the protocol, and only one prover from the subset executes the protocol at a time. We do not restrict provers of corrupted subset from doing this.

For each security property, the adversary has certain goals, which is reflected as restrictions of $\mathcal{X}^*$;

- in *Completeness* $\mathcal{X}^* = \emptyset$,
- in *Soundness* $\mathcal{X}^* \subseteq \mathcal{T}$,
- in *DF-resistance* $\mathcal{X}^* \subseteq \mathcal{P}$,
- in *TF-resistance* and *Traceability* $\mathcal{X}^* \subseteq \mathcal{P} \cup \mathcal{T}$, and
- in *Anonymity* $\mathcal{X}^* \subseteq \mathcal{V} \cup \mathcal{T}$.

Below we use the approach of [2] to define `AnonDB` scheme.

Definition 41 (Anonymous Distance-Bounding Scheme). *For a security parameter λ, an anonymous distance-bounding (`AnonDB`) scheme is defined by a tuple* $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathcal{D}, p_{noise}, \texttt{Init}, \texttt{CertGen}, \texttt{CertVer}, \Pi, \texttt{Revoke}, \texttt{Open})$*, where;*

(I) $\mathbb{X}$ and $\mathbb{Y}$ are sets of possible system master keys and group public-keys, respectively.

$\texttt{Init}(1^\lambda)$ *is the function that the group manager uses to generate the system master key msk, and the group public-key gpk.*

(II) $\mathbb{S}$ *is set of possible user membership certificates.* $\texttt{CertGen}(1^\lambda, msk, gpk, i)$ *function generates a user membership certificate* s_i*, and* $\texttt{CertVer}(s_i, gpk)$ *validates a user's certificate with respect to the group public-key.*

(III) Π *is a DB protocol between prover* $P(s_i, gpk)$ *and verifier* $V(gpk)$*, in which V verifies that a group member is located within the distance bound* $\mathcal{D}$ *to the verifier.*

(IV) The transmitted bits of a fast challenge-response round is affected by noise where $p_{noise} \in [0,1]$ *is the probability of a bit flip on each fast challenge-response message.*

(V) $\texttt{Revoke}(msk, gpk, i)$ *is an algorithm that removes a user* (u_i) *from the system and updates the master secret-key and group public key accordingly.*

(VI) $\texttt{Open}(msk, View_{\mathcal{V}}^{\Pi})$ *is an algorithm that identifies the user* (u_i) *that is involved in the* Π *protocol, using view of the verifier.*

The operations Open and Revoke are optional in an AnonDB scheme. Note that $I-V$ above is the same as $I-V$ in Definition 23, with the only difference that in DBID, each user owns a key-pair, while in AnonDB, each user owns a membership certificate that allows them to prove their membership according to the group public-key.

Adversary's capability is modeled as their access to queries presented to the challenger. The security properties of an anonymous DB protocol are based on a game (AnonDB Game) between a challenger and an adversary. Note that we allow provers to have access to directional antenna (to captures the directional attack introduced in Sec 3.1), and presence of multiple, possibly colluding users (with different secret keys) in the system (to capture multiple user collusion attack introduced in Sec 3.2).

Below we describe a general execution of an instance of the AnonDB scheme, which we call AnonDB experiment. And after that, we define AnonDB game as an special AnonDB experiment.

Definition 42 (AnonDB Experiment). *An* AnonDB *experiment is defined by a tuple* $(\texttt{AnonDB}; \mathcal{U}; \mathcal{P}; \mathcal{V}; \mathcal{T})$*, where*

(i) AnonDB *is an anonymous distance-bounding scheme as defined in Definition 41.*

(ii) $\mathcal{U}$ *is the set of users that are members of the group; each user* $u_j \in \mathcal{U}$ *has the following attributes:*

- u_j*.Cert that is a secret group membership certificate generated by the group manager,*
- u_j*.RT that is the registration time of the user that can be any time, and*
- u_j*.Rev that is a flag that shows if the user is revoked.*

(iii) $\mathcal{P}$ *is the set of provers; each prover has access to the membership certificate of a single user.*

(iv) $\mathcal{V}$ *is the set of verifiers; that have access to the group public-key of the* AnonDB *system. We consider the case where* $\mathcal{V}$ *has a single member.*

(v) $\mathcal{T}$ *is the set of actors; each actor has access to the group public key of the* AnonDB *system.*

Members of the set $\mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ *are called participants of the system. Each of the participants* $x \in \mathcal{X}$ *has the following attributes:*

a1. $x.Loc$ *is the location of the participant,*

a2. $x.Code$ *is the code to run by the participant,*

a3. $x.St$ *that is the start time of the* $x.Code$ *execution, and*

a4. $x.Corr$ *is a flag indicating if the participant is corrupted or not.*

In addition to these attributes, each prover $p \in \mathcal{P}$ *has one extra attribute:*

a5. $p.Key$ *that is the secret certificate of the corresponding user, i.e.,* $p.Key = u_j.Cert$ *for user* $u_j \in \mathcal{U}$.

All the provers that share the same certificate are called a prover subset, i.e., $\mathcal{P}^j = \{p : p \in \mathcal{P}, p.Key = u_j.Cert\}$. *The start time of all provers is after registration time of the corresponding user, i.e.,* $\forall u_j \in \mathcal{U}, \forall p \in \mathcal{P}^j : p.St > u_j.RT$.

Members of a prover subset are either all honest or all dishonest. i.e., $\forall \mathcal{P}^j \in \mathcal{P}, \forall p \in \mathcal{P}^j : p.Corr = flag$, *where* $flag \in \{true, flase\}$. *All members of an honest prover subset* $p \in \mathcal{P}^j$ *follow the* Π *protocol (i.e.,* $p.Code = \texttt{AnonDB}.\Pi.P(.)$*) and there is no overlap in the execution time of the members of an honest prover subset. If the verifier is honest, then it follows the* Π *protocol (i.e.,* $v.Code = \texttt{AnonDB}.\Pi.V(.)$ *for* $v \in \mathcal{V}$*).*

The experiment is run by a simulator that sets the attributes of the participants, and interacts with the group manager to assign keys to the provers of a user. If there is an adversary in the system, it interacts with the simulator to influence the experiment.

The experiment, without any adversary, proceeds as follows:

1. **Setup.**
   - *(a)* **Initialize:** *The group manager runs* $(\texttt{msk}/\texttt{gpk}) \leftarrow \texttt{AnonDB.Init}(1^\lambda)$ *algorithm to generate the master secret-key and group public-key.*
   - *(b)* **Generate Players:** *The simulator forms the sets* $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ *and sets their attributes. The simulator interacts with the group manager obtain and assign keys of the provers. It also simulates the behavior of malicious players by setting their code (*$x.Code$*).*
2. **Run:** *The simulator starts the execution of* $x.Code$ *for all participants* $x \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ *at time* $x.St$.

We assume the existence of a system clock that assigns time to events. The start and finish time of a protocol Γ is denoted as $stTime(\Gamma)$ and $fshTime(\Gamma)$ respectively, which form the execution time $exTime(\Gamma) = (stTime(\Gamma), fshTime(\Gamma))$ as a range of time and the execution length $exLen(\Gamma) = fshTime(\Gamma) - stTime(\Gamma)$. Members of a prover list ($\mathcal{P}^i$, $1 \leq i \leq m$) have different execution time period (*i.e.*, they participate in a protocol from time t_1 to t_2), and possibly different locations.

Table 1 is the summary of the notations used in this paper.

Symbol	Meaning
$\mathcal{V}$	set of verifiers with only one member
$\mathcal{U}$	set of users, each identifiable with a secret certificate
$\mathcal{P}$	set of provers
$\mathcal{P}^i$	subset of provers who have access to the certificate of i^{th} user in $\mathcal{U}$
$\mathcal{T}$	set of actors
$\mathcal{X}$	the set of all participants in an experiment
$\mathcal{X}^*$	the set of corrupted participants that are controlled by the adversary
Π	the distance bounding protocol between the prover and the verifier
$\Pi.V$	the interactive algorithm of the honest verifier in the Π protocol
$\Pi.P$	the interactive algorithm of the honest prover in the Π protocol
Out_V	output of the verifier a Π protocol
$\mathcal{D}$	the distance upper bound used in the protocol Π
p_{noise}	the bit flip probability of any message in the fast phase of the protocol Π
$stTime(\Gamma)$	the execution start time of the protocol Γ
$fshTime(\Gamma)$	the execution finish time of the protocol Γ
$exTime(\Gamma)$	the execution time of Γ, as a time range from $stTime(\Gamma)$ to $fshTime(\Gamma)$
$exLen(\Gamma)$	the execution time length of Γ, as $fshTime(\Gamma) - stTime(\Gamma)$
$View_x^\Gamma(e)$	the view of x right after the event e in the protocol Γ
$View_x^\Gamma$	short form of $View_x^\Gamma(e_{last})$ where e_{last} is the last event in the protocol Γ
$d(x,y)$	the distance between the two locations x and y

Table 1: Notations

In the following, we define a game between challenger and an adversary. This game is a limited AnonDB experiment that is run by the challenger who interacts with an adversary. In this game, the challenger plays both roles of the simulator and the group manager in the AnonDB experiment (Definition 42). The adversary's capabilities is modeled as access to a query that it presents to the challenger.

Definition 43 (AnonDB Game). *An* AnonDB *game between a challenger and adversary is an* AnonDB *experiment that is defined by a tuple* $(\texttt{AnonDB}; \mathcal{U}; \mathcal{P}; \mathcal{V}; \mathcal{T}; \texttt{CorruptParties})$ *where*

- AnonDB *is an anonymous distance-bounding scheme as defined in Definition 41.*

- $\mathcal{U}, \mathcal{P}, \mathcal{V}$ *and* $\mathcal{T}$ *are the sets of users, provers, verifiers and actors, as defined in Definition 42, that are determined through interaction of the challenger and the adversary.*
- $\texttt{CorruptParties}(Q)$ *is a query that allows the adversary to plan their attack.* Q *is a set of participants, that may exist in the system or be introduced by the adversary.*

In more details:

1. **Setup.**
   (a) **Initialize:** *Challenger runs* $(\texttt{msk}/\texttt{gpk}) \leftarrow \texttt{AnonDB.Init}(1^\lambda)$ *and publishes* gpk. *Note that the execution codes of honest prover and verifier are known by the challenger and the adversary at this point, and referred as* $\texttt{AnonDB}.\Pi.P$ *and* $\texttt{AnonDB}.\Pi.V$ *respectively.*
   (b) **Generate Players:** *The sets* $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ *are formed through the interaction of the challenger and the adversary:*
      i. *The challenger creates the sets* $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ *as follows:*
         - $\mathcal{V} = \{v\}$, *where:*
           a1. $v.Loc = loc_0$,
           a2. $v.Code = \texttt{AnonDB}.\Pi.V$,
           a3. $v.St = 0$, *and*
           a4. $v.Corr = false$.
         - $\mathcal{U} = \{u_j\}_{j=\{1,\ldots,m\}}$, *where* $u_j.Cert$ *is generated by* $\texttt{AnonDB.CertGen}(1^\lambda, msk, gpk, j)$ *function. The registration time of the users are set as* $u_j.RT = 0$ *and their revocation flag is set as* $u_j.Rev = false$.
         - $\mathcal{P} = \cup_{j=1}^{m} \mathcal{P}^j$, *where* $\mathcal{P}^j$ *is created as the prover subset of* $u_j \in \mathcal{U}$. *For all* $p \in \mathcal{P}^j_{\{j=1\ldots m\}}$ *assigns their attributes:*
           a1. $p.Loc$ *is set arbitrarily,*
           a2. $p.Code = \texttt{AnonDB}.\Pi.P$,
           a3. $p.St$ *is set arbitrarily such that there is no overlap in the execution time of the provers in* $\mathcal{P}^j$,
           a4. $p.Corr = false$, *and*
           a5. *secret-key* $p.Key = u_j.Cert$.
         - $\mathcal{T} = \emptyset$
      ii. *The challenger sends the attributes* $(x.Loc, x.Code, x.St)$ *for all* $x \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$, *along with all prover subsets* $\mathcal{P}^j \in \mathcal{P}$ *to the adversary. The size of the set* $\mathcal{X}$ *is n.*

iii. The adversary generates $\texttt{CorruptParties}(Q)$ *query and sends to the challenger. The challenger sends the secret information of the corrupted participants in* Q *to the adversary and their behavior (Code, Location and Start Time) is assigned according to the adversary instruction and their corruption flag is set to True. For all values of* $j = 1...m$*, if any prover* $p \in \mathcal{P}^j$ *gets corrupted, then all provers in* $\mathcal{P}^j$ *get corrupted too.*

iv. Upon receiving the $\texttt{CorruptParties}(Q)$ *where* $Q = \{q_1, ..., q_{n'}\}$*, the challenger runs:*

- *For a* q_i *that* $q_i.type = verifier$*, then* $v.Code = q_i.code$ *and* $v.Corr = true$ *for* $v \in \mathcal{V}$*.*
- *For each* q_i *that* $q_i.type = user$ *and* $q_i.usr \leq m$*, sets the users' revocation flag as* $u_j.Rev = true$ *where* $j = q_i.usr$*, runs* $(msk', gpk') \leftarrow \texttt{Revoke}(msk, gpk, q_i.usr)$*, then update the group master key* $msk \leftarrow msk'$ *and the group public key* $gpk \leftarrow gpk'$*. This applies only if the* `AnonDB` *scheme has user revocation.*
- *For each* q_i *that* $q_i.type = prover$*, find the prover subset* $\mathcal{P}^j$ *for* $j = q_i.usr$*. For each member* p *of subset* $\mathcal{P}^j$*, set their corruption flag* $p.Corr = true$*. If* q_i *is not corresponding to an existing prover, then create a new prover* p *and add it to the prover subset* $\mathcal{P}^j$*. Set the attributes of the participant* p *as follows:*

  a1. location $p.Loc = q_i.location$*,*

  a2. execution code $p.Code = q_i.code$*,*

  a3. start time $p.St = q_i.time$*,*

  a4. corruption flag $p.Corr = true$*, and*

  a5. secret-key $p.Key = u_j.Cert$*.*
- *For each* q_i *that* $q_i.type = actor$*, add a new actor* x *to the set* $\mathcal{T}$*, and assign its attributes as follows:*

  a1. location $x.Loc = q_i.location$*,*

  a2. execution code $x.Code = q_i.code$*,*

  a3. start time $x.St = q_i.time$*, and*

  a4. corruption flag $x.Corr = true$*.*

v. The challenger sends the key of the corrupted provers and the certificate of revoked users to the adversary, i.e., $p.Key$ *for all* $p \in \mathcal{P}$ *such that* $p.Corr = true$ *and* $u.Cert$ *for all* $u \in \mathcal{U}$ *such that* $u.Rev = true$*.*

2. **Run:** *Challenger activates all participants* $x \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ *at time* $x.St$ *for execution of* $x.Code$*.*

The game ends when the last participant's code completes its execution.

We define five properties for anonymous distance-bounding protocols based on AnonDB Game, conditions to win the game varies from one property to another.

Property 41 (AnonDB **Completeness).** *Consider an* AnonDB *scheme and an* AnonDB *game when $Q = \emptyset$ in the* CorruptParties(Q) *query and the set $\mathcal{P}$ is not empty.*

The AnonDB *scheme is (τ, δ)-complete for $0 \leq \tau, \delta \leq 1$, if the verifier returns $Out_V = 1$ with probability at least $1 - \delta$, under the following assumptions:*

- *the fast challenge-response rounds are independently affected by noise and at least τ portion of them are noiseless, and*
- $\tau > 1 - p_{noise} - \varepsilon$ *for some constant* $\varepsilon > 0$.

A complete protocol must have negligible δ to be able to function in the presence of communication noises.

Property 42 (AnonDB **Soundness).** *Consider an* AnonDB *scheme and an* AnonDB *game with the following restrictions:*

- $\forall p$ *in the nonempty set $\mathcal{P}$, and v as the only member of $\mathcal{V}$, we have $d(p.Loc, v.Loc) >$* AnonDB.$\mathcal{D}$, *and*
- *in the* CorruptParties(Q) *query we have $q_i.type = actor$ for all $q_i \in Q$.*

The AnonDB *scheme is γ-sound if the probability of the verifier outputting $Out_V = 1$ is at most γ.*

This general definition captures the following attacks by considering special values for game parameters;

- *relay attack* [13] happens when the MiM attacker only relays the messages between the honest verifier and a far-away honest prover. The MiM attacker tries to convince the verifier that the prover is located close to the verifier. This attack is achieved by adding extra restrictions on the adversary of Property 42 as follows:
  $\Rightarrow \forall q_i \in Q$ we have $q_i.code =$ "relay messages".
- *mafia-fraud* [20] is when there is an honest verifier, an honest far-away prover, and a close-by MiM attacker who tries to convince the verifier that the prover is located close to the verifier. The attacker listens to the legitimate communications for a while, before running the attack as the learning phase. This attack corresponds to adding extra restrictions on the adversary in Property 42 as follows:
  $\Rightarrow$ the set of provers consists of only one prover subset, *i.e.*, $\mathcal{P} = \mathcal{P}^1$, and
  $\Rightarrow \forall q_i \in Q$ we have $d(q_i.location, v.Loc) \leq$ AnonDB.$\mathcal{D}$ for $v \in \mathcal{V}$.
- *impersonation attack* [7] happens when there is an honest verifier and a single close-by attacker who tries to convince the verifier that the prover is located close to the verifier. The attacker can have a learning phase before running the attack. We can achieve this attack by adding extra restrictions on the adversary of Property 42 as follows:
  $\Rightarrow \mathcal{P}$ is nonempty, and

$\Rightarrow \forall q_i \in Q$ we have $d(q_i.location, v.Loc) \leq \texttt{AnonDB}.\mathcal{D}$ for $v \in \mathcal{V}$, and
$\Rightarrow$ among all the successful $\texttt{AnonDB}.\Pi$ protocols (Π^{succ} set) during the game, $\exists \pi \in \Pi^{succ}, \forall p \in \mathcal{P} : t = fshTime(\pi), t \notin [p.St, p.St + exLen(p.Code)]$.

- *multi-user MF*: there is an honest verifier, multiple honest far-away provers, and a close-by MiM attacker who tries to convince the verifier that one of the provers is located close to the verifier. The attacker can have a learning phase before running the attack. The extra restrictions on the adversary in Property 42 is as follows:
  $\Rightarrow$ the set of provers consists of a least two prover subsets, *i.e.*, $\exists p_1, p_2 \in \mathcal{P} : p_1.Key \neq p_2.Key$, and
  $\Rightarrow \forall q_i \in Q$ we have $d(q_i.location, v.Loc) \leq \texttt{AnonDB}.\mathcal{D}$ for $v \in \mathcal{V}$.

- *strong-impersonation* [2] happens when either *mafia-fraud* or *impersonation* happens. We can achieve this attack by adding extra restrictions on the adversary of Property 42 as follows:
  $\Rightarrow$ the set of provers consists of one prover subset, *i.e.*, $\mathcal{P} = \mathcal{P}^1$,
  $\Rightarrow \forall q_i \in Q$ we have $d(q_i.location, v.Loc) \leq \texttt{AnonDB}.\mathcal{D}$ for $v \in \mathcal{V}$, and
  $\Rightarrow$ among all the successful $\texttt{AnonDB}.\Pi$ protocols (Π^{succ} set) during the game, at least one of the following conditions hold:

  (i) $\exists \pi \in \Pi^{succ}, \forall p \in \mathcal{P} : t = fshTime(\pi), t \notin [p.St, p.St + exLen(p.Code)]$

  (ii) $\exists p \in \mathcal{P}, \exists \pi \in \Pi^{succ}, v \in \mathcal{V} : t = fshTime(\pi), t \in [p.St, p.St + exLen(p.Code)] \wedge d(p.Loc, v.Loc) > \texttt{AnonDB}.\mathcal{D}$.

We consider two types of attacks by a dishonest prover: multi-user far-away dishonest provers (Property 43), and multi-user far-away dishonest provers with close-by helpers (Property 44).

Property 43 (AnonDB **Distance-Fraud).** *Consider an* AnonDB *scheme and an* AnonDB *game with the following restrictions:*

- $\forall p$ *in the nonempty set* $\mathcal{P}$*, and* v *as the only member of* $\mathcal{V}$*, we have* $d(p.Loc, v.Loc) > \texttt{AnonDB}.\mathcal{D}$*, and*
- *in the* $\texttt{CorruptParties}(Q)$ *query,* $q_i.type = prover$ *and* $d(q_i.location, v.Loc) > \texttt{AnonDB}.\mathcal{D}$ *for all* $q_i \in Q$*.*

The AnonDB *scheme is* α*-DF-resistant if, for any* $\texttt{AnonDB}.\Pi$ *protocol in such game, we have* $\Pr[Out_V = 1] \leq \alpha$*.*

In the following we define the TF-resistance of anonymous DB protocols.

Property 44 (AnonDB **Terrorist-Fraud).** *Consider an* AnonDB *scheme and an* AnonDB *game with the following restrictions:*

- $\forall p$ *in the nonempty set* $\mathcal{P}$*, and* v *as the only member of* $\mathcal{V}$*, we have* $d(p.Loc, v.Loc) > \texttt{AnonDB}.\mathcal{D}$*,*
- *corrupted parties are either prover or actor* $\forall q_i \in Q : q_i.type \in \{prover, actor\}$*, and*

– *at least for one value of* $j \in \{1...m\}$ *we have* $d(q_i.location, v.Loc) > \texttt{AnonDB}.\mathcal{D}$ *for all* $q_i \in Q \cap \mathcal{P}^j$.

The AnonDB *scheme is* μ*-TF-resistant, if the following holds about the above game: if the verifier returns* $Out_V = 1$ *in the* Π *protocol of game* Γ *with non-negligible probability* κ *that is not traceable to any user with close-by provers (Property 46), then there is an impersonation attack (as an* AnonDB *game* Γ' *with honest verifier, no prover and one close-by actor) that takes the view of close-by participants of game* Γ *–excluding the verifier– as input, and makes the verifier return* $Out_V = 1$ *with probability at least* $\kappa - \mu$ *in the* Π *protocol of* Γ' *game.*

In this definition, any directional message that is sent to the verifier from outside the distance bound, is not included in the input of the impersonator. Therefore any protocol that is secure in this property, is also secure against directional TF attacks. Note that this definition captures collusion TF (Figure 10a and Figure 10b). In anonymous DB, breaking traceability is the only target of the adversary in collusion TF Type 2. Lemma 4 formally shows this claim.

The above attacks define security of the DB game. Now we define *anonymity* in terms of the distinguishing advantage of the adversary between two protocol sessions of two users.

Property 45 (AnonDB **Anonymity).** *Consider an* AnonDB *scheme and an* AnonDB *game with the following restrictions:*

– $\mathcal{P} = \{\mathcal{P}^1, \mathcal{P}^2\}$ *where the size of each of the sets* $\mathcal{P}^1$ *and* $\mathcal{P}^2$ *is equal to* $l > 0$*, and*

– *in the* CorruptParties(Q) *query,* $q_i.type \in \{verifier, actor\}$ *for all* $q_i \in Q$.

In this game, there are two subsets of honest provers of the same size, the adversary corrupts the verifier and adds a set of actors and sets their locations. Before activating the participants, the challenger randomly chooses $b \in_R \{0,1\}^l$*, and deactivates the* i^{th} *prover in* $\mathcal{P}^{b[i]}$*, i.e.,* $\forall 1 \leq i \leq l : \mathcal{P}_i^{b[i]}.Code = \varnothing$.

At the end of the game, $\mathcal{A}$ *returns* $b' \in \{0,1\}^l$*. A protocol is* α*-anonymous if for any such experiment, for all values of* $i \in \{1,...,l\}$ *we have* $|\Pr[b[i] = b'[i]] - \frac{1}{2}| \leq \alpha$.

We define *traceability* as a guarantee for the group manager to be able to identify the users from their protocol transcripts.

Property 46 (AnonDB **Traceability).** *Consider an* AnonDB *scheme and an* AnonDB *game with the following restrictions:*

– $\mathcal{P}$ *is nonempty, and*

– *in the* CorruptParties(Q) *query,* $q_i.type \in \{prover, actor\}$ *for all* $q_i \in Q$.

A protocol is called γ*-traceable, if the success chance of the* AnonDB.*Open algorithm in identifying a user that has a prover in* AnonDB.Π *protocol, from the transcript that is seen by the verifier, is a least as high as the chance of verifier outputting* $Out_V = 1$ *in the* AnonDB.Π *protocol plus* γ*. i.e.,* $\Pr[identify\ user] \geq \gamma + \Pr[Out_V = 1]$.

Note that an AnonDB game considers multiple honest users being active at the same time. Therefore, all properties are according to collusion scenarios.

Lemma 4 *An* AnonDB *scheme that is μ-TF-resistant according to Property44, is μ'-directional TF-resistant for negligible values of μ and μ'.*

Proof 4 *The main difference between directional antenna and omni-directional antenna, from information security perspective, is that omni-directional antenna allows the participants, within the communication range, to have similar view from the transmitted messages, while the directional antenna makes the view of those participant to be different.*

The rationality of Property 44 is that the higher the chance of future impersonation is, the scheme is more TF-resistant. So, the goal of a successful directional TF attack is to add the lowest amount of information to the view of the impersonation attacker.

In a TF attack (Property 44), all close-by participants, except the verifier, are controlled by the adversary. So, using any directional antenna to communicate with close-by participants that is not towards the verifier, is adding the transmitted message to the view of adversary. As a result, replacing that antenna with an omni-directional antenna does not reduce the knowledge of adversary, and so does not decrease its' chance in future impersonation. Therefore, we assume the only communications that are done by directional antenna, are those that are sent directly to the verifier.

Messages that are sent by directional antenna to the verifier, are not included in the view of impersonation adversary, i.e., $View_{\beta}^{\Gamma}$. Based on Property 44, if there is a TF attack against a scheme, the TF-resistant property guarantees the existence of impersonation attack by taking $View_{\beta}^{\Gamma}$ as input, which is the minimum view of the adversary from a directional TF attack. Therefore, in a TF-resistant AnonDB *scheme, a successful directional TF attack implies the existence of future impersonation attack.* ■

5 AnonDB Construction: dbid2an^{GM}

We refer to our AnonDB scheme as dbid2an^{GM} to emphasize conversion of a DBID scheme to an *anonymous* DBID. The DBID scheme has to use Goldwasser-Micali encryption system [24] for key generation. We first give an overview of our proposed scheme and then provide the details. In dbid2an^{GM}, a user is first enrolled in the system and is provided with a verifiable "membership" certificate. In addition to verifying the membership of a user, the certificate is used to generate a temporary public-key, which is later used in a public-key DBID protocol. At the end of a successful execution, the verifier is convinced that a valid member of the group is within the given distance bound.

Recall (Definition 41) that for a security parameter λ, an anonymous distance-bounding (AnonDB) scheme is defined by a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathcal{D}, p_{noise}, \mathtt{Init}, \mathtt{CertGen}, \mathtt{CertVer}, \Pi, \mathtt{Revoke}, \mathtt{Open})$. For our proposed (AnonDB) scheme dbid2an^{GM}, we name these oper-

ations as $\texttt{dbid2an}^{\texttt{GM}}$.Init, $\texttt{dbid2an}^{\texttt{GM}}$.CertGen, $\texttt{dbid2an}^{\texttt{GM}}$.CertVer, $\texttt{dbid2an}^{\texttt{GM}}.\Pi$, $\texttt{dbid2an}^{\texttt{GM}}$.Revoke and $\texttt{dbid2an}^{\texttt{GM}}$.Open.

In $\texttt{dbid2an}^{\texttt{GM}}$.CertGen, the group manager generates a membership certificate for a new user, and accumulates the certificates of all users to form a public commitment on them. Then the $\texttt{dbid2an}^{\texttt{GM}}.\Pi$ protocol takes place as below:

i) a prover of the user u_i, $i = 1..l$, anonymously proves that it owns one of the accumulated certificates (according to the public accumulated commitment).

ii) a temporary public-key is generated for the prover. The temporary public-key is generated using Goldwasser-Micali encryption, *i.e.*, we have $C[j] = Enc^{GM}(x_l[j], v_l[j])$ where for the $j = 1...\lambda$: $x_l[j]$ is certificate of the user, $v_l[j]$ is a random value chosen by the prover, and $C[j]$ is temporary public-key. In this paper we refer to $Enc^{GM}(.,.)$, as $\texttt{Commit}^{\texttt{GM}}(.,.)$ function. This temporary public-key generation is equivalent to the $\texttt{DBID}^{\texttt{GM}}$.KeyGen function. After establishing the temporary public-key, the prover and the verifier run a $\texttt{DBID}^{\texttt{GM}}.\Pi$ protocol, where the prover uses (x_l, v_l) as input, and the verifier uses C as input.

In our construction of $\texttt{dbid2an}^{\texttt{GM}}$, we use ProProx [32] as the $\texttt{DBID}^{\texttt{GM}}$ scheme, which is proven secure in the model of DBID schemes (directional antenna and single user attacks) [2].

In this scheme we use a hash function H making coins for $\texttt{Commit}^{\texttt{GM}}$, we define a deterministic commitment by $Com_{H_e}(x,v) = (\texttt{Commit}^{\texttt{GM}}(x_1, H(x,1).H(v,1)^e), ..., \texttt{Commit}^{\texttt{GM}}($ $x_\lambda, H(x,\lambda).H(v,\lambda)^e), \texttt{Commit}^{\texttt{GM}}(v_1, H(v,1)), ..., \texttt{Commit}^{\texttt{GM}}(v_\lambda, H(v,\lambda)))$ for $x, v \in \mathbb{Z}_2^\lambda$ and $\texttt{Commit}^{\texttt{GM}}(.,.)$ being Goldwasser-Micali encryption function. We assume $H(0,i) = 1$ for all values of i, and also assume that Com_H is a one-way function.

Now we provide the details of the operations:

$\texttt{dbid2an}^{\texttt{GM}}$.Init*:* $(msk, gpk) \leftarrow \texttt{Init}(1^\lambda)$. The group manager initiates the system as follows:

- Initialize Goldwasser-Micali cryptosystem: $(p,q,N,\theta) \leftarrow \texttt{DBID}^{\texttt{GM}}.\texttt{Init}(1^\lambda)$ for λ bit security choose $N = p.q$ and θ as a quadratic residue modulo N. Private: (p,q) and Public: (N,θ).
- Initialize RSA cryptosystem for the same N: generate (d,e) such that $gcd(e, \phi(N)) = 1$ and $d = e^{-1} (\bmod \phi(N))$. d is private and e is public.

The group master key is $msk = (p,q,d,U)$ where U is the list of all user private-keys, initialized to $U = \emptyset$. The group public-key is $gpk = (e,N,\theta,\hat{y},\tilde{y},\Xi)$ where $\hat{y}$ is commitment accumulation vector of user private-keys, $\tilde{y}$ is signature vector of group manager on $\hat{y}$ and Ξ is the list of all user membership signatures. These are initialized to $\hat{y} = \tilde{y} = [0]_\lambda$ and $\Xi = \emptyset$.

$\texttt{dbid2an}^{\texttt{GM}}$.CertGen*:* $(s, msk', gpk') \leftarrow \texttt{CertGen}(msk, gpk)$. The group manager first generates a certificate $s = (x_l, \sigma_l)$ and sends it to a new user (x_l is called user private-

key, and σ_l is called user membership signatures). And second, the system master key and public-key get updated accordingly, i.e., $msk \leftarrow msk'$ and $gpk \leftarrow gpk'$. The details is as follows, assuming $l-1$ users have already joined the group:

1. randomly choose $x_l \in \mathbb{Z}_2^\lambda$,
2. $y_l = Com_{H_e}(x_l, 0)$, which implies $\forall j \in \{1,\ldots,\lambda\} : y_l[j] = \texttt{Commit}^{\texttt{GM}}(x_l[j], H(x_l, j)) = \theta^{x_l[j]}.H(x_l, j)^2 \bmod N$ and $y_l[\lambda + j] = 1$. Sign $\sigma_l[j] = (y_l[j])^d$.
3. $\forall j \in \{1,\ldots,\lambda\}$:
   - accumulate j^{th} bit of all user private-keys into a single bit $\hat{x}[j] = x_1[j] \oplus \ldots \oplus x_l[j]$,
   - accumulate hash values $\hat{v}[j] = \prod_{1 \le i \le l} H(x_i, j)$, and
   - commit to accumulated values $\hat{y}[j] = \texttt{Commit}^{\texttt{GM}}(\hat{x}[j], \hat{v}[j]) = \theta^{\hat{x}[j]}\hat{v}[j]^2 \bmod N$.
4. Sign accumulated values $\tilde{y} = [\hat{y}[1]^{-d}, ..., \hat{y}[\lambda]^{-d}]$.

The updated group master key is $msk' = (p, q, d, U)$ where $U = \{x_1, ..., x_l\}$, and the updated group public-key is $gpk' = (e, N, \theta, \hat{y}, \tilde{y}, \Xi)$ where $\Xi = \{\sigma_1, ..., \sigma_l\}$. The certificate $s = (x_l, \sigma_l)$ is securely sent to the new user.

$\texttt{dbid2an}^{\texttt{GM}}.\texttt{CertVer}$*:* $accept/reject \leftarrow \texttt{CertVer}(s, gpk)$. Upon receiving a certificate $s = (x, \sigma)$, the user can check its validity. By reading the group public-key $gpk = (e, N, \theta, \hat{y}, \tilde{y}, \Xi)$, the user calculates $y = Com_{H_e}(x, 0)$ and checks $y[j] \stackrel{?}{=} (\sigma[j])^e \bmod N$, for $j = \{1...\lambda\}$.

$\texttt{dbid2an}^{\texttt{GM}}.\Pi$*:* $accept/reject \leftarrow \Pi\{P(s, gpk) \leftrightarrow V(gpk)\}$. When a prover ($\mathcal{P}_l$) of a registered user wants to run the $\texttt{AnonDB}.\Pi$ protocol with the verifier, they will follow the protocol described in Figure 11. The protocol consists of two main steps. The first step is a message from the prover to the verifier (y', π) that generates a temporary public-key (C), and then provides a non-interactive zero-knowledge (NIZK), which proves that the prover knows the privates related to the temporary public-key C. Note that in the non-interactive zero-knowledge proof, the verifier does not send any message to the prover [10,9]. The second step is running the $\texttt{DBID}^{\texttt{GM}}.\Pi$ protocol.

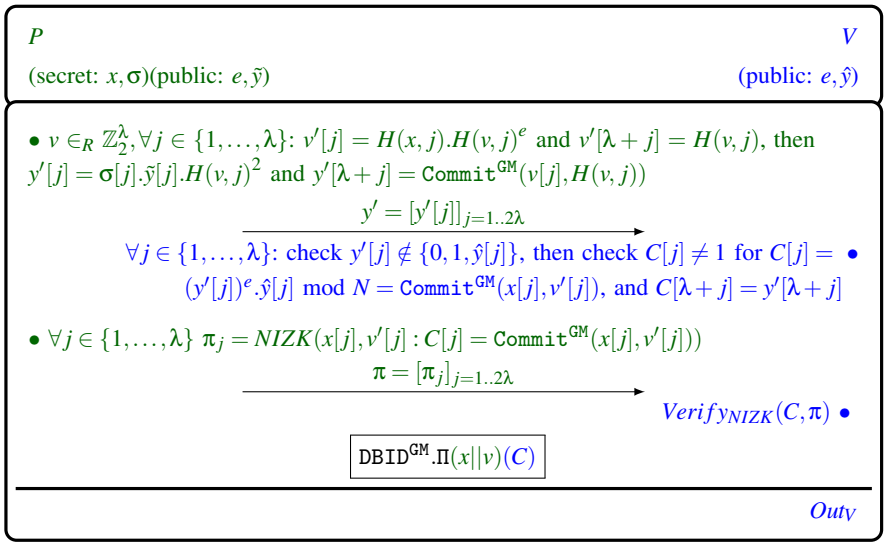

Fig. 11: Π protocol in $\mathtt{dbid2an}^{\mathtt{GM}}$ scheme for the l^{th} user. $C = Com_{H_e}(x,v)$.

$\mathtt{dbid2an}^{\mathtt{GM}}.\mathtt{Open}$: $(l) \leftarrow \mathtt{Open}(msk, transcript)$. The tracing authority who holds the group master key msk, uses the verifier's view of a successful run of Π with the prover $\mathcal{P}_l$, and returns index of the corresponding user in $\mathcal{U}$. The algorithm runs as follows, knowing that the group master key is $msk = (p,q,d,U = \{x_1,...,x_m\})$:

1. Determine inverse of 2 as $\hat{2} = 2^{-1}(\bmod \phi(N))$, *i.e.*, $\hat{2}$ is the multiplicative inverse of 2 $(\bmod \phi(N))$.
2. $\hat{y}^d = [\hat{y}[1]^d, ..., \hat{y}[\lambda]^d]$.
3. Parse verifier's view of the protocol to obtain y' and C.
4. Return the first $i \in \{1,\ldots,m\}$ that all the following holds:
   - $\forall j \in \{1,\ldots,\lambda\}$: $C[j] \stackrel{?}{=} \mathtt{Commit}^{\mathtt{GM}}(x_i[j], v[j])$ where $v'^2_j = y'[j].\hat{y}[j]^d.(y_i[j])^{-d}$, calculate $v'_j = (v'^2_j)^{\hat{2}} \bmod N$, and find $v[j] = H(x_i,j).v'^e_j$.

$\mathtt{dbid2an}^{\mathtt{GM}}.\mathtt{Revoke}$: $(msk', gpk') \leftarrow \mathtt{Revoke}(msk, gpk, l)$. In this operation, the entity holding the group master key msk, updates the group master key and the group public key such that the provers of l^{th} user ($l \in \{1...m\}$) cannot succeed in any Π protocol anymore. The algorithm runs as follows, knowing that the group master key is $msk = (p,q,d,U = \{x_1,...,x_m\})$ and the group public key is $gpk = (e,N,\theta,\hat{y},\tilde{y},\Xi)$ where $\Xi = \{\sigma_1,...,\sigma_m\}$;

1. $\forall j \in \{1,\ldots,\lambda\}$:
   - $\hat{x}[j] = x_1[j] \oplus \ldots x_{l-1}[j] \oplus x_{l+1}[j] \oplus \ldots \oplus x_m[j]$,

- $\hat{v}[j] = \prod_{i\in\{1,\ldots,l-1,l+1,\ldots m\}} H(x_i, j)$,
- $\hat{y'}[j] = \texttt{Commit}^{\texttt{GM}}(\hat{x}[j]; \hat{v}[j]) = \theta^{\hat{x}[j]}\hat{v}[j]^2 \bmod N$,
- $\tilde{y'}[j] = \hat{y'}[j]^{-d}$, and

2. $\Xi' = \Xi \setminus \{\sigma_l\}$.

After this operation, the group master key is $msk' = (p, q, d, U = \{x_1, \ldots, x_{l-1}, x_{l+1}, \ldots, x_m\})$ and the group public key is $gpk' = (e, N, \hat{y'}, \tilde{y'}, \Xi')$.

5.1 Security Analysis

In this section we provide the security analysis of $\texttt{dbid2an}^{\texttt{GM}}$ protocol, assuming that the adopted DBID protocol is secure.

Theorem 1 ($\texttt{dbid2an}^{\texttt{GM}}$ **Security Properties).** *If (i) the* $\texttt{DBID}^{\texttt{GM}}$ *scheme is* (τ, δ)*-complete,* γ'*-sound,* θ*-DF-resistant,* μ'*-TF-resistant and* $\texttt{DBID}^{\texttt{GM}}.\Pi$ *is zero-knowledge, and (ii) the temporary public-key (C) and the private key* (x_l, v_l) *of* $\texttt{DBID}^{\texttt{GM}}.\Pi$ *are related as* $C = Enc_N(x_l, v_l)$ *where* $Enc_N(.,.)$ *is the Goldwasser-Micali encryption algorithm for modulus N with* λ*-bit security,*

then $\texttt{dbid2an}^{\texttt{GM}}$ *is an* AnonDB *scheme that is* (τ, δ)*-complete (Prop41),* θ*-DF-resistant (Prop43),* γ*-Sound (Prop42),* μ*-TF-resistant (Prop44),* α*-anonymous (Prop45) and* γ*-traceable (Prop46), for negligible values of* α*,* δ*,* γ*,* γ'*,* μ*,* μ' *and* θ*, assuming that quadratic residuosity, factorization and RSA problems are hard problems.*

In the following, we prove each of the properties of the theorem in a separate lemma. We prove security properties of the protocol based on the model described in Section 4. The underlying $\texttt{DBID}^{\texttt{GM}}$ protocol provides single user directional antenna security [2]. The main challenge for the new model is to prove collusion security.

Lemma 5 (Completeness). $\texttt{dbid2an}^{\texttt{GM}}$ *is a* (τ, δ)*-complete* AnonDB *(Prop41) scheme, if the* DBID *scheme is* (τ, δ)*-complete.*

Proof 5 *Consider an* AnonDB *game with* $\texttt{dbid2an}^{\texttt{GM}}$ *scheme, in which the provers and the verifier are honest. In each* $\texttt{dbid2an}^{\texttt{GM}}.\Pi$ *protocol, the steps before the* $\texttt{DBID}.\Pi$ *protocol succeed. The* $\texttt{DBID}.\Pi$ *protocol succeeds with probability at least* δ*, since the* DBID *scheme is* (τ, δ)*-complete. Therefore, the* $\texttt{dbid2an}^{\texttt{GM}}.\Pi$ *succeeds with probability at least* δ*, which implies* (τ, δ)*-completeness of* $\texttt{dbid2an}^{\texttt{GM}}$ *scheme.* ■

Lemma 6 (DF-resistance). $\texttt{dbid2an}^{\texttt{GM}}$ *is a* θ*-DF-resistant* AnonDB *(Prop43) scheme, if the* DBID *scheme is* θ*-DF-resistant.*

Proof 6 *In this proof, we reduce any successful* AnonDB *DF adversary to a successful* DBID *DF adversary. Consider an* AnonDB *game with* $\texttt{dbid2an}^{\texttt{GM}}$ *scheme, in which the provers are far-away and the verifier is honest. In each* $\texttt{AnonDB}.\Pi$ *protocol, the verifier gets a temporary public-key and then participates in a* $\texttt{DBID}.\Pi$ *protocol with that temporary public-key. Note that dishonest provers of a single user can operate simultaneously, that implies they can generate different temporary public-keys at the same time*

in different $\mathtt{DBID}.\Pi$ *protocols. Therefore, having multiple users in the system does not increase the chance of adversary against the* DBID *scheme.*

Let's consider the case that the verifier is simultaneously interacting with multiple provers (either from one user or more) in different $\mathtt{AnonDB}.\Pi$ *protocols. We assume all the temporary public-key generations are successful. As a result, the verifier has access to a list of public-keys* $\{C\}$*, and provers have access to the corresponding secret-key* x *and the related randomness* Δ*. The relation between a public-key, secret-key and the corresponding randomness is* $C = Enc(x||\Delta, r)$*, where Enc is the Goldwasser-Micali encryption algorithm and r is pseudo-random.*

After generation of the temporary public-keys, the adversary runs different $\mathtt{DBID}.\Pi$ *protocols simultaneously with the verifier. Since the* DBID *scheme is* θ*-DF-resistant, then for all instances of* $\mathtt{DBID}.\Pi$ *protocols, the acceptance chance of the verifier is limited by negligible value* θ*. As a result, the acceptance chance of the* $\mathtt{dbid2an}^{\mathtt{GM}}.\Pi$ *protocol is limited by* θ *too. Note that the DF-resistant property of* DBID *scheme is considering collusion scenario.* ■

Lemma 7 (TF-resistance). $\mathtt{dbid2an}^{\mathtt{GM}}$ *is a* μ*-TF-resistant* AnonDB *(Prop44) scheme, if the* $\mathtt{DBID}^{\mathtt{GM}}$ *scheme is* μ'*-TF-resistant for negligible values of* μ *and* μ'*.*

Proof 7 *According to the TF-resistance definition, we need to show that for* $\mathtt{dbid2an}^{\mathtt{GM}}$ *scheme, if there is a successful TF attack that is not traceable to any close-by prover, then one can impersonate the far-away prover with the view of all close-by participants, excluding the verifier. This is by assuming that the underlying* $\mathtt{DBID}^{\mathtt{GM}}.\Pi$ *scheme is single-user TF-resistant.*

We divide the $\mathtt{dbid2an}^{\mathtt{GM}}.\Pi$ *protocol into two parts: (i) temporary public-key generation, that is before the* $\mathtt{DBID}.\Pi$ *protocol, and (ii) the* $\mathtt{DBID}^{\mathtt{GM}}.\Pi$ *protocol. In the first part, the verifier receives a message* y' *that allows it to calculate a temporary public-key C as a commitment on secret x and random v. And then in the second part, the two parties run the* $\mathtt{DBID}^{\mathtt{GM}}.\Pi$ *protocol based on the provided public-key C.*

In any valid transcript that uses y' *as the first commitment, the sub-transcript from the* $\mathtt{DBID}^{\mathtt{GM}}.\Pi$ *is a valid transcript according to the temporary public-key C, where for* $j = 1..\lambda$*:* $C[j] = \mathtt{Commit}^{\mathtt{GM}}(x[j], H(x,j).H(v,j)^e)$ *and* $C[\lambda + j] = \mathtt{Commit}^{\mathtt{GM}}(v[j], H(v,j))$*. Because of the binding property of commitment scheme* $\mathtt{Commit}^{\mathtt{GM}}(,)$*, finding any* $x' \neq x$ *such that* $C[j] = \mathtt{Commit}^{\mathtt{GM}}(x'[j], r)$ *for all* $j = 1..\lambda$ *is negligible, for any value of r. This implies finding any* $x' \neq x$ *such that* $C = Com_{H_e}(x,v)$ *is negligible. Therefore, succeeding in the* $\mathtt{DBID}^{\mathtt{GM}}.\Pi$ *sub-protocol with any prover input* $x' \neq x_l$ *is negligible.*

Collusion TF: *The only difference between a close-by prover of another user with a close-by actor, is the possession of secret value* (x_i, σ_i)*. The value of* σ_i *never gets used by the close-by prover, because it makes the session to be traceable to the close-by prover, which is not a TF attack based on definition. So we can consider the close-by prover owns the value* x_i*, while in a normal TF attack the close-by actor owns nothing (or a random value* x'_i*). Since* x_i *is randomly chosen by the group manager in* CertGen *operation, the statistical difference between* x_i *and x is the same as the statistical difference between* x'_i *and x. Therefore, possession of* x_i *or* x'_i *by a close-by party in helping*

the TF attack against $\texttt{DBID}^{\texttt{GM}}.\Pi$ *sub-protocol with public-key C, makes no difference in success chance of the attack. So we can replace the close-by prover of another user with an actor.*

Let's consider a successful TF attack $\mathcal{J}$ *that succeeds with non-negligible probability* κ*. If the transcript is traceable to a close-by prover, this is not an attack according to the definition. Now we consider success chance of attack when no close-by prover is traceable. Since* $\mathcal{J}$ *generates a transcript that is valid with probability* κ*, then the sub-transcript from* $\texttt{DBID}^{\texttt{GM}}.\Pi$ *is valid with at least probability* κ*. And according to the TF-resistance property of* $\texttt{DBID}^{\texttt{GM}}.\Pi$*, there is an impersonator* $\mathcal{E}_{dbid}$ *for the* $\texttt{DBID}^{\texttt{GM}}.\Pi$ *protocol that succeeds with probability* $\kappa - \mu$ *for negligible* μ*.* $\mathcal{E}_{dbid}$ *takes the view of all close-by participants in the attack* $\mathcal{J}$*, excluding the verifier, as input.*

Impersonation against $\texttt{AnonDB}.\Pi$**:** *first we use* $\mathcal{E}_{dbid}$ *to extract the secret x, then find the related* σ *as the first value in the public list* Ξ *that accept* $\leftarrow \texttt{CertVer}(x,\sigma,gpk)$*. By having* (x,σ) *one can impersonate the prover.*

Note that the key extraction x from impersonator $\mathcal{E}_{dbid}$*, depends on the construction of* $\texttt{DBID}^{\texttt{GM}}.\Pi$ *protocol. In zero-knowledge based models, such as* `ProProx` *[32], the impersonator* $\mathcal{E}_{dbid}$ *extracts the key x itself. However, in identification based models,* $\mathcal{E}_{dbid}$ *generates a valid* Σ*-protocol transcript, i.e.,* $(A,[c],[r])$ *for random* $[c]$*. Here we use the following technique to extract the key:*

We divide $\mathcal{E}_{dbid}$ *into two parts:* $\mathcal{J}_1$ *is from the beginning of attack up to after the verifier receives A, and* $\mathcal{J}_2$ *is after that till the end. The first part* $\mathcal{J}_1$ *is run independent from the verifier, and the challenge values* $[c]$ *are randomly chosen by the verifier, where* $[c]$ *is n bits.*

Key extractor: *run* $\mathcal{J}_1$ *once, followed by polynomial* ℓ *times of* $\mathcal{J}_2$*. Before running* $\mathcal{J}_2$ *at any time, we rewind the memory state of the algorithm to the end of* $\mathcal{J}_1$*. This generates the set* Σ *with* ℓ *transcripts (each valid with probability* $\kappa - \mu$*), where* $[c]$ *is randomly generated for each of them. If* ℓ *is chosen polynomially large enough according to n and* κ*, then for every fast-phase challenge-response bit of* $[c]$*, there are at least two valid transcripts that have different values on that bit. An index i is called bad index, if no pair of transcripts in* Σ *have complementary values in this challenge index, which happens with negligible probability* $2^{-\ell}$*. This allows us to extract the whole fast-phase challenge-response table with probability at least* $\kappa - \mu - \lambda.2^{-\ell}$*, where* $\lambda <= n$ *is the length of the table (and size of key x). Finally by having the table, we can find the key x.* ■

Lemma 8 (Soundness). $\texttt{dbid2an}^{\texttt{GM}}$ *is a* γ*-sound* `AnonDB` *(Prop42) scheme for* $\gamma = negl(\lambda)$ *if* $\texttt{DBID}.\Pi$ *is* $negl(\lambda)$*-sound and zero-knowledge, assuming that quadratic residuosity, factorization and RSA problems are hard problems.*

Proof 8 *Before starting the proof, we need to note that since the* $\texttt{dbid2an}^{\texttt{GM}}$ *scheme does not have user revocation, then the corruption query of adversary only consists of actors. i.e., there is no user in the corruption query.*

According to the AnonDB *game settings, we have some prover subsets* $\mathcal{P}^j \in \mathcal{P}$ *that there is no overlap in the execution time of any list* $\mathcal{P}^j$*, however the provers of two different subsets* $\mathcal{P}^j \neq \mathcal{P}^i$ *can run simultaneously. The corrupted actors of* $\mathcal{T}$ *are controlled by the adversary.*

In this game, the adversary succeeds if among all the successful $\texttt{AnonDB}.\Pi$ *protocols* (Π^{succ} *set), at least one of the following conditions hold:*

(i) $\exists \pi \in \Pi^{succ}, \forall p \in \mathcal{P} : t = fshTime(\pi), t \notin [p.St, p.St + exLen(p.Code)]$

(ii) $\exists p \in \mathcal{P}, \exists \pi \in \Pi^{succ}, v \in \mathcal{V} : t = fshTime(\pi), t \in [p.St, p.St + exLen(p.Code)] \wedge d(p.Loc, v.Loc) > \texttt{AnonDB}.\mathcal{D}$.

In this proof, we calculate the success chance of the adversary in both conditions.

First we specify the view and effect of the adversary; as a Σ^**-protocol (Definition 22), the honest parties expect three types of messages in the following order: (1) commitment A, (2) challenge sequence* $[c]$*, and (3) response sequence* $[r]$*. In protocol* $\texttt{dbid2an}^{\texttt{GM}}.\Pi$ *each of these messages are as follows:*

(1) $A = \texttt{DBID}.\Pi.A$*,*

(2) $[c] = \texttt{DBID}.\Pi.c$*, and*

(3) $[r] = (\texttt{DBID}.\Pi.r, y', \pi)$.

Based on the definition, the adversary is able to modify or generate any of these messages. Now we consider the two winning conditions of the adversary:

(i) No prover. *The first condition for the adversary is equivalent to generating a valid transcript* $(A, [c], [r])$ *with random challenges* ($[c]$)*, without the help of any prover. In order to succeed,* $\mathcal{A}$ *needs to successfully pass the* $\texttt{DBID}.\Pi$ *protocol, i.e., generate a valid transcript (*$\texttt{DBID}.\Pi.A, \texttt{DBID}.\Pi.c, \texttt{DBID}.\Pi.r$*) for public-key* $C[j] = (y'[j])^e.\hat{y}[j]$ *mod N and* $C[j+\lambda] = y'[j+\lambda]$ *for* $j = 1...\lambda$*, where* y' *is included in* $[r]$.

$\mathcal{A}$ *has to choose the value of* y' *in a way that the components of derived public-key* $\forall j : C[j] = \texttt{Commit}^{\texttt{GM}}(X, Z)$ *is known to the them, as otherwise the success chance of generating a valid ZKP* $\pi \in [r]$ *is negligible. Therefore,* $\mathcal{A}$ *needs to find a tuple* $(x = X, y' = Y, \Delta = Z)$ *such that it holds in the following relation with the public parameters:* $\forall j : \texttt{Commit}^{\texttt{GM}}(X[j], Z[j]) = (Y[j])^e.\hat{y}[j]$ *mod N. In order to succeed,* $\mathcal{A}$ *needs to solve at least one of the following problems; (a) find some information about the secret of a registered user, or (b) forge the tuple* (X, Y, Z) *such that X is independent from the secret of any registered user.*

> **Case (a).** *We want to find the probability of any information leakage in* $\texttt{dbid2an}^{\texttt{GM}}.\Pi$ *experiment. The provers sends three pieces of information that is dependent to the secret:* $\texttt{DBID}.\Pi.r$*,* y'*, and* π*. The message* π *is zero-knowledge with independent randomness by definition and same about the* $\texttt{DBID}.\Pi$ *protocol according to the assumptions, so we can remove them from the view of* $\mathcal{A}$.

As a result, we only need to find the probability of information leakage in the message y'. Since the message y' perfectly pads the private certificate values with fresh randomness, so the collection of multiple messages of y' does not help the adversary to leak any information about the secrets (x,σ). Therefore the adversary is limited to break the computational hiding of $\texttt{Commit}^{\texttt{GM}}(X,Y) = \theta^X Y^2$ mod N function in order to find the committed values. Note that each bit of the secrets (x,σ) are independent in the protocol, so $\mathcal{A}$'s chance in gaining any information about the secrets is negligible.

Case (b). *$\mathcal{A}$ has to find a tuple (X,Y,Z) such that $\forall j : Y[j]^e = \frac{\texttt{Commit}^{\texttt{GM}}(X[j],Z[j])}{\hat{y}_j} = \frac{\theta^{X[j]}.Z[j]^2}{\hat{y}_j}$. Note that the adversary have seen many values of Y in the learning phase, without knowing the related values of X and Z. Moreover, the learning phase values of Y look random to the adversary as they are perfectly padded by fresh randomness. So we can remove them from the view of adversary, which makes the view of adversary to be completely random (i.e., $View_{\mathcal{A}} = \emptyset$).*

As a result, in order to find this tuple, $\mathcal{A}$ has to solve this equation that needs solving at least one of the following three hard problems:

- *Finding $Y[j]$ as e^{th} root of $\frac{\theta^{X[j]}.Z[j]^2}{\hat{y}_j}$.*
- *Finding $Z[j]$ as square root of $\frac{\hat{y}[j].Y[j]^e}{\theta^{X[j]}}$.*
- *Finding $X[j]$ as discreet log of $\frac{\hat{y}[j].Y[j]^e}{Z[j]^2}$.*

Therefore, all possible ways of soundness adversary to succeed under the condition (i) have negligible chance.

(ii) Far-away provers. *In the following we consider the condition (ii) by assuming that the adversary has no information about the secret of any of the provers involved in the* AnonDB *game. Without loss of generality, we assume there are only two active provers with two different secrets ((x_1,σ_1) and $(x_2,\overline{y_2})$). Since the provers are honest, then they generate two independent values for y'_1 and y'_2 as each one is padded with fresh randomness. Let's assume there is a MiM adversarial algorithm $\mathcal{A}$, in which the provers have (x_1,Δ_1) and (x_2,Δ_2) as their secret in the* DBID.Π *protocols and the verifier accepts with non-negligible probability, while C is the temporary public-key that the verifier calculates. Here we consider two cases; (a) $\exists b \in \{1,2\} : C = Com_{H_e}(x_b,\Delta_b)$, (b) otherwise:*

Case (a). *Without loss of generality, we assume $C = Com_{H_e}(x_1,\Delta_1)$. Now let's consider the* DBID.Π *sub-protocol in this setting. We name the related sub-procedure of $\mathcal{A}$ that runs during the* DBID.Π *protocol, as $\mathcal{A}^{\texttt{DBID}}$. Since* $\texttt{dbid2an}^{\texttt{GM}}$.$\Pi$ *includes the* DBID.Π *protocol, then the acceptance of the verifier in a* $\texttt{dbid2an}^{\texttt{GM}}$.$\Pi$ *session implies the acceptance of the* DBID.Π *sub-protocol. Therefore, the $\mathcal{A}^{\texttt{DBID}}$ algorithm is a successful MiM adversary for the* DBID *protocol with non-negligible success chance. This is in contradiction with the $negl(\lambda)$-soundness property of the* DBID *protocol.*

Case (b). *Since both of the active provers generate non-interactive-ZKP for a different public-key value than C, then the adversary cannot send those proofs to the verifier, because both* $Verify(C,\pi_1)$ *and* $Verify(C,\pi_2)$ *fail. Therefore, the adversary has to generate a different* π *such that* $Verify(C,\pi)$ *succeeds. This implies that the related secret* (x,Δ) *is different from the secrets of the two far-away provers. As a result, in the sub-experiment of the* DBID.Π *protocol, the two far-away provers are counted as actors. Therefore, any non-negligible success chance in the* DBID.Π *protocol is in contradiction with the* $negl(\lambda)$*-soundness property of the* DBID *protocol.*

Therefore, all possible ways of soundness adversary to succeed under the condition (ii) have negligible chance. ■

In above attacks, security against collusion attacks is obtained by simulating the credentials of extra users without having considerable impact on the success chance of the attacker, hence reducing the security to the case of single-user security model. To capture directional TF attack, we reduce the view of the impersonator messages that are sent directly to the helper.

Lemma 9 (Anonymity). $\texttt{dbid2an}^{\texttt{GM}}$ *is an* α*-anonymous* AnonDB *(Prop45) scheme for* $\alpha = negl(\lambda)$*, if the* $\texttt{DBID}^{\texttt{GM}}$ *scheme is zero-knowledge.*

Proof 9 *We consider users* $\mathcal{U} = \{u_1,u_2\}$ *where* $u_b = (x_b,\sigma_b)$ *for* $b \in \{1,2\}$*, and two prover subsets of the same size (i.e.,* $\mathcal{P} = \mathcal{P}^1 \cup \mathcal{P}^2$ *and* $|\mathcal{P}^1| = |\mathcal{P}^2| = n$*). There is no overlap in the execution time of any prover subset* $\mathcal{P}^j$*, however the provers of two different subsets* $\mathcal{P}^j \neq \mathcal{P}^i$ *can run simultaneously. The corrupted actors* $\mathcal{T}$ *and the verifier* $\mathcal{V}$ *are controlled by the adversary. The view of the adversary at the end of this game is:* $\forall i \in \{1,...,n\}, b_i \in_R \{1,2\} : (y'_{b_i}, \pi_i, View_{\mathcal{A}}^{\texttt{DBID}^{\texttt{GM}}.\Pi})$.

The values π_i *and* $View_{\mathcal{A}}^{\texttt{DBID}^{\texttt{GM}}.\Pi}$ *are the outputs of the two zero-knowledge protocols. Therefore, there is an efficient simulator* $\mathcal{S}$ *that can simulate both of these values without having access to the secrets* (x_{b_i}, v_i)*, without decreasing distinguishing advantage of adversary by a non-negligible amount. We thus consider the simulated view of adversary as:* $\forall i \in \{1,...,n\}, b_i \in_R \{1,2\} : y'_{b_i}$*. However, Since each element of* y'_{b_i} *is padded with a fresh pseudo-random (i.e., padded with* $H(x,j).H(v,j)^e$ *for* $1 \leq j \leq \lambda$ *and* $H(v,j)$ *for* $\lambda < j \leq 2\lambda$*, where v is random), the simulated view of the adversary computationally looks random (i.e.,* $View_{\mathcal{A}} = \emptyset$*) and guessing* b_i *will remain random.* ■

Lemma 10 (Traceability). $\texttt{dbid2an}^{\texttt{GM}}$ *is a* γ*-traceable* AnonDB *(Prop46) scheme for* $\gamma = negl(\lambda)$..

Proof 10 *Consider an* AnonDB *game with* $\texttt{dbid2an}^{\texttt{GM}}$ *scheme, in which the verifier are honest. In each* $\texttt{dbid2an}^{\texttt{GM}}.\Pi$ *protocol that the verifier accepts, the* Open *algorithm identifies the user, unless the prover doesn't use the certificate of a user (i.e., forgery), which has negligible probability according on soundness property (Lemma 8). So we have* $\Pr[identify\ user] \geq \gamma + \Pr[\Pi\ succeeds]$*, which implies* $negl(\lambda)$*-traceability.* ■

6 AnonDB Construction: dbid2an$^{\mathsf{P}}$

We refer to our AnonDB scheme as dbid2an$^{\mathsf{P}}$ to emphasize conversion of a DBID scheme to an *anonymous* DBID. The DBID scheme has to use Pedersen commitment scheme [28] for key generation. We first give an overview of our proposed scheme and then provide the details. In dbid2an$^{\mathtt{GM}}$, a user is first enrolled in the system and is provided with a verifiable "membership" certificate. The membership certificate is generated by a CLSig signature scheme (Section 2.2), such as BBS^{+} [16].

In addition to verifying the membership of a user, the certificate is used to generate a temporary public-key, which is later used in a public-key DBID protocol. At the end of a successful execution, the verifier is convinced that a valid member of the group is within the given distance bound.

Recall (Definition 41) that for a security parameter λ, an anonymous distance-bounding (AnonDB) scheme is defined by a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathcal{D}, p_{noise}, \mathtt{Init}, \mathtt{CertGen}, \mathtt{CertVer}, \Pi, \mathtt{Revoke}, \mathtt{Open})$. For our proposed (AnonDB) scheme dbid2an$^{\mathsf{P}}$, we name these operations as dbid2an$^{\mathsf{P}}$.Init, dbid2an$^{\mathsf{P}}$.CertGen, dbid2an$^{\mathsf{P}}$.CertVer and dbid2an$^{\mathsf{P}}$.Π. Note that dbid2an$^{\mathsf{P}}$ scheme does not have Open and Revoke operations.

In dbid2an$^{\mathsf{P}}$.CertGen, the group manager generates a membership certificate for a new user, by running the BSign protocol of CLSig signature scheme. Then dbid2an$^{\mathsf{P}}$.Π protocol takes place as below:

(i) a prover of the user u_i, $i = 1..l$, anonymously proves that it owns a membership certificate signed by group manager, by running the SPK protocol of CLSig signature scheme.

(ii) a temporary public-key is generated for the prover. The temporary public-key is generated by using Pedersen commitment, as a result of SPK protocol. So we have $C = Commit^{P}(x, \Delta)$ where x is secret of the user, Δ is a random value chosen by the prover, and C is temporary public-key. This temporary public-key generation is equivalent to the DBID$^{\mathsf{P}}$.KeyGen function. After establishing the temporary public-key, the prover and the verifier run a DBID$^{\mathsf{P}}$.Π protocol, where the prover uses (x, Δ) as input, and the verifier uses C as input.

In our construction of dbid2an$^{\mathsf{P}}$, we use POXY [**?**] as the DBID$^{\mathsf{P}}$ scheme, which is proven secure in the model of DBID schemes (directional antenna and single user attacks).

Now we provide the details of the operations:

dbid2an$^{\mathsf{P}}$.Init*:* $(msk, gpk) \leftarrow \mathtt{Init}(1^{\lambda})$. The group manager initiates the system as follows:

- Initialize Pedersen commitment: $(msk, pk, p) \leftarrow$ CLSig.KeyGen(1^{λ}) for λ bit security choose large prime p. Private: (msk) and Public: (pk, p).

The group master key is msk; the group public-key is $gpk = (pk, p, \Xi)$, where Ξ is the list of all user membership signatures that is initialized to $\Xi = \emptyset$.

$\texttt{dbid2an}^{\texttt{P}}.\texttt{CertGen}$: $(s, msk', gpk') \leftarrow \texttt{CertGen}(msk, gpk)$. The group manager generates a membership certificate ($s = (x, \sigma)$) and sends securely to the new user.The public parameters of the system are updated accordingly, i.e., $msk \leftarrow msk'$ and $gpk \leftarrow gpk'$. The details are as follows:

1. randomly chooses $x \in \texttt{CLSig}.\mathbb{M}$, and
2. sign x using $\sigma \leftarrow \texttt{CLSig.Sign}(x, msk)$.

The group master key stays unchanged is $msk' = msk$, and the updated group public-key is $gpk' = (pk, p, \Xi')$ for $\Xi' = \Xi \cup \sigma$. The certificate $s = (x, \sigma)$ is securely sent to the new user.

This operation can also be implemented as a protocol between the user and group manager, *i.e.*, $\texttt{CertGen}\{U(gpk) \leftrightarrow GM(msk, gpk)\}$. The steps of protocol would be as follows

1. U randomly chooses $x \in \texttt{CLSig}.\mathbb{M}$, and
2. U and GM run the blind signature protocol $\texttt{CLSig.BSign}\{U(x, pk, p) \leftrightarrow GM(msk)\}$. At the end of this protocol, both the user and the group manager output a signature σ on the message x.

$\texttt{dbid2an}^{\texttt{P}}.\texttt{CertVer}$: $accept/reject \leftarrow \texttt{CertVer}(s, gpk)$. Upon receiving a certificate $s = (x_l, \sigma_l)$, the user can check its validity. By reading the group public-key $gpk = (pk, p, \Xi)$ for $\Xi = \{\sigma_1, ..., \sigma_l\}$, the user checks if σ is included in Ξ and verifies its' validity using $accept \leftarrow \texttt{CLSig.Verify}(x, \sigma, pk, p)$ function.

$\texttt{dbid2an}^{\texttt{P}}.\Pi$: $accept/reject \leftarrow \Pi\{P(s, gpk) \leftrightarrow V(gpk)\}$. When a prover ($\mathcal{P}_l$) of a registered user wants to run the $\texttt{AnonDB}.\Pi$ protocol with the verifier, they will follow the protocol described in Figure 12. The protocol consists of two main steps. The first step is a message from the prover to the verifier (π) that includes a temporary public-key C on prover's secret x and a non-interactive $\texttt{CLSig.SPK}$ which proves that the prover knows a signature of the group manager on the secret x without leaking information about the secret or the signature. The second step is running the $\texttt{DBID}^{\texttt{P}}.\Pi$ protocol.

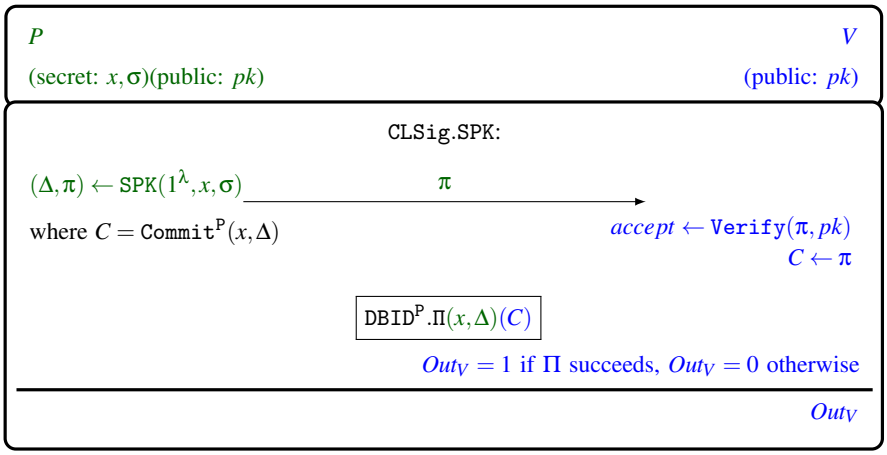

Fig. 12: Π protocol in $\mathtt{dbid2an}^{\mathtt{P}}$ scheme for the l^{th} user. $\mathtt{Commit}^{\mathtt{P}}(x,\Delta)$ is the Pedersen commitment function as $\mathtt{Commit}^{\mathtt{P}}(x,\Delta) = g^x h^\Delta \bmod p$. Note that we are using a non-interactive protocol CLSig.SPK, which allows us to break down the protocol into two pieces $\mathtt{CLSig.SPK} = (\mathtt{SPK}, \mathtt{Verify})$. Note that the value of C is embedded inside π, and we use the notation $C \leftarrow \pi$ to indicate the extraction of C from π.

6.1 Security Analysis of $\mathtt{dbid2an}^{\mathtt{P}}$

In this section we provide the security analysis of $\mathtt{dbid2an}^{\mathtt{P}}$ protocol, assuming that the adopted DBID protocol and CLSig schemes are secure.

Theorem 2 **($\mathtt{dbid2an}^{\mathtt{P}}$ Security Properties).** *If (i) the* $\mathtt{DBID}^{\mathtt{P}}$ *scheme is* (τ,δ)*-complete,* γ'*-sound,* θ*-DF-resistant,* μ'*-TF-resistant and* $\mathtt{DBID}^{\mathtt{P}}.\Pi$ *is zero-knowledge protocol, (ii)* CLSig *scheme is complete, unforgeable, sound, and zero-knowledge with non-interactive* SPK *protocol based on the* CLSig *model (Definition 24), and (iii) the temporary public-key C and the secret key x of* $\mathtt{DBID}^{\mathtt{P}}.\Pi$ *are related as* $C = \mathtt{Commit}^{\mathtt{P}}(x,\Delta)$ *for a known value of* Δ *to prover, where* $\mathtt{Commit}^{\mathtt{P}}(.,.)$ *is Pedersen commitment,*

then $\mathtt{dbid2an}^{\mathtt{P}}$ *is an* AnonDB *scheme that is* (τ,δ)*-complete (Prop41),* θ*-DF-resistant (Prop43),* γ*-Sound (Prop42),* μ*-TF-resistant (Prop44), and* α*-anonymous (Prop45), for negligible values of* α*,* δ*,* γ*,* γ'*,* μ*,* μ' *and* θ*, assuming that the discrete logarithm is a hard problem.*

In the following, we prove each of the properties of the theorem in a separate lemma. We prove security properties of the protocol based on the model described in Section 4. The underlying $\mathtt{DBID}^{\mathtt{P}}$ protocol provides single user directional antenna security. The main challenge for the new model is to prove collusion security.

Lemma 11 **(Completeness).** $\mathtt{dbid2an}^{\mathtt{P}}$ *is a* (τ,δ)*-complete* AnonDB *(Prop41) scheme, if the* CLSig *scheme is complete and the* DBID *scheme is* (τ,δ)*-complete.*

Proof 11 *Consider an* AnonDB *game with* dbid2an$^{\mathrm{P}}$ *scheme, in which the provers and the verifier are honest. In each* dbid2an$^{\mathrm{P}}$.Π *protocol, since the* CLSig *scheme is complete, then the verifier accepts in the* CLSig.SPK *protocol and receives the correct value of C. Since the* DBID *scheme is* (τ,δ)*-complete, then the verifier accepts the* DBID.Π *protocol with probability at least* δ*. Therefore, the* dbid2an$^{\mathrm{P}}$ *scheme is* (τ,δ)*-complete.* ■

Lemma 12 (DF-resistance). dbid2an$^{\mathrm{P}}$ *is a* θ*-DF-resistant* AnonDB *(Prop43) scheme, if the* DBID *scheme is* θ*-DF-resistant.*

Proof 12 *In this proof, we reduce any successful* AnonDB *DF adversary to a successful* DBID *DF adversary. Consider an* AnonDB *game with* dbid2an$^{\mathrm{P}}$ *scheme, in which the provers are far-away and the verifier is honest. In each* AnonDB.Π *protocol, the verifier gets a commitment and then participates in a* DBID.Π *protocol with that commitment. Note that dishonest provers of a single user can operate simultaneously, that implies they can generate different commitments at the same time in different* DBID.Π *protocols. Therefore, having multiple users in the system does not increase the chance of adversary.*

Let's consider the case that the verifier is simultaneously interacting with multiple provers (either from one user or more) in different AnonDB.Π *protocols. We assume all the commitment generations are successful. As a result, the verifier has access to a list of commitments* $\{C\}$*, and the provers have access to the corresponding secret-key* x *and the related randomness* Δ*. The relation between a commitment, secret-key and the corresponding randomness is* $C = \mathtt{Commit}^{\mathrm{P}}(x,\Delta)$*, where* Commit$^{\mathrm{P}}$ *is the Pedersen commitment.*

After generation of the commitment, the adversary runs different DBID.Π *protocols simultaneously with the verifier. Since the* DBID *scheme is* θ*-DF-resistant, then for all instances of* DBID.Π *protocols, the acceptance chance of the verifier is limited by negligible value* θ*. As a result, the acceptance chance of the* dbid2an$^{\mathrm{P}}$.Π *protocol is limited by* θ *too. Note that the DF-resistant property of* DBID *scheme is considering collusion scenario.* ■

Lemma 13 (TF-resistance). dbid2an$^{\mathrm{P}}$ *is a* μ*-TF-resistant* AnonDB *(Prop44) scheme, when there is no close-by prover, if the* DBID *scheme is* μ'*-TF-resistant for negligible values of* μ *and* μ'*.*

Note that since there is no traceability in dbid2an$^{\mathrm{P}}$ scheme, then the case of having the special case of TF attack as an AnonDB game (Prop44) with far-away provers of user u_1 and close-by provers of user u_2 is meaningless. Therefore, as stated in this lemma, we do not consider any close-by prover, which means the only close-by participants are actors and the verifier.

Proof 13 *According to the TF-resistance definition, we need to show that for* dbid2an$^{\mathrm{P}}$ *scheme, if there is a successful TF attack, then one can impersonate the far-away prover with the view of all close-by participants, excluding the verifier. This is by assuming that the underlying* DBID$^{\mathrm{P}}$.Π *scheme is single-user TF-resistant.*

We divide the $\texttt{dbid2an}^{\texttt{P}}.\Pi$ *protocol into two parts: (i) temporary public-key generation that is by* $\texttt{CLSig.SPK}$ *protocol, and (ii) the* $\texttt{DBID}.\Pi$ *protocol. In the first part, the verifier receives a message* π *that allows it to extract a temporary public-key C as a commitment on secret x. And then in the second part, the two parties run the* $\texttt{DBID}^{\texttt{P}}.\Pi$ *protocol based on the provided public-key C.*

In any valid transcript that uses π *as the first commitment, the sub-transcript from the* $\texttt{DBID}^{\texttt{P}}.\Pi$ *is a valid transcript according to the temporary public-key C, where* $C = \texttt{Commit}^{\texttt{P}}(x,\Delta)$. *Note that because of the binding property of commitment scheme* $\texttt{Commit}^{\texttt{P}}(,)$, *finding any* $x' \neq x$ *such that* $C = \texttt{Commit}^{\texttt{P}}(x',r)$ *is negligible, for any value of r. Therefore, succeeding in the* $\texttt{DBID}^{\texttt{P}}.\Pi$ *sub-protocol with any prover input* $x' \neq x$ *is negligible.*

Let's consider a successful TF attack $\mathcal{J}$ *that succeeds with non-negligible probability* κ. *Since* $\mathcal{J}$ *generates a transcript that is valid with probability* κ, *then the sub-transcript from* $\texttt{DBID}^{\texttt{P}}.\Pi$ *is valid with at least probability* κ. *And according to the TF-resistance property of* $\texttt{DBID}^{\texttt{P}}.\Pi$, *there is an impersonator* $\mathcal{E}_{dbid}$ *for the* $\texttt{DBID}^{\texttt{P}}.\Pi$ *protocol that succeeds with probability* $\kappa - \mu$ *for negligible* μ. $\mathcal{E}_{dbid}$ *takes the view of all close-by participants in the attack* $\mathcal{J}$, *excluding the verifier, as input.*

Impersonation against $\texttt{AnonDB}.\Pi$**:** *first we use* $\mathcal{E}_{dbid}$ *to extract the secret x, then find the related* σ *as the first value in the public list* Ξ *that accept* $\leftarrow \texttt{CertVer}(x,\sigma,gpk)$. *By having* (x,σ) *one can impersonate the prover.*

We divide $\mathcal{E}_{dbid}$ *into two parts:* $\mathcal{J}_1$ *is from the beginning of attack up to after the verifier receives A, and* $\mathcal{J}_2$ *is after that till the end. The first part* $\mathcal{J}_1$ *is run independent from the verifier, and the challenge values* $[c]$ *are randomly chosen by the verifier, where* $[c]$ *is n bits.*

Key extractor: *run* $\mathcal{J}_1$ *once, followed by polynomial* ℓ *times of* $\mathcal{J}_2$. *Before running* $\mathcal{J}_2$ *at any time, we rewind the memory state of the algorithm to the end of* $\mathcal{J}_1$. *This generates the set* Σ *with* ℓ *transcripts (each valid with probability* $\kappa - \mu$*), where* $[c]$ *is randomly generated for each of them. If* ℓ *is chosen polynomially large enough according to n and* κ, *then for every fast-phase challenge-response bit of* $[c]$, *there are at least two valid transcripts that have different values on that bit. An index i is called bad index, if no pair of transcripts in* Σ *have complementary values in this challenge index, which happens with negligible probability* $2^{-\ell}$. *This allows us to extract the whole fast-phase challenge-response table with probability at least* $\kappa - \mu - \lambda.2^{-\ell}$, *where* $\lambda <= n$ *is the length of the table (and size of key x). Finally by having the table, we can find the key x.* ■

Lemma 14 (Soundness). $\texttt{dbid2an}^{\texttt{P}}$ *is a* γ*-sound* $\texttt{AnonDB}$ *(Prop42) scheme for* $\gamma = negl(\lambda)$ *if (i)* $\texttt{CLSig}$ *scheme is unforgeable, sound and zero-knowledge, and (ii)* $\texttt{DBID}$ *scheme is* $negl(\lambda)$*-sound and zero-knowledge, assuming that discrete logarithm is a hard problem.*

Proof 14 *Before starting the proof, we need to note that since the* $\texttt{dbid2an}^{\texttt{P}}$ *scheme does not have user revocation, then the corruption query of adversary only consists of actors. i.e., there is no user in the corruption query.*

According to the AnonDB *game settings, we have some prover subsets* $\mathcal{P}^j \in \mathcal{P}$ *that there is no overlap in the execution time of any list* $\mathcal{P}^j$*, however the provers of two different subsets* $\mathcal{P}^j \neq \mathcal{P}^i$ *can run simultaneously. The corrupted actors of* $\mathcal{T}$ *are controlled by the adversary.*

In this game, the adversary succeeds if among all the successful AnonDB.Π *protocols* (Π^{succ} *set), at least one of the following conditions hold:*

(i) $\exists \pi \in \Pi^{succ}, \forall p \in \mathcal{P} : t = fshTime(\pi), t \notin [p.St, p.St + exLen(p.Code)]$

(ii) $\exists p \in \mathcal{P}, \exists \pi \in \Pi^{succ}, v \in \mathcal{V} : t = fshTime(\pi), t \in [p.St, p.St + exLen(p.Code)] \wedge d(p.Loc, v.Loc) > \texttt{AnonDB}.\mathcal{D}$.

In this proof, we calculate the success chance of the adversary in both conditions.

First we specify the view and effect of the adversary; as a Σ^**-protocol (Definition 22), the honest parties expect three types of messages in the following order: (1) commitment A, (2) challenge c, and (3) response r. In protocol* $\texttt{dbid2an}^{\mathsf{P}}.\Pi$ *each of these messages are as follows:*

(1) $A = (\texttt{CLSig.SPK}.\pi^{\bar{x}}, \texttt{DBID}.\Pi.A)$*, where* $\texttt{CLSig.SPK}.\pi^{\bar{x}}$ *is the sections of message* π *that are independent from x,*

(2) $c = \texttt{DBID}.\Pi.c$*, and*

(3) $r = (\texttt{CLSig.SPK}.\pi^{x}, \texttt{DBID}.\Pi.r)$*, where* $\texttt{CLSig.SPK}.\pi^{x}$ *is the sections of message* π *that are dependent to x.*

Based on the definition, the adversary is able to modify or generate any of these messages. Now we consider the two winning conditions of the adversary:

(i) No prover. *The first condition for the adversary is equivalent to generating a valid transcript* (A, c, r) *with random challenges (c), without the help of any prover. In order to succeed,* $\mathcal{A}$ *needs to successfully pass the* DBID.Π *protocol, i.e., generate a valid transcript (*$\texttt{DBID}.\Pi.A, \texttt{DBID}.\Pi.c, \texttt{DBID}.\Pi.r$*) for a public-key C. Here we consider two cases about C: (a)* $C = \texttt{Commit}^{\mathsf{P}}(x, \Delta)$ *where x is the secret of a user in set* $\mathcal{U}$*, (b) there is no user in* $\mathcal{U}$ *that has the secret x, where* $C = \texttt{Commit}^{\mathsf{P}}(x, \Delta)$ *and the adversary know the values of x and* Δ*.*

Case (a). *In order to succeed in the* CLSig.SPK *protocol, the adversary needs to either know x based on the soundness property of the* CLSig *scheme, or replay an earlier valid message* π*. Knowing x doesn't happen in this case, unless negligible probability, because it is in contradiction with the zero-knowledge property of the* CLSig *scheme and harness of discrete logarithm problem. Replaying the* π *message of a valid legitimate prover* $p \in \mathcal{P}$*, implies that the* DBID.Π *protocol is running with the same public-key C as the prover p has used in earlier* DBID.Π *protocol. This doesn't happen, unless negligible probability, because it is in contradiction with the soundness property of the* DBID *scheme.*

Case (b). *This case doesn't happen, unless negligible probability, because it is in contradiction with the soundness property of the* CLSig *scheme.*

Therefore, all possible ways of the MiM adversary to succeed under the condition (i) have negligible chance.

(ii) Far-away provers. *In the following we consider the condition (ii) by assuming that the adversary has no information about the secret of any of the provers. Without loss of generality, we assume there are only two active provers with two different secrets* (x_1, σ_1) *and* (x_2, σ_2)*. Since the provers are honest, then they generate two different values of* π_1 *and* π_2 *that implies two public-keys* $C_1 = \texttt{Commit}^{\texttt{P}}(x_1, r_1)$ *and* $C_2 = (x_2, r_2)$*. The two values* C_1 *and* C_2 *are independent, because the inputs of the related commitment functions are independent keys and fresh randomness.*

Let's assume that there is a MiM adversarial algorithm $\mathcal{A}$*, in which the provers have* (x_1, r_1) *and* (x_2, r_2) *as their secret in the* DBID.Π *protocols and the verifier accepts with non-negligible probability, while C is the temporary public-key that the verifier calculates. Here we consider two cases; (a)* $C = \texttt{Commit}^{\texttt{P}}(x, \Delta)$ *where* $x \in \{x_1, x_2\}$*, (b) there is no* $x \in \{x_1, x_2\}$ *where* $C = \texttt{Commit}^{\texttt{P}}(x, \Delta)$ *and the adversary know the values of x and* Δ*.*

Case (a). *Without loss of generality, we assume* $C = \texttt{Commit}^{\texttt{P}}(x_1, \Delta)$*. Now let's consider the* DBID.Π *sub-protocol in this setting. We name the related sub-procedure of* $\mathcal{A}$ *that runs during the* DBID.Π *protocol, as* $\mathcal{A}^{\texttt{DBID}}$*. Since* $\texttt{dbid2an}^{\texttt{P}}.\Pi$ *includes the* DBID.Π *protocol, then the acceptance of the verifier in a* $\texttt{dbid2an}^{\texttt{P}}.\Pi$ *session implies the acceptance of the* DBID.Π *sub-protocol. Therefore, the* $\mathcal{A}^{\texttt{DBID}}$ *algorithm is a successful MiM adversary for the* DBID *protocol with non-negligible success chance. This is in contradiction with the* $negl(\lambda)$*-soundness property of the* DBID *protocol.*

Case (b). *The active provers generate the messages* π_1 *and* π_2 *that respectively contain two independent public-keys* C_1 *and* C_2*. Let's assume the adversary sends the message* π *to the verifier, that contains the public-key C. Based on the assumption of the case, the related x is not among* $\{x_1, x_2\}$*. Therefore, the adversary does not have access to a valid signature on x, based on the unforgeability property of the* CLSig *scheme. As a result, if the adversary succeed in the* CLSig.SPK *protocol with non-negligible probability, then we can use it to break the soundness property of the* CLSig *scheme.*

Therefore, all possible ways of the MiM adversary to succeed under the condition (ii) have negligible chance. ■

Lemma 15 (Anonymity). $\texttt{dbid2an}^{\texttt{P}}$ *is an* α*-anonymous* AnonDB *(Prop45) scheme for* $\alpha = negl(\lambda)$*, if the* CLSig *scheme and the* DBID *scheme are zero-knowledge.*

Proof 15 *We consider users* $\mathcal{U} = \{u_1, u_2\}$ *where* $u_b = (x_b, \sigma_b)$ *for* $b \in \{1, 2\}$*, and two prover subsets of the same size (i.e.,* $\mathcal{P} = \mathcal{P}^1 \cup \mathcal{P}^2$ *and* $|\mathcal{P}^1| = |\mathcal{P}^2| = n$*). There is no overlap in the execution time of any prover subset* $\mathcal{P}^j$*, however the provers of two different subsets* $\mathcal{P}^j \neq \mathcal{P}^i$ *can run simultaneously. The corrupted actors* $\mathcal{T}$ *and the verifier* $\mathcal{V}$ *are controlled by the adversary. The view of the adversary at the end of this game is:* $\forall i \in \{1, ..., n\}, b_i \in_R \{1, 2\} : (\pi_i, View_{\mathcal{A}}^{\texttt{DBID}^{\texttt{P}}.\Pi})$.

The values π_i and $View_{\mathcal{A}}^{\mathtt{DBID}^{\mathtt{P}}.\Pi}$ are the outputs of two zero-knowledge protocols. Therefore, there is an efficient simulator $\mathbb{S}$ that can simulate both of these values without having access to the secrets (x_{b_i}, Δ_i), without decreasing distinguishing advantage of adversary by a non-negligible amount. Therefore, the simulated view of the adversary computationally looks random (i.e., $View_{\mathcal{A}} = \emptyset$) and guessing b_i will remain random. ■

7 Related Works

There are three known anonymous DB protocols [3,14,8], that are designed to be secure against all distance-bounding attacks, which were all shown insecure against our proposed attacks.

[3] formally defines *Distance-Fraud*, *Mafia-Fraud*, *Strong-Impersonation*, *Original Terrorist-Fraud*, *Distance-Hijacking* and considers *Anonymity* of provers. In this model, the verifier only has access to the public parameters of the system. However it has some disadvantages: the scheme does not provide `revocation` and uses the *Original TF* definition that, as argued in Section 1, is not appropriate for anonymous DB.

[14] proposed an anonymous distance bounding model, which considers *Distance-Fraud*, *Mafia-Fraud* and *Terrorist-Fraud* in addition to anonymity of provers against the verifier. This model achieves anonymity and revocability by using a revocable group signature scheme, that allows `join`, `revocation` and `escrow` operations for provers. However, in this protocol the verifier must be registered in the system which makes its application more limited compared to that of [3]. [8] uses the same model and structure as [14].

8 Conclusion

We showed the security challenges that arise when identity information is not directly used in DB protocols, and proposed a new model that captures all known attacks and a construction with provable security in this model. We introduced two attacks; directional attack that uses the capability of an attacker at the physical layer of communication, and collusion attack that the provers of multiple user collude to deceive the verifier. And we showed that all existing anonymous DB schemes are vulnerable against our attack.

We proposed two constructions for different cryptosystems that convert public-key DB protocols to anonymous DB protocols. These constructions are modular and can use similar components that follows the designed cryptosystem. These two protocols are the first that are resistant against all distance-bounding attacks, including directional antenna attacks. The security properties of these protocols are provided.

References

1. Mamta Agiwal, Abhishek Roy, and Navrati Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2016.
2. Ahmad Ahmadi and Reihaneh Safavi-Naini. Distance-bounding identification. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 202–212. INSTICC, SciTePress, 2017.
3. Ahmad Ahmadi and Reyhaneh Safavi-Naini. Privacy-preserving distance-bounding proof-of-knowledge. In *16th ICICS*, 2014.
4. Ahmad Ahmadi and Reyhaneh Safavi-Naini. Directional distance-bounding identification protocols. volume 2018, page 349, 2018.
5. Ahmad Ahmadi, Reyhaneh Safavi-Naini, and Mamunur Rashid Akand. New attacks and secure design for anonymous distance-bounding. In *Australasian Conference on Information Security and Privacy*. Springer, 2018.
6. Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-taa. In *Security and Cryptography for Networks*, pages 111–125. Springer, 2006.
7. Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardaş, Cédric Lauradoux, and Benjamin Martin. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security*, 2011.
8. Gildas Avoine, Xavier Bultel, Sébastien Gambs, David Gérault, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. A terrorist-fraud resistant and extractor-free anonymous distance-bounding protocol. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 800–814. ACM, 2017.
9. Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In *Advances in Cryptology-CRYPTO'89*, 1990.
10. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 1988.
11. Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Practical & provably secure distance-bounding. In *The 16th Information Security Conference*, 2013.
12. Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Secure & lightweight distance-bounding. In *International Workshop on Lightweight Cryptography for Security and Privacy*, 2013.
13. Stefan Brands and David Chaum. Distance-bounding protocols. In *Advances in Cryptology–EUROCRYPT'93*, pages 344–359. Springer, 1994.
14. Xavier Bultel, Sébastien Gambs, David Gérault, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *WiSec '16*, 2016.
15. Laurent Bussard and Walid Bagga. Distance-bounding proof of knowledge protocols to avoid terrorist fraud attacks. Technical report, Technical report, Institut Eurecom, France, 2004.
16. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO'04*, pages 56–72, 2004.
17. Cas Cremers, Kasper Bonne Rasmussen, Benedikt Schmidt, and Srdjan Capkun. Distance hijacking attacks on distance bounding protocols. In *Security and Privacy*, 2012.
18. Ivan Damgård. On Σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, 2002.
19. Ivan Damgård, Kasper Dupont, and Michael Østergaard Pedersen. Unclonable group identification. In *Advances in Cryptology-EUROCRYPT 2006*, pages 555–572. Springer, 2006.

20. Yvo Desmedt. Major security problems with the ünforgeable¨(feige-)fiat-shamir proofs of identity and how to overcome them. In *Securicom'88*, 1988.
21. Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A formal approach to distance-bounding rfid protocols. In *International Conference on Information Security*, pages 47–62. Springer, 2011.
22. Aurélien Francillon, Boris Danev, and Srdjan Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*, 2011.
23. Rosario Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In *Annual International Cryptology Conference*, 2004.
24. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.
25. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, pages 281–308, 1988.
26. Louis C Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *EUROCRYPT '88*, 1988.
27. Kaoru Kurosawa and Swee-Huay Heng. The power of identification schemes. In *Public Key Cryptography-PKC 2006*, pages 364–377. Springer, 2006.
28. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO'91*, pages 129–140. Springer, 1992.
29. Kasper Bonne Rasmussen and Srdjan Capkun. Realization of rf distance bounding. In *USENIX Security Symposium*, pages 389–402, 2010.
30. C P Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 1991.
31. Serge Vaudenay. On modeling terrorist frauds. In *Provable Security*. Springer, 2013.
32. Serge Vaudenay. Proof of proximity of knowledge. *IACR Eprint*, 695, 2014.