Efficient Public-Key Distance Bounding Protocol

Handan Kılınç and Serge Vaudenay

EPFL, Lausanne, Switzerland

Abstract. Distance bounding protocols become more and more important because they are the most accurate solution to defeat relay attacks. They consist of two parties: a verifier and a prover. The prover shows that (s)he is close enough to the verifier. In some applications such as payment systems, using public-key distance bounding protocols is practical as no pre-shared secret is necessary between the payer and the payee. However, public-key cryptography requires much more computations than symmetric key cryptography. In this work, we focus on the efficiency problem in public-key distance bounding protocols and the formal security proofs of them. We construct two protocols (one without privacy, one with) which require fewer computations on the prover side compared to the existing protocols, while keeping the highest security level. Our construction is generic based on a key agreement model. It can be instantiated with only one resp. three elliptic curve computations for the prover side in the two protocols, respectively. We proved the security of our constructions formally and in detail.

Keywords: distance bounding, RFID, NFC, relay attack, key agreement, mafia fraud, distance fraud, distance hijacking

1 Introduction

Nowadays, various technologies, such as contactless payment (e.g. NFC), access control in a building, remote keyless system (e.g. car keys) are part of our lives since they provide us efficient usage of time and accessibility. However, these applications are exposed to simple but dangerous attacks such as relay attacks. A malicious person can abuse all these technologies by just relaying messages.

Distance bounding (DB) is a solution to detect the relay attacks. The detection of the attack is simpler, cheaper and more practical than preventing it because prevention could require a special hardware equipment [4]. The first DB protocol is introduced by Brands and Chaum [9]. Basically in DB, the verifying party measures the physical distance of the proving party by sending the challenges and receiving the responses (they are generally 1 or 2 bit(s)). In the end, if too many rounds have too long round trip times or too many incorrect responses, the verifier rejects the proving party since he may be exposed to a relay attack.

Threats for DB is not limited to only relay attacks. The other threats are the following:

Distance Fraud (DF): A malicious, far-away prover tries to prove that (s)he is close enough.

Mafia Fraud (MiM) [13]: A man-in-the-middle (MiM) adversary between a verifier and a far-away honest prover tries to make the verifier accept.

Terrorist fraud (TF) [13]: A far-away malicious prover, with the help of the adversary, tries to make the verifier accept, but without giving any advantage to the adversary to later pass the protocol alone.

Distance Hijacking (DH) [12]: A far-away malicious prover takes advantage of some honest and active provers who are close to the verifier to make the verifier grant privileges to the far-away prover.

Privacy threat: An adversary tries to learn any useful information such as the identity of a prover. In *strong privacy*, the adversary tries to identify the identity of a prover with access to the prover's secret (e.g. by corruption).

DB protocols are categorized as symmetric DB protocols (the verifier and the prover share a secret) [34, 16, 5, 7, 6, 8, 23, 25, 24] and public-key DB protocols (the verifier and the prover only know the public key of each other) [9, 10, 20, 17, 38, 37, 35].

In some applications, we cannot assume that the prover and the verifier have established a secret. For example, in a payment system, it is not realistic to assume that the payment terminal and the customer share a secret. We can mention as an instance of a payment protocol the EMV standard [1] which now uses the public-key DB protocol PaySafe from [11]. However, this protocol sends nonces of several bits through the time-critical channel. Normally, a time-critical exchange should only take a few nanoseconds to reach a distance bound of meters with the speed of light, but sending a string of several bits typically takes microseconds. This is why usual DB protocols only exchange single bits through the time-critical phases. Actually, the protocol from [11] does not protect against adversaries running computations at the speed of light but only against adversaries using standard equipment which induce natural delays.

Although public-key distance bounding protocols are useful, it can cause **considerable energy consumption** on the prover side since public-key cryptography needs heavier computations than symmetric-key cryptography. Energy constraints on most of the powerless devices using RFID and NFC technologies cause very limited computation resources. One of the solutions could be to add more computational power to these devices but it increases their costs.

In this paper, we construct new protocols called Eff-pkDB, Eff-pkDB^p, and Simp-pkDB (Eff-pkDB^p is the privacy-preserving variant of Eff-pkDB).

Table 1 shows the security and the efficiency properties of previous protocols and our protocols. We can see that most of the previous public-key DB protocols [9, 10, 17, 38, 37, 35] do not concentrate on this efficiency problem, except HPO [20]. So far, HPO is the most efficient one among them since it requires only 4 elliptic curve (EC) multiplications on the prover side, but it is not strong private [36] and it is not secure against DH [22] and TF. In addition to this, its security is based on several ad-hoc assumptions [20] which are not so well studied: "OMDL", "Conjecture 1", "extended ODH" and "XL".

Protocol	**MiM**	**DF**	**DH**	**TF**	**Privacy**	**Strong Privacy**	**PK Computations for the Prover**
Brands-Chaum [9]	✓	✓	×	×	×	×	1 commitment, 1 signature
HPO [20]	✓	✓	×	×	✓	×	4 EC multiplications
GOR [17]	✓	✓	×	×	×	×	4 EC multiplications, 1 encryption, 1 NIZK proof
PaySafe [11]	✓*	×	×	×	×	×	1 signature
PrivDB [37]	✓	✓	✓	×	✓	✓	1 signature, 1 IND-CCA encryption
ProProx [38]	✓	✓	✓	✓	×	×	$n+1$ commitments, n ZK proofs
eProProx [35]	✓	✓	✓	✓	✓	✓	1 encryption, s hashing, $n+1$ commitments, n ZK proofs
Simp-pkDB	✓	✓	×	×	×	×	1 IND-CCA decryption
Eff-pkDB	✓	✓	✓	×	×	×	1 AKA protocol
Eff-pkDB^p	✓	✓	✓	×	✓	✓	1 IND-CCA Encryption, 1 AKA protocol

Table 1. The review of the existing public-key DB protocols. ✓ means that it is secure for corresponding threat model and × means it is not. ✓* means that it is secure against the adversaries that cannot relay the messages close to the speed of light. EC is elliptic curve, ZK is zero knowledge, NIZK is non-interactive zero knowledge, AKA is authenticated key agreement. Public key (PK) computations are counted only on prover side. n is the number of rounds in the challenge phase and s is the security parameter.

GOR [17] was constructed to have strong privacy, but it has been shown in [36] that it is neither strong private nor private.

ProProx [38] satisfies all the security properties except privacy. Its version eProProx [35] is secure against all threat models and strong private. However, both ProProx and eProProx suffer from heavy cryptographic operations as zero-knowledge (ZK) proofs. These are the only TF-secure protocols, but we can see that their cost is unreasonable.

PrivDB [37] and our new protocol Eff-pkDB^p have the same security properties. However, PrivDB is a bit less efficient on the prover side than Eff-pkDB^p and it has no light privacy-less variant, contrarily to Eff-pkDB^p.

Our lighter protocol Eff-pkDB and our first attempt Simp-pkDB in Appendix B are the most efficient public-key DB protocols as seen in Table 1. Eff-pkDB is secure against DF, MF, DH but it is not private. Simp-pkDB is secure only against DF, MiM and not private. It is more efficient than the Brand-Chaum protocol which has the same security level with Simp-pkDB. We focus on Eff-pkDB in the rest of the paper since it gives higher security level. Eff-pkDB's variant Eff-pkDB^p uses one extra encryption and it is strong private. We propose an instance of these protocols based on the Gap Diffie-Hellman (GDH) problem [30] in EC with a random oracle. The detailed efficiency analysis is presented in Section 6.

PaySafe [11] is very efficient but we do not compare it with the other protocols and our protocols since it assumes weaker adversarial model. It is only secure against MiM. It is not secure against DF, DH and TF because the response of the prover in the time critical phase which is a nonce picked by the prover does not depend on any message of the verifier. It also does not protect the privacy of the prover.

Our contributions are:

- We design two public-key DB protocols. The first protocol is secure against **DF, MF and DH** but it is not private. It uses **only one public key related operation** on the prover side. Basically, this protocol can be used in applications not requiring privacy in a very efficient way. Then, we modify this protocol by adding a public-key encryption to make it **strong private**. Both protocols are **quite efficient compared with the previous protocols**. Our constructions are generic based on a key agreement protocol, a weakly-secure symmetric DB protocols, and a cryptosystem. We formally prove the security following the model of Boureanu-Vaudenay [8] which was adapted to public-key DB in Vaudenay [37].
- We define a new key agreement (KA) security game (D-AKA). In literature, the extended Canetti-Krawczyk (eCK) security model [27] is widely accepted for KA. However, **a weaker security model (D-AKA) is sufficient** for the security of our new public-key DB protocols since we care both the efficiency and the security. Finally, we design a D-AKA secure key agreement protocol (Nonce-DH) based on the hardness of the GDH problem and a random oracle. The Nonce-DH key agreement protocol can be used in our DB constructions.

We show in Appendix B another reasonable protocol Simp-pkDB which was our first attempt to construct an efficient and a secure protocol. Although this protocol is quite efficient and does not require any public-key of a verifier, it fails in DH-security. This shows that it is hard to make a protocol which is secure for MiM, DF, and DH at the same time. Adding privacy in protocols is yet another challenge. Strong privacy cannot be achieved so easily as shown in Section 5.2. HPO and GOR failed to on this.

Organization of the paper: In Section 2, we give the formal definitions for the notion of DB and all necessary security definitions we are considering in our new protocols. In Section 3, we describe one time DB protocol OTDB [37] and give new security results on this protocol. OTDB and all the results about OTDB can be employed by Eff-pkDB or Eff-pkDB^p in a very efficient way. In Section 4, we introduce our new and weaker KA security model (D-AKA). Then, we construct a new KA protocol Nonce-DH which is D-AKA secure. We have Nonce-DH to show that both Eff-pkDB and Eff-pkDB^p can employ it and to make more precise efficiency analysis on these protocols. In Section 5, we introduce Eff-pkDB and Eff-pkDB^p with all security and privacy proofs. Finally, in Section 6, we do the efficiency and security analyses of all previous public-key DB protocols in detail.

Acknowledgements. This work was partly sponsored by the ICT COST Action IC1403 Cryptacus in the EU Framework Horizon 2020.

2 Definitions

The formalism in DB started by Avoine et al. [2]. Then, the first complete model was introduced by Dürholz et al. [15] where the threat models are defined according to the number of tainted time critical phase. The SKI model by Boureanu et al. [5–7] is another formal model which includes a clear communication model between parties in DB. The last model BV model [8] by Boureanu and Vaudenay is a more natural multi-party security model.

In this section, we give the definitions from the literature that we use in our security proofs.

2.1 Public Key Distance Bounding

Definition 1 (Public key DB Protocol [37]). *A public key distance bounding protocol is a two-party probabilistic polynomial-time (PPT) protocol and it consists of a tuple $(\mathcal{K}_P, \mathcal{K}_V, V, P, B)$. Here, $(\mathcal{K}_P, \mathcal{K}_V)$ are the key generation algorithms of P and V, respectively. The output of $\mathcal{K}_P$ is a secret/public key pair $(\mathsf{sk}_P, \mathsf{pk}_P)$ and similarly the output of $\mathcal{K}_V$ is a secret/public key pair $(\mathsf{sk}_V, \mathsf{pk}_V)$. P is the proving algorithm, V is the verifying algorithm where the inputs of P and V are from $\mathcal{K}_P$ and $\mathcal{K}_V$. B is the distance bound. $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$ and $V(\mathsf{sk}_V, \mathsf{pk}_V)$ interact with each other. At the end of the protocol, $V(\mathsf{sk}_V, \mathsf{pk}_V)$ outputs a final message Out_V and have pk_P as a private output. If $\mathsf{Out}_V = 1$, then V accepts. If $\mathsf{Out}_V = 0$, then V rejects.*

A public-key DB protocol is correct if and only if under honest execution, whenever a verifier $\mathcal{V}$ and a close (to $\mathcal{V}$) prover P run the protocol, then $\mathcal{V}$ always outputs $\mathsf{Out}_V = 1$ and pk_P.

Remark that this definition combines identification with DB: pk_P is not an input of the algorithm V, but it is an output. So, V learns the identity of P during the protocol.

We formalize the security notions of DB protocols. In the setting below, we have parties called provers, verifiers and other actors. Each party has instances and each instance I has its own location. It is called *close* to the instance J, if $d(I,J) \leq B$ and *far* from J, if $d(I,J) > B$ where d is a distance function.

An instance of an honest prover runs the algorithm denoted by $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$. An instance of a malicious prover runs an arbitrary algorithm denoted by P^*. The verifier is always honest and its instances run $V(\mathsf{sk}_V, \mathsf{pk}_V)$. Without loss of generality, we say that the other actors are malicious. They may run any algorithm.

The locations of the participants are elements of a metric space. We summarize the *communication and adversarial model* (See [5] for the details):

DB protocols run in natural communication settings. There is a notion of time, e.g. time-unit, a notion of measurable distance and a location. Besides, timed communication follows the laws of physics, e.g., communication cannot be faster than the speed of light. An adversary can see all messages (whenever they reach him). He can change the destination of a message subject to constraints.

This communication and adversarial model will only play a role in the DF and MiM security (defined below) but we will not have to deal with it. Indeed, we will start from an existing weakly secure symmetric DB protocol (such as OTDB [37]) and reduce the DF and MiM security of our protocol to the security of that protocol. So, we do not need to formalize more this model.

Now, we explain the security games for the distance fraud, mafia fraud and distance hijacking from [37].

Definition 2 (Distance fraud [37]). *The game begins by running the key setup algorithm $\mathcal{K}_V$ which outputs $(\mathsf{sk}_V, \mathsf{pk}_V)$. The game includes a verifier instance $\mathcal{V}$ and instances of an adversary. Given pk_V, the adversary generates $(\mathsf{sk}_P, \mathsf{pk}_P)$ with an arbitrary key setup algorithm $\mathcal{K}^*(\mathsf{pk}_V)$ (instead of $\mathcal{K}_P$). There is no participant close to $\mathcal{V}$. The adversary wins if $\mathcal{V}$ outputs $\mathsf{Out}_V = 1$ and pk_P. A DB protocol is DF-secure, if for any such game, the adversary wins with negligible probability.*

Definition 3 (Mafia fraud (MiM security) [37]). *The game begins by running the key setup algorithms $\mathcal{K}_V$ and $\mathcal{K}_P$ which output $(\mathsf{sk}_V, \mathsf{pk}_V)$ and $(\mathsf{sk}_P, \mathsf{pk}_P)$, respectively. The adversary receives pk_V and pk_P. The game consists of several verifier instances including a distinguished one $\mathcal{V}$, an honest prover P with its instances which are far away from $\mathcal{V}$ and an adversary with its instances at any location. The adversary wins if $\mathcal{V}$ outputs $\mathsf{Out}_V = 1$ and pk_P. A DB protocol is MiM-secure if for any such game, the probability of an adversary to win is negligible.*

Definition 4 (Distance hijacking [37]). *The game consists of several verifier instances $\mathcal{V}, V_1, V_2, \ldots$, a far away adversary P, and also honest prover instances $\mathsf{P}', \mathsf{P}'_1, \mathsf{P}'_2 \ldots$. A DB protocol $(\mathcal{K}_P, \mathcal{K}_V, V, P, B)$ having an initialization, a challenge and a verification phases is DH-secure if for all PPT algorithms $\mathcal{K}^*_P$ and $\mathcal{A}$, the probability of P to win the following game is negligible.*

- $\mathcal{K}_V \to (\mathsf{sk}_V, \mathsf{pk}_V)$, $\mathcal{K}_{P'} \to (\mathsf{sk}_{P'}, \mathsf{pk}_{P'})$.
- $\mathcal{K}^*_P(\mathsf{pk}_{P'}, \mathsf{pk}_V) \to (\mathsf{sk}_P, \mathsf{pk}_P)$ *and if* $\mathsf{pk}_P = \mathsf{pk}_{P'}$*, the game aborts. Then, instances of* P *run* $\mathcal{A}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V, \mathsf{pk}_{P'})$*,* $\mathsf{P}', \mathsf{P}'_1, \mathsf{P}'_2, ...$ *run* $P(\mathsf{sk}_{p'}, \mathsf{pk}_V)$*,* $\mathcal{V}, V_1, V_2, ...$ *run* $V(\mathsf{sk}_V, \mathsf{pk}_V)$*.*
- P *interacts with* $\mathsf{P}', \mathsf{P}'_1, \mathsf{P}'_2, ...$ *and* $\mathcal{V}, V_1, V_2, ...$ *during the initialization phase of* $\mathcal{V}$ *and* P' *concurrently.*
- P' *and* $\mathcal{V}$ *continue interacting with each other in their challenge phase and* P *remains passive even though he sees the exchanged messages.*
- P *interacts with* $\mathsf{P}', \mathsf{P}'_1, \mathsf{P}'_2, ...$ *and* $\mathcal{V}, V_1, V_2, ...$ *in the verification phase concurrently.*

The adversary wins if $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$ *and* pk_P*.*

The notion of initialization/challenge/verification phase is arbitrary but the notion of DH-security depends on this. To make it correspond to the notion in [12], the challenge phase must correspond to the time critical part where the verifier and the prover exchange challenge/response so fast that responses from far away would be rejected.

Definition 5 (HPVP Privacy Game [19]). *The privacy game is the following: Pick* $b \in \{0, 1\}$ *and let the adversary* $\mathcal{A}$ *play with the following oracles:*

- $\texttt{CreateP}(ID) \to P_i$: *It creates a new prover identity of* ID *and returns its identifier* P_i*.*
- $\texttt{Launch}() \to \pi$: *It launches a new protocol with the verifier* V_j *and returns the session identifier* π*.*
- $\texttt{Corrupt}(P_i)$: *It returns the current state of* P_i*. Current state means the all the values in* P_i*'s current memory. It does not include volatile memory.*
- $\texttt{DrawP}(P_i, P_j) \to vtag$: *It draws either* P_i *(if* $b = 0$*) or draws* P_j *(if* $b = 1$*) and returns the virtual tag reference vtag. If one of the provers was already an input of* $\texttt{DrawP} \to vtag'$ *query and* $vtag'$ *has not been released, then it outputs* $\emptyset$*.*
- $\texttt{Free}(vtag)$: *It releases vtag which means vtag can no longer be accessed.*
- $\texttt{SendP}(vtag, m) \to m'$: *It sends the message* m *to the drawn prover and returns the response* m' *of the prover. If vtag was not drawn or was released, nothing happens.*
- $\texttt{SendV}(\pi, m) \to m'$: *It sends the message* m *to the verifier in the session* π *and returns the response* m' *of the verifier. If* π *was not launched, nothing happens.*
- $\texttt{Result}(\pi) \to b'$: *It returns a bit that shows if the session* π *is accepted by the verifier (i.e the message* Out_V*).*

In the end of the game, the adversary outputs a bit g*. If* $g = b$*, then* $\mathcal{A}$ *wins. Otherwise, it loses.*

A DB protocol is strong private *if for all PPT adversaries, the advantage of winning the privacy game is negligible.*

We distinguish strong and weak privacy [33]. The weak privacy game does not include any '`Corrupt`' oracle. The other kind of classification is *wide* and *narrow* private. Wide privacy game is allowing to use the '`Result`' oracle while the narrow privacy game does not. In this paper, we implicitly consider wide privacy by making Out_V a protocol message, which means we always obtain this bit without using '`Result`' oracle.

2.2 Symmetric Distance Bounding

In this section, we give the useful definitions about the symmetric distance bounding that we need to use for our public key distance bounding protocols. Therefore, we do not explain all security notions for symmetric DB protocols.

Definition 6 (Symmetric DB Protocol [37]). *A symmetric distance bounding protocol is a two-party PPT protocol and it consists of a tuple* $(\mathcal{K}, V, P, B)$*. Here,* $\mathcal{K}$ *is the key generation algorithm,* P *is the proving algorithm and* V *is the verifying algorithm. The inputs of* P *and* V *is the output* s *of* $\mathcal{K}$*.* B *is the distance bound.* $P(s)$ *and* $V(s)$ *interact with each other. At the end of the protocol,* $V(s)$ *outputs a final message* Out_V*. If* $\mathsf{Out}_V = 1$*, then* V *accepts. If* $\mathsf{Out}_V = 0$*, then* V *rejects.*

A symmetric DB protocol is correct if and only if under honest execution, whenever a verifier $\mathcal{V}$ *and a close (to* $\mathcal{V}$*) prover* P *run the protocol, then* $\mathcal{V}$ *always outputs* $\mathsf{Out}_V = 1$*.*

Definition 7 (One Time DF (OT-DF) [37]). *The game begins by running a malicious key setup algorithm* K^* *which outputs* s*. It consists of a single verifier instance* $\mathcal{V}$ *running* $V(s)$ *and instances of an adversary* P^**.* P^* *receives* s*. There is no participant close to* $\mathcal{V}$*. The adversary wins if* $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$*. A symmetric DB protocol is OT-DF-secure, if for any such game, the adversary wins with negligible probability.*

Definition 8 (One Time MiM (OT-MiM) [37]). *The game begins by running the key setup algorithm* $\mathcal{K}$ *which outputs* s*. It consists of a single verifier instance* $\mathcal{V}$ *running* $V(s)$*, a single far away prover instance* P *running* $P(s)$ *and instances of an adversary. The adversary wins if* $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$*. A symmetric DB protocol is OT-MiM-secure, if for any such game, the probability that the adversary wins is negligible.*

Multi-verifier OT-MiM: The OT-MiM game with more than one verifier instance is called as *multi-verifier OT-MiM-security.* We defined this new notion to be able to have the result in Theorem 1 which helps us to prove the security of our constructions.

Definition 9 (One Time DH (OT-DH) [37]). *The game consists of a verifier instance* $\mathcal{V}$*, a far away adversary* P*, and also honest (and close) prover instance* P'*. A symmetric DB protocol* $(\mathcal{K}, V, P, B)$ *having an initialization, a challenge and a verification phases is OT-DH-secure if for all PPT algorithms* $\mathcal{A}, \mathcal{K}^*$*, the probability of* P *to win the following game is negligible.*

– *$\mathcal{K}^* \to s$, $\mathcal{K} \to s'$. Then, P' runs $P(s')$, $\mathcal{V}$ runs $V(s)$ and P runs $\mathcal{A}(s)$.*
– *P interacts with P' and $\mathcal{V}$ in their initialization phase concurrently.*
– *P' and $\mathcal{V}$ continue interacting with each other in their challenge phase and P remains passive even though he sees the exchanged messages.*
– *P interacts with P' and $\mathcal{V}$ in their verification phase concurrently.*

The adversary wins if $\mathcal{V}$ outputs $\mathsf{Out}_V = 1$.

Definition 10 (Multi-verifier Impersonation Fraud (IF) [3]). *The game begins by running the key setup algorithm $\mathcal{K}$ which outputs s. It consists of verifier instances running $V(s)$ and an adversary with no inputs. The adversary wins if any verifier instance outputs $\mathsf{Out}_V = 1$. A distance bounding protocol is multi-verifier IF-secure, if for any such game, the probability of an adversary to win is negligible.*

The above definition is with several verifiers, contrarily to others, because we will only use multi-verifier IF security.

MiM-security covers multi-verifier IF-security. So, if a DB protocol is MiM-secure, then it is multi-verifier IF-secure.

We will see in Theorem 2 that OT-MiM-security also implies multi-verifier IF-security for a DB following the canonical structure.

Definition 11 (Canonical Structure [37]). *A symmetric DB protocol $(\mathcal{K}, V, P, B)$ follows the canonical structure, if there exist an initialization/challenge/verification phases, P does not use s during the initialization phase, $\mathcal{V}$ does not use s at all except for computing the final Out_V, and the verification phase is not interactive.*

Remark that the notion of phase is used in DH and OT-DH security.

3 OTDB

As an example of one-time secure protocol, we can give the protocol OTDB by Vaudenay [37] which is a symmetric DB adapted from Hancke-Kuhn protocol [18]. The OTDB protocol follows the canonical structure (See Definition 11), only requires one xor operation before the challenge phase on the prover side and it is OT-DF, OT-MiM, multi-verifier OT-MiM and OT-DH secure [37]. (See Figure 1.) We complement these known results by showing multi-verifier OT-MiM security and multi-verifier IF-security.

Theorem 1. *OTDB is multi-verifier OT-MiM secure.*

Proof. Γ_0: In this game, an adversary $\mathcal{A}$ plays multi-verifier OT-MiM game. Here, we have a distinguished verifier instance $\mathcal{V}$ with other instances $\{V_1, ..., V_k\}$ and one prover instance P. The success probability of Γ_0 is p_0.

Γ_1 : We reduce Γ_0 to Γ_1 where at most one verifier instance outputs 1. Let's say E is an event in Γ_0 where at least two verifier instances output 1 ($\mathsf{Out}_V = 1$). To reduce Γ_0 to Γ_1, we show that $\Pr[E]$ is negligible.

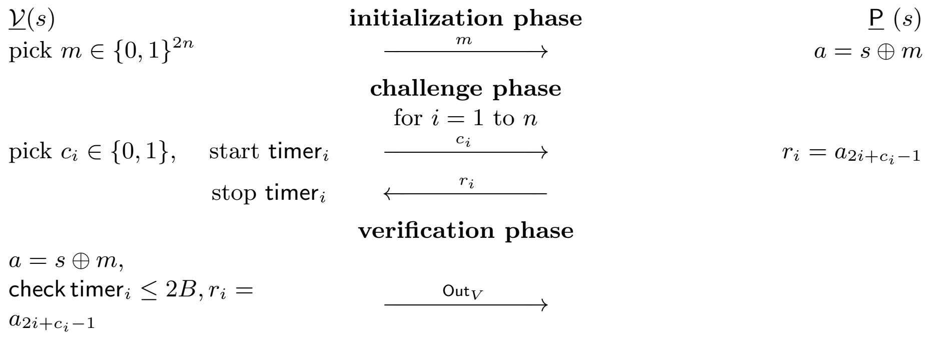
$\underline{\mathcal{V}}(s)$
initialization phase
$\underline{\mathsf{P}}\ (s)$
pick $m \in \{0,1\}^{2n}$
m
$a = s \oplus m$
challenge phase
for $i = 1$ to n
pick $c_i \in \{0,1\}$, start timer_i
c_i
$r_i = a_{2i+c_i-1}$
stop timer_i
r_i
verification phase
$a = s \oplus m$, $\mathsf{check\,timer}_i \leq 2B, r_i = a_{2i+c_i-1}$
Out_V

Fig. 1. OTDB

First, we define hybrid games $\Gamma_{i,j}$'s to analyze $\Pr[E]$. $\Gamma_{i,j}$ is similar to Γ_0 except the game stops right after V_i and V_j have sent their final outputs and all Out_V is replaced by 0 except V_i and V_j. The adversary wins the game if $\mathsf{Out}_{V_i} = \mathsf{Out}_{V_j} = 1$.

In $\Gamma_{i,j}$, we define three kinds of arrays for the challenges. The first array C_{V_i} includes the challenges sent by V_i, the second array C_{V_j} includes the challenges sent by V_j and the third array C_P includes the challenges seen by P. The bits in C_{V_i} and C_{V_j} are independent. We also define a response function $\mathsf{resp}_k(c) = a_{2k+c-1}$ for each round k. Since the bits of the secret s are independent, the bits of $\{\mathsf{resp}_k(0)||\mathsf{resp}_k(1)\}_{k=1}^{n}$ are independent as well. If $C_{V_i}[k] \neq C_{V_j}[k]$, then the adversary could have taken $C_P[k] = c$ where c is equal either $C_{V_i}[k]$ or $C_{V_j}[k]$ and learned $\mathsf{resp}_k(c)$. So, he responds correctly to either V_i or V_j for sure, but to the other instance with probability $\frac{1}{2}$. We define an event $E_{ij,k}$ where the responses are correct for V_i and V_j in round k. Clearly, all events $\{E_{ij,k}\}_{k=1}^{n}$ are independent. So, $\Gamma_{i,j} = \prod_k \Pr[E_{ij,k}]$. Hence,

$$\Pr[E_{ij,k}] \leq \Pr[C_{V_i}[k] = C_{V_j}[k]] + \Pr[E_{ij,k}|C_{V_i}[k] \neq C_{V_j}[k]] \\ \times \Pr[C_{V_i}[k] \neq C_{V_j}[k]] \leq \frac{3}{4}$$

So, the adversary wins $\Gamma_{i,j}$ with the probability $(\frac{3}{4})^n$ which is negligible.
Now, we can analyze E.

$$\Pr[E] \leq \sum_{i,j} \Pr[\Gamma_{i,j}] = \mathsf{negl}(n)$$

Since E happens with the negligible probability, we can reduce Γ_0 to Γ_1 and conclude $p_1 - p_0$ is negligible. For Γ_1 to succeed, only $\mathcal{V}$ must produce $\mathsf{Out}_V = 1$.

Γ_2 : We reduce Γ_1 to Γ_2 where we simulate all verifier instances except $\mathcal{V}$. We can do this simulation because the messages but Out_V sent by a verifier does not depend on the secret. Since $\mathsf{Out_V} = 0$ for all verifier instance except $\mathcal{V}$ in the winning case (only $\mathcal{V}$ can output 1), $p_1 \leq p_2$.

Now in Γ_2, we are in OT-MiM game where there is only one verifier instance $\mathcal{V}$ and one prover instance P. By using the OT-MiM-security result of OTDB [37], we deduce p_2 is negligible so p_0 is negligible. □

We prove the following result which will be used in Theorem 6.

Theorem 2. *If a (symmetric) DB protocol following the canonical structure is OT-MiM secure, then it is multi-verifier IF-secure.*

Proof. We take an adversary $\mathcal{M}$ playing the multi-verifier IF game. $\mathcal{M}$ interacts with polynomially many verifier instances V_j's. We define adversaries $\mathcal{A}_i$'s playing the OT-MiM game. $\mathcal{A}_i$ simulates $\mathcal{M}$ and takes the verifier instance V_i as $\mathcal{V}$ in the OT-MiM game. Concretely, we number the V_j's by their order of appearance during the simulation of $\mathcal{M}$. When $\mathcal{M}$ queries $V_1, ..., V_{i-1}, V_{i+1}, ..., V_k$ (where k is the total number of verifier instances), $\mathcal{A}_i$ just simulates them (this is possible since the protocol follows the canonical structure. So, no message from the verifier except Out_V depends on s). If Out_V needs to be returned to $\mathcal{M}$, $\mathcal{A}_i$ returns 0. When $\mathcal{M}$ queries V_i, $\mathcal{A}_i$ relays it to $\mathcal{V}$ and sends the response of $\mathcal{V}$ to $\mathcal{M}$.

Let E_i be the event in the multi-verifier IF game which is $\mathsf{Out}_{V_i} = 1$ and all previously released Out_V are equal to 0. Clearly, we have $\Pr[\mathcal{M}\,\mathsf{wins}] = \sum_{i\geq 1} \Pr[\mathcal{M}\,\mathsf{wins} \wedge E_i]$. On the other hand, $\Pr[\mathcal{M}\,\mathsf{wins} \wedge E_i] \leq \Pr[\mathcal{A}_i\,\mathsf{wins}]$ because for all coins making $\mathcal{M}$ win the multi-verifier IF-game and E_i occur at the same time, we have $\mathsf{Out}_{V_j} = 0$ for all $j < i$ and $\mathsf{Out}_{V_i} = 1$ so the same coins make $\mathcal{A}_i$ win the OT-MiM game. So, $\Pr[\mathcal{M}\,\mathsf{wins}] \leq \sum_{i\geq 1} \Pr[\mathcal{A}_i\,\mathsf{wins}]$. Due to OT-MiM security, $\Pr[\mathcal{A}_i\,\mathsf{wins}]$ is negligible for every i. So, $\Pr[\mathcal{M}\,\mathsf{wins}]$ is negligible. So, we have multi-verifier IF-security. □

Thanks to Theorem 2, OTDB is multi-verifier IF-secure.

4 Authenticated Key Agreement (AKA) Protocols

In this section, we show our new KA security model and some preliminaries about the AKA protocols. The security models in this section are used *to construct secure and private public-key DB protocols* in Section 5.

We note that the DB protocols we constructed in Section 5 can employ any eCK-secure [27] key agreement protocol to have the same security properties. However, eCK-security is stronger than we need in our protocols. Therefore, we define a weaker notion **to have simpler, more efficient and secure public-key DB.** Table 3 in Appendix A shows that Nonce-DH which is secure in our weaker model is more efficient than the previous KA protocols.

Definition 12 (AKA in one-pass). *A one-pass AKA protocol is a tuple* $(\mathsf{Gen}_A, \mathsf{Gen}_B, D, A, B)$ *of PPT algorithms. Let A and B be the two parties. A and B generate secret/public key pairs* $(\mathsf{sk}_A, \mathsf{pk}_A)$ *and* $(\mathsf{sk}_B, \mathsf{pk}_B)$ *with the algorithms* $\mathsf{Gen}_A(1^n)$ *and* $\mathsf{Gen}_B(1^n)$*, respectively where n is the security parameter. B picks N from the sampling algorithm D and runs* $B(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A, N)$ *which outputs the session key s. Then, (s)he sends N and finally, A gets the session key s by* *running* $A(\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B, N)$ *(See Figure 2). We say that AKA is correct, if A and B obtain the same s at the end of the protocol.*

$$\begin{tikzcd}
\underline{\mathbf{A}}(\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B) & & \underline{\mathbf{B}}(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A) \\
 & & N \leftarrow D(1^n) \\
A(\mathsf{sk_A}, \mathsf{pk_A}, \mathsf{pk_B}, \mathsf{N}) \to s & & B(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A, N) \to s \arrow[ll, "\mathsf{N}"']
\end{tikzcd}$$

Fig. 2. The structure of an authenticated key agreement (AKA) protocols in one pass.

Definition 13 (Decisional-Authenticated Key Agreement (D-AKA) security). *We define two oracles set up with* $\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{sk}_B, \mathsf{pk}_B$.

```
$\underline{\mathcal{O}_A(.,.):}$
  return $A(\mathsf{sk}_A, \mathsf{pk}_A, ., .)$

$\underline{\mathcal{O}_B(.):}$
  $N' \leftarrow D(1^n)$
  $s' \leftarrow B(\mathsf{sk}_B, \mathsf{pk}_B, ., N')$
  return $s', N'$
```

Given $b \in \{0,1\}$ *and the oracles* $\mathcal{O}_A(.,.), \mathcal{O}_B(.)$, *the game* $\mathsf{KA}^{d-aka}_{b,\mathcal{A}(n)}$ *is:*

1. *Challenger executes* $\mathsf{Gen}_A(1^n) \to (\mathsf{sk}_A, \mathsf{pk}_A), \mathsf{Gen}_B(1^n) \to (\mathsf{sk}_B, \mathsf{pk}_B)$, *sets up the oracles, calls* $\mathcal{O}_B(\mathsf{pk}_A) \to (s_0, N)$ *and picks* $s_1 \in \{0,1\}^n$. *Then, he sends* $s_b, N, \mathsf{pk}_B, \mathsf{pk}_A$ *to the adversary* $\mathcal{A}$.
2. $\mathcal{A}$ *has access to the oracle* $\mathcal{O}_B(.)$ *and* $\mathcal{O}_A(.,.)$ *under the condition of not querying the oracle* $\mathcal{O}_A$ *with the input* (pk_B, N). *Eventually,* $\mathcal{A}$ *outputs* b'.
3. *The advantage of the game is*

$$\mathsf{Adv}(\mathsf{KA}^{d-aka}_{\mathcal{A}(n)}) = \mathsf{Pr}[\mathsf{KA}^{d-aka}_{0,\mathcal{A}(n)} = 1] - \mathsf{Pr}[\mathsf{KA}^{d-aka}_{1,\mathcal{A}(n)} = 1].$$

A KA protocol ($\mathsf{Gen}_A(1^n), \mathsf{Gen}_B(1^n), D, A, B$*) is D-AKA secure if for all PPT algorithms* $\mathcal{A}$, $\mathsf{Adv}(\mathsf{KA}^{d-aka}_{\mathcal{A}(n)})$ *is negligible.*

We show that eCK-security implies D-AKA security in Theorem 8 in Appendix A. It means that our new public-key DB protocols can employ eCK-secure key agreement protocols as well.

Note that as a result of Lemma 1 in Appendix A, the probability that the same nonce is picked by the oracle B is negligible when we have D-AKA security.

Definition 14 (D-AKA^p privacy). *Given* $b \in \{0,1\}$ *and the oracle* $\mathcal{O}_A(.,.)$ *(defined in Definition 13), the game* $\mathsf{KA}^{d-aka^p}_{b,\mathcal{A}(n)}$ *is:*

1. *Challenger runs* $\mathsf{Gen}_A(1^n) \to (\mathsf{sk}_A, \mathsf{pk}_A)$ *and* $\mathsf{Gen}_B(1^n) \to (\mathsf{sk}_{B_1}, \mathsf{pk}_{B_1})$ *, sets up the oracle and gives* $\mathsf{pk}_A, \mathsf{pk}_{B_1}$ *and* sk_{B_1} *to* $\mathcal{A}$.
2. $\mathcal{A}$ *selects* sk_{B_0} *and* pk_{B_0} *and sends them to the challenger.*
3. *Challenger executes* $D(1^n) \to N$, $B(\mathsf{sk}_{B_b}, \mathsf{pk}_{B_b}, \mathsf{pk}_A^{\mathsf{sk}_{B_b}}, N) \to s$. *Then, he sends* s *to the adversary* $\mathcal{A}$.
4. $\mathcal{A}$ *has access to the oracle* $\mathcal{O}_A$. *Eventually,* $\mathcal{A}$ *outputs* b'. *(Remark that* $\mathcal{A}$ *does not know* N.*)*

5. The advantage of the game is

$$\mathsf{Adv}(\mathsf{KA}^{d-aka^p}_{\mathcal{A}(n)}) = \mathsf{Pr}[\mathsf{KA}^{d-aka^p}_{0,\mathcal{A}(n)} = 1] - \mathsf{Pr}[\mathsf{KA}^{d-aka^p}_{1,\mathcal{A}(n)} = 1].$$

A KA protocol ($\mathsf{Gen}_A(1^n), \mathsf{Gen}_B(1^n), D, A, B$) is D-AKA^p private if for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}(\mathsf{KA}^{d-aka^p}_{\mathcal{A}(n)})$ is negligible.

$$\begin{array}{lcr} \underline{\mathbf{A}}(\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B) & & \underline{\mathbf{B}}(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A) \\ & & \texttt{pick}\, N \in \{0,1\}^\ell, \\ H(g, \mathsf{pk}_B, \mathsf{pk}_A, \mathsf{pk}_B^{\mathsf{sk}_A}, N) \to s & \xleftarrow{\quad N \quad} & H(g, \mathsf{pk}_B, \mathsf{pk}_A, \mathsf{pk}_A^{\mathsf{sk}_B}, N) \to s \end{array}$$

Fig. 3. The Nonce-DH key agreement protocol.

A One-Pass AKA Protocol (Nonce-DH): We construct a D-AKA secure protocol (Nonce-DH) based on the Diffie-Hellman (DH) [14] as in Figure 3. Here g is a generator of a group of prime order q. g and q depend on a security parameter. The parties know each others' public keys beforehand where $\mathsf{pk}_A = g^{\mathsf{sk}_A}$ and $\mathsf{pk}_B = g^{\mathsf{sk}_B}$ and sk_A and sk_B are the corresponding secret keys which are uniformly picked in $\mathbb{Z}_q$.

The party B has input $(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A)$. He randomly picks N from $\{0,1\}^\ell$ and computes $B(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A, N) = H(g, \mathsf{pk}_B, \mathsf{pk}_A, \mathsf{pk}_A^{\mathsf{sk}_B}, N)$ to get s. The party A computes $A(\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B, N) = H(g, \mathsf{pk}_B, \mathsf{pk}_A, \mathsf{pk}_B^{\mathsf{sk}_A}, N)$ and gets s. Here, H is a deterministic function.

Clearly, Nonce-DH is correct since H is deterministic.

Theorem 3. *Assuming that the Gap Diffie-Hellman problem [30] is hard and $\ell = \Omega(n)$, Nonce-DH is D-AKA secure and D-AKA^p private in the random oracle model.*

The proof is in Appendix C.

5 Efficient Public Key Distance Bounding Protocol

In this section, we first introduce Eff-pkDB which is secure against DF, MF and DH and then Eff-pkDB^p a variant of it preserving the strong privacy as well.

5.1 Eff-pkDB

Eff-pkDB is constructed on an AKA in one-pass and a symmetric DB protocol. P and $\mathcal{V}$ first agree on a secret key s using an AKA protocol. Then, they together run a symmetric key DB protocol (symDB) by using s. Using OTDB as symDB and Using Nonce-DH as an AKA protocol will appear to be enough for its security.

Theorem 4. *If symDB is OT-DF-secure, then Eff-pkDB is DF-secure.*

$$\begin{tikzcd}
\underline{\mathcal{V}}(\mathsf{sk}_V, \mathsf{pk}_V) & & \underline{\mathsf{P}}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V) \\
 & & N \leftarrow D(1^n) \\
A(\mathsf{sk}_V, \mathsf{pk}_V, \mathsf{pk}_P, N) \to s & & B(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V, N) \to s \arrow[ll, "\mathsf{N,pk_P}"'] \\
 & {} \arrow[r, leftrightarrow, "\mathsf{symDB}(s)"] & {} \\
 & {} \arrow[r, "\mathsf{Out}_V"] & {}
\end{tikzcd}$$

Fig. 4. Public-key DB protocol based on D-AKA secure KA (Eff-pkDB)

Proof sketch: The malicious and far away prover with its instances play the DF game. We can easily reduce it to the game where $\mathcal{V}$ and the adversary receive the same s' from outside (even if maliciously selected). Since symDB is OT-DF-secure, the prover passes the protocol with negligible probability. $\square$

Theorem 5. *If symDB is multi-verifier OT-MiM-secure and the key agreement protocol with the algorithms* $\mathsf{Gen}_A, \mathsf{Gen}_B, A, B, D$ *is D-AKA secure then Eff-pkDB is MiM-secure.*

Proof. Γ_i is a game and p_i denotes the probability that Γ_i succeeds.

Γ_0 : The adversary plays the MiM game in Eff-pkDB with the distinguished verifier $\mathcal{V}$, $\mathcal{V}$'s instances and the prover instances. $\mathcal{V}$ receives pk_P and a given N. We call "matching instance" the instance who sends this N.

Γ_1 : We reduce Γ_0 to Γ_1 where no nonce produced by any prover instance is duplicated or equal to any nonce received by any a verifier instance before. Thanks to Lemma 1 in Appendix 1, $p_1 - p_0$ is negligible. So, the matching instance (if any) is unique and sets N before it is sent to $\mathcal{V}$.

Γ_2 : We simulate the prover instances and $\mathcal{V}$ as below in this game. Basically, in Γ_2, the prover and the verifier do not use the secret generated by the oracles $\mathcal{O}_B$ and $\mathcal{O}_A$, respectively.

```
P(.) (in $\Gamma_2$)
    run $\mathcal{O}_B(\mathsf{pk}_V) \to (s_0, N')$
    send $N', \mathsf{pk}_P$
    pick $s_1$
    store $(N', s_1, \mathsf{pk}_P)$ in $T$
    run symDB($s_1$)
```

```
V(.) (in $\Gamma_2$)
    receive $N', \mathsf{pk}_P$
    if $(N', ., \mathsf{pk}_P) \in T$
        retrieve $s$ from $T$
        where $(N', s, \mathsf{pk}_P) \in T$
    else:
        $s \leftarrow \mathcal{O}_A(\mathsf{pk}_P, N')$
    run symDB($s$)
```

With the reduction from Γ_1 to Γ_2, we show that the secret generated by A and B are indistinguishable from the randomly picked secret. The reduction is showed below:

We define the hybrid games $\Gamma_{2,t}$ to show $p_2 - p_1$ is negligible. Here, $t \in \{0, 1, 2, ..., k\}$ and k is the number of prover instances bounded by a polynomial.

$\Gamma_{2,i}$: $\mathcal{V}$ is simulated as in Γ_2 and the j^{th} instance of P is simulated as in Γ_2 for $j \leq i$ and as in Γ_1 for $j > i$. Clearly, $\Gamma_{2,0} = \Gamma_1$ and $\Gamma_{2,k} = \Gamma_2$.

First, we show that $\Gamma_{2,i}$ and $\Gamma_{2,i+1}$ are indistinguishable. For this, we use an adversary $\mathcal{B}$ that plays the D-AKA game. $\mathcal{B}$ receives $\mathsf{pk}_A, \mathsf{pk}_B, s_b, N$ from the

D-AKA challenger and simulates against the adversary $\mathcal{A}$ which distinguishes $\Gamma_{2,i}$ and $\Gamma_{2,i+1}$. $\mathcal{B}$ assigns $\mathsf{pk}_V = \mathsf{pk}_A$ and $\mathsf{pk}_P = \mathsf{pk}_B$. $\mathcal{B}$ simulates each prover P_j as described below.

```
$\underline{P_j(.)}$
    **if** $j \neq i+1$
        $\mathcal{O}_B(\mathsf{pk}_V) \to (s', N')$
        **if** $j \leq i$
            **pick** $s'$
    **else:**
        $s' \leftarrow s_b$ **and** $N' \leftarrow N$
    **if** $j \leq i+1$
        **store** $(N', s', \mathsf{pk}_P)$ **to** $T$
    **send** $N', \mathsf{pk}_P$
    **run** symDB$(s')$
```

Note that if $b = 0$ which means s_b is generated by the oracle B then $\mathcal{B}$ simulates the game $\Gamma_{2,i}$. Otherwise, he simulates $\Gamma_{2,i+1}$.

For the verifier simulation, $\mathcal{B}$ first checks, if $(N', ., \mathsf{pk}_P)$ is stored by himself as $\mathcal{V}$ in Γ_2. Otherwise, he sends (pk_P, N') to the oracle $\mathcal{O}_A$ and receives s'. Since (N, s_b, pk_P) is always stored in T, (pk_P, N) is not queried to $\mathcal{O}_A$ oracle. In the end of the game, $\mathcal{A}$ sends his decision. If $\mathcal{A}$ outputs i, then $\mathcal{B}$ outputs 0. If $\mathcal{A}$ outputs $i+1$, then $\mathcal{B}$ outputs 1. Clearly, the advantage of $\mathcal{B}$ is $p_{2,i} - p_{2,i+1}$. Due to the D-AKA security, we obtain that $p_{2,i} - p_{2,i+1}$ is negligible. From the hybrid theorem, we can conclude that $p_{2,0} - p_{2,k}$ is negligible where $p_{2,0} = p_1$ and $p_{2,k} = p_2$.

Γ_3 : We simulate the prover instances as below so that they do not run the oracle $\mathcal{O}_B$ to have N. The only change in this game is the generation of the nonce. Since the prover in Γ_3 picks the nonce from the same distribution that $\mathcal{O}_B$ picks, $p_3 = p_2$. This game shows that the prover generates N' (and also s_1) independently from $\mathcal{O}_B$.

```
$\underline{P(.)}$ (in $\Gamma_3$)
    **pick** $N' \in D(1^n)$
    **send** $N', \mathsf{pk}_P$
    **pick** $s_1$
    **store** $(N', s_1, \mathsf{pk}_P)$ **to** $T$
    **run** symDB$(s_1)$
```

Γ_4 : We reduce Γ_3 to the multi-verifier OT-MiM-security game Γ_4 where there is only matching instance and the other instances are simulated. With this final reduction, we show that the adversary has to break the multi-verifier OT-MiM-security of symDB in order to break the MiM-security of Eff-pkDB.

The reduction is the following. $\mathcal{A}^3$ plays the Γ_3 game. We construct an adversary $\mathcal{A}_i^4$ in Γ_4. $\mathcal{A}_i^4$ receives N from the matching prover in Γ_4. $\mathcal{A}_i^4$ takes P_i as a matching prover in Γ_3 where $i \in \{1, ..., k\}$. $\mathcal{A}_i^4$ simulates all of the provers except P_i against $\mathcal{A}^3$. For P_i, $\mathcal{A}_i^4$ just sends (pk_P, N). In the end, if P_i is the matching instance in Γ_3 and $\mathcal{A}^3$ wins then $\mathcal{A}_i^4$ wins. Therefore $p_3 \leq \sum_i p_{4,i}$ where $p_{4,i}$ is the probability that $\mathcal{A}_i^4$ wins. Due to multi-verifier OT-MiM-security, all $p_{4,i}$'s are negligible. So, p_3 is negligible. Hence, p_0 is negligible. $\square$

Theorem 6. *If symDB is OT-MiM-secure, OT-DH-secure and follows the canonical structure and if the key agreement protocol with the algorithms* $\mathsf{Gen}_A, \mathsf{Gen}_B, A, B, D$ *is D-AKA secure then Eff-pkDB is DH-secure.*

Proof. Γ_i is a game and p_i denotes the probability that Γ_i succeeds.

Γ_0 : The adversary P with its instances plays the DH-security game in Eff-pkDB with the distinguished verifier $\mathcal{V}$ and its instances and an honest prover P'. The probability that the adversary succeeds in Γ_0 is p_0.

Γ_1 and Γ_2 : These games are like in the proof of Theorem 5 except that P_j is replaced by P'_j. The reduction from Γ_0 to Γ_1 and Γ_2 is similar to the proof of Theorem 5. So we can conclude that $p_2 - p_0$ is negligible.

We let N be the nonce produced by the instance of P' and s_1 be its key which is playing a role during the challenge phase of $\mathcal{V}$ in the DH game.

We reduce Γ_2 to Γ_3 in which all Out_V from a verifier instance who receives pk_P and N is replaced by 0 during the initialization phase. Intuitively, in this case, Out_V cannot be equal 1 because if it is 1, it means P' impersonates P. The reduction is as follows: During the initialization game, P' sends messages which do not depend on s_1 because of the canonical structure, and which can be simulated. So, we can reduce this phase to the multi-verifier IF game and use Theorem 2 to show that $p_3 - p_2$ is negligible. This reduction shows that the DH-adversary P cannot win the game with sending pk_P and N generated by P'.

We reduce Γ_3 to Γ_4 where the game stops after the challenge phase for $\mathcal{V}$. Since the verification phase which is after the challenge phase is non-interactive and Out_V is determined at the end of the challenge phase, $p_4 = p_3$.

We reduce Γ_4 to Γ_5 which is OT-DH game. In Γ_4, s_1 has never been used so s (the key of $\mathcal{V}$ which is given by the adversary) is independent from s_1. In this case, P' and $\mathcal{V}$ run symDB with independent secrets. So, $p_5 = p_4$. Because of the OT-DH security of symDB, p_5 is negligible. $\square$

5.2 Eff-pkDB^p

Eff-pkDB is not strong private as the public key of the prover is sent in clear. Adding one encryption operation to Eff-pkDB is enough to have strong privacy.

Eff-pkDB^p in Figure 5 is the following: The prover and the verifier generate their secret/public key pairs by running the algorithms $\mathsf{Gen}_P(1^n)$ and $\mathsf{Gen}_V(1^n)$, respectively. We denote $(\mathsf{sk}_P, \mathsf{pk}_P)$ for the secret/public key pair of the prover and $(\mathsf{sk}_V, \mathsf{pk}_V)$ for the secret/public key pair of the verifier where $\mathsf{sk}_V = (\mathsf{sk}_{V_1}, \mathsf{sk}_{V_2})$ and $\mathsf{pk}_V = (\mathsf{pk}_{V_1}, \mathsf{pk}_{V_2})$ and the first key is used for the encryption and the second key is used for the AKA protocol. The prover picks N from the sampling algorithm D and generates s with the algorithm $B(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_{V_2}, N)$. Then, he encrypts pk_P and N with pk_{V_1}. After, he sends the ciphertext e to the verifier. The verifier decrypts e with sk_{V_1} and learns N and pk_P which helps him to understand who is interacting with him. Next, the verifier runs $A(\mathsf{sk}_{V_2}, \mathsf{pk}_{V_2}, \mathsf{pk}_P, N)$

and gets s. Finally, the prover and verifier run a symmetric DB protocol symDB protocol with s.

$$\begin{array}{lcr}
\mathcal{V}(\mathsf{sk}_V, \mathsf{pk}_V) & & \mathsf{P}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V) \\
 & & N \leftarrow D(1^n) \\
 & & B(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_{V_2}, N) \to s \\
\mathsf{pk}_P, N = \mathsf{Dec}_{\mathsf{sk}_{V_1}}(e) & \xleftarrow{\quad e \quad} & e = \mathsf{Enc}_{\mathsf{pk}_{V_1}}(\mathsf{pk}_P, N) \\
A(\mathsf{sk}_{V_2}, \mathsf{pk}_{V_2}, \mathsf{pk}_P, N) \to s & & \\
 & \xleftrightarrow{\mathsf{symDB}(s)} & \\
 & \xrightarrow{\quad \mathsf{Out}_V \quad} &
\end{array}$$

Fig. 5. Eff-pkDB^p: private variant of Eff-pkDB

Assuming that the AKA protocol is D-AKA secure and symDB is OT-X secure symmetric key DB protocol for all $X \in \{DF, MiM, DH\}$ and follows canonical structure, we can easily show that Eff-pkDB^p is X-secure from Theorem 4, 5, 6. To prove this, we start from an adversary playing the X-security game against Eff-pkDB^p. We construct an adversary playing the same game against Eff-pkDB to whom we give sk_{V_1}. The simulation is straightforward.

Theorem 7. *Assuming the key agreement protocol is D-AKA^p secure and the cryptosystem is IND-CCA secure, then the Eff-pkDB^p is strong private in the HPVP model (Definition 5).*

Proof. Γ_i is a game and p_i denotes the probability that Γ_i succeeds.

Γ_0 : The adversary $\mathcal{A}$ plays the HPVP privacy game.

Γ_1 : The verifiers skip the decryption when they receive a ciphertext produced by any prover and continue with the values encrypted by the prover. Because of the correctness of the encryption scheme $p_1 = p_0$.

Γ_2 : This game is the same with Γ_1 except the provers encrypt a random string instead of pk_P, N. The verifier retrieves e and s from the table T so that it does not decrypt any ciphertext that comes from a prover as in Γ_1. Thanks to the IND-CCA security (Verifiers are simulated using a decryption oracle due to our Γ_1 reduction. The use of this oracle is valid in IND-CCA game), $p_2 - p_1$ is negligible. So, P and $\mathcal{V}$ works as follows:

```
P(.) (in $\Gamma_2$)
    pick $N \in D(1^n)$
    $s \leftarrow B(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_{V_2}, N)$
    pick $r$
    $e \leftarrow \mathsf{Enc}_{\mathsf{pk}_{V_1}}(r)$
    store $(e, s)$ to $T$
    send $e$
    run symDB($s$)

V(.) (in $\Gamma_2$)
  receive $e$
  if $(e, .) \in T$
    retrieve $s$ from $T$
    where $(e, s) \in T$
  else:
    $(\mathsf{pk}', N) \leftarrow \mathsf{Dec}_{\mathsf{sk}_{V_1}}(e)$
    $s \leftarrow A(\mathsf{sk}_{V_2}, \mathsf{pk}_{V_2}, \mathsf{pk}', N)$
  run symDB($s$)
```

This reduction shows that the adversary cannot retrieve pk_P and N from the encryption.

Γ_3 : It is the same with Γ_3 except that we simulate the prover as below. In this game, s is generated independently from sk_P and pk_P.

```
$\underline{P(.)}$ (in $\Gamma_3$)
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}_B(1^n)$
    **pick** $N \in D(1^n)$
    **run** $s \leftarrow B(\mathsf{sk}, \mathsf{pk}, \mathsf{pk}_{\mathsf{V}_2}, \mathsf{N})$
    **pick** $r$
    $e \leftarrow \mathsf{Enc}_{\mathsf{pk}_{V_1}}(r)$
    **store** $(e, s)$ **to** $T$
    **send** $e$
    **run** symDB$(s)$
```

We defined the hybrid games $\Gamma_{3,t}$ to show $p_3 - p_2$ is negligible. Here, $t \in \{0, 1, 2, ..., k\}$ and k is the number of prover instances bounded by a polynomial.

$\Gamma_{3,i}$: $\mathcal{V}$ is simulated as in Γ_3 and the j^{th} instance of P is simulated as in Γ_3 if $j \leq i$ and as in Γ_2 if $j > i$.

First, we show that $\Gamma_{3,i}$ and $\Gamma_{3,i+1}$ are indistinguishable. For this, we use an adversary $\mathcal{B}$ that plays D-AKA^p game. $\mathcal{B}$ receives $\mathsf{pk}_A, \mathsf{pk}_{B_1}$ and sk_{B_1} from the D-AKA^p challenger, picks $(\mathsf{sk}_{B_0}, \mathsf{pk}_{B_0})$ and sends them to the challenger. Finally, $\mathcal{B}$ receives s. After, he begins simulating against the adversary $\mathcal{A}$ that wants to distinguish $\Gamma_{3,i}$ and $\Gamma_{3,i+1}$.

```
$\underline{P_{i+1}(.)}$
    **pick** $r$
    $e \leftarrow \mathsf{Enc}_{\mathsf{pk}_V}(r)$
    **store** $(e, s)$ **to** $T$
    **send** $e$
    **run** symDB$(s)$
```

$\mathcal{B}$ assigns $\mathsf{pk}_V = \mathsf{pk}_A$ and $\mathsf{pk}_P = \mathsf{pk}_{B_1}$. For all of the prover simulations, if $j \neq i+1$, P_j is simulated normally. $\mathcal{V}$ is simulated using the $\mathcal{O}_A$ oracle. `Corrupt` can be simulated since sk_{B_1} is available.

Note that if s is generated from $B(\mathsf{sk}_{B_0}, \mathsf{pk}_{B_0}, \mathsf{pk}_V, N)$ then $\mathcal{B}$ simulates $\Gamma_{3,i+1}$ and if it is generated from $B(\mathsf{sk}_{B_1}, \mathsf{pk}_{B_1}, \mathsf{pk}_V, N)$ then $\mathcal{B}$ simulates $\Gamma_{3,i}$.

For the verifier simulation, $\mathcal{B}$ first checks if $(e, .)$ is stored by himself as $\mathcal{V}$ in Γ_3. Otherwise, he decrypts e and sends (pk_{P_j}, N) to the oracle $\mathcal{O}_A(\mathsf{pk}_P, N)$ and receives s. In the end of the game, $\mathcal{A}$ sends his decision. If $\mathcal{A}$ outputs i, then $\mathcal{B}$ outputs 1. If $\mathcal{A}$ outputs $i+1$, then $\mathcal{B}$ outputs 0. Clearly, the advantage of $\mathcal{B}$ is $p_{3,i} - p_{3,i+1}$ which is negligible because of the D-AKA^p assumption. From the hybrid theorem, we can conclude that $p_{3,0}$ and $p_{3,k}$ is negligible where $p_{3,0} = p_2$ and $p_{3,k} = p_3$.

Now, in Γ_3, no identity is used by the provers. Hence, $\mathcal{A}$ does not have any advantage to guess the prover which means $p_3 = \frac{1}{2}$. As a result of it, $p_0 - \frac{1}{2}$ is negligible.

Consequently, if we use D-AKA secure and D-AKA^p private key agreement protocol in Eff-pkDB^p, then we have DF, MF, DH secure and strong private public-key DB protocol. For instance, Nonce-DH key agreement protocol is a good candidate for Eff-pkDB^p.

Difficulties of having strong privacy: The strong privacy is the hardest privacy notion to achieve in DB protocols. Sending all prover messages with an IND-CCA secure encryption is not always enough to have strong privacy. We exemplify our argument as follows: Clearly, Eff-pkDB protocol is still DF-MiM and DH-secure, if we replace the nonce selection by a counter. So, we can make a new version of Eff-pkDB^p based on the counter version of Eff-pkDB where the prover sends his public key and the counter by an IND-CCA encryption. However, clearly, it does not give strong privacy because when an adversary calls `Corrupt` oracle, he learns the counter of two drawn provers. Since the adversary knows the corresponding secret keys for both of them, he can easily differentiate the drawn provers based on the counter. This attack is not possible in Eff-pkDB^p which uses a nonce instead of a counter because the nonce is in the volatile memory. So, the adversary does not learn it with the `Corrupt` oracle.

6 Conclusion

Our main purpose in this work was to design an efficient and a secure public-key DB protocol. First, we designed Eff-pkDB which is secure against DF, MiM and DH. We did not consider privacy in this one because privacy is not the main concern of some applications. Therefore, Eff-pkDB can be employed by the applications that do not need privacy. Eff-pkDB is one of the most efficient public key DB protocols compared to the previous ones (See Table 2).

Second, we added strong privacy to the Eff-pkDB protocol and obtained Eff-pkDB^p. We succeeded it by adding one public-key IND-CCA secure encryption. In this case, the protocol is not as efficient as before but still one of the most efficient ones with the same security and privacy properties.

In Table 2, we give the security properties of existing public-key DB protocols along with the number of computations done on prover side. We use the number of elliptic curve multiplications and hashing as a metric in our efficiency analysis. We exclude GOR, ProProx and eProProx (in Table 1) since they clearly require a lot more computation than the other public-key DB protocols. In our counting for the number of computations in Table 2, 1 commitment is counted as 1 hashing operation. For the signature, we prefer an efficient and existentially unforgeable under chosen-message attacks resistant signature scheme ECDSA [21]. ECDSA requires 1 EC multiplication, 1 mapping, 1 hashing, 1 modular inversion and 1 random string selection. For the IND-CCA encryption scheme, we use ECIES [31] which requires 2 EC multiplications, 1 KDF, 1 symmetric key encryption, 1 MAC and 1 random string selection. For the D-AKA secure key agreement protocol, we use Nonce-DH which requires 1 EC multiplication, 1 hashing and 1 random string selection.

We first compare the protocols considering the security and the efficiency trade-off. Eff-pkDB and Simp-pkDB are the most efficient ones. However, Simp-pkDB is secure only against MiM and DF. After Eff-pkDB, the second most

Protocol	**Security**	**Privacy**	**PK operations**	**Number of Computations**
Brands-Chaum [9]	MiM, DF	No Privacy	1 commitment, 1 signature	1 EC multiplication, 2 hashings, 1 mapping, 1 modular inversion, 1 random string selection
HPO [20]	MiM, DF	Weak Private		4 EC multiplications, 2 random string selections, 2 mappings
PrivDB [37]	MiM, DF, DH	Strong Private	1 signature, 1 IND-CCA encryption	3 EC multiplications, 2 hashings, 2 random string selection, 1 modular inversion, 1 mapping, 1 symmetric key encryption, 1 MAC
Simp-pkDB	MiM, DF	No Privacy	1 decryption	1 EC multiplication, 1 hashing, 1 symmetric key decryption, MAC
Eff-pkDB	MiM, DF, DH	No privacy	1 D-AKA secure KA protocol	1 EC multiplication, 1 hashing, 1 random string selection
Eff-pkDB^p	MiM, DF, DH	Strong Private	1 IND-CCA Encryption, 1 D-AKA secure KA protocol	3 EC multiplications, 2 hashings, 2 random string selections, 1 symmetric key encryption, 1 MAC

Table 2. The review of the existing public-key DB protocols.

efficient protocol is Brands-Chaum protocol [9] but this protocol is only secure against MiM and DF while Eff-pkDB is secure against DH as well.

Now, we compare the protocols considering security, privacy and efficiency trade-off. In this case, HPO requires 4 EC multiplications while PrivDB and Eff-pkDB^p require 3 EC multiplications and 1 hashing. Hashing is more efficient than elliptic curve multiplication so it looks like PrivDB and Eff-pkDB^p are more efficient. However, HPO has an advantage in efficiency if it is used in a dedicated hardware allowing only EC operations. On the other hand, Eff-pkDB^p and PrivDB are secure against MiM, DF, DH and strong private while HPO is only MiM and DF secure and only private.

Eff-pkDB^p and PrivDB have the same security and privacy properties and almost the same efficiency level. However, if we analyze the efficiency with more metrics, we see that PrivDB requires extra 1 modular inversion and 1 mapping. More importantly, Eff-pkDB^p has lighter version Eff-pkDB which can be used efficiently in the applications which do not need privacy.

One of the important useful property of Eff-pkDB is that it can employ any D-AKA secure key agreement protocol to satisfy DF, MiM and DH security.

References

1. EMVCo version 2.6 in book c-2 kernel 2 specification.
2. G. Avoine, M. A. Bingöl, S. Kardaş, C. Lauradoux, and B. Martin. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security*, 19(2):289–317, 2011.
3. G. Avoine and A. Tchamkerten. An efficient distance bounding RFID authentication protocol: Balancing false-acceptance rate and memory requirement. In *Information Security*, LNCS 5735, pages 250–261. Springer, 2009.

4. S. Bengio, G. Brassard, Y. G. Desmedt, C. Goutier, and J.-J. Quisquater. Secure implementation of identification systems. *Journal of Cryptology*, 4(3), 1991.
5. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Secure and lightweight distance-bounding. In *LightSec*, LNCS 8162, pages 97–113. Springer, 2013.
6. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Towards secure distance bounding. In *Fast Software Encryption*, LNCS 8424, pages 55–67. Springer, 2013.
7. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical and provably secure distance-bounding. IOS Press, 2015.
8. I. Boureanu and S. Vaudenay. Optimal proximity proofs. In *Inscrypt*, LNCS 8957, pages 170–190. Springer, 2014.
9. S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *EUROCRYPT*, LNCS 765, pages 344–359. Springer-Verlag, 1993.
10. L. Bussard and W. Bagga. Distance-bounding proof of knowledge to avoid real-time attacks. In *Security and Privacy in the Age of Ubiquitous Computing*, volume 181 of *IFIP AICT*, pages 223–238. Springer, 2005.
11. T. Chothia, F. D. Garcia, J. de Ruiter, J. van den Breekel, and M. Thompson. Relay cost bounding for contactless EMV payments. In *Financial Cryptography and Data Security*, pages 189–206. Springer, 2015.
12. C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *SP*, pages 113–127, 2012.
13. Y. Desmedt. Major security problems with the unforgeable (Feige-) Fiat-Shamir proofs of identity and how to overcome them. In *SECURICOM*, pages 147–159, 1988.
14. W. Diffie and M. E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.
15. U. Dürholz, M. Fischlin, M. Kasper, and C. Onete. A formal approach to distance-bounding rfid protocols. In *International Conference on Information Security*, pages 47–62. Springer, 2011.
16. M. Fischlin and C. Onete. Terrorism in distance bounding: modeling terrorist-fraud resistance. In *ACNS*, LNCS 7954, pages 414–431. Springer, 2013.
17. S. Gambs, C. Onete, and J.-M. Robert. Prover anonymous and deniable distance-bounding authentication. In *ASIA CCS*, ACM Symposium, pages 501–506, 2014.
18. G. P. Hancke and M. G. Kuhn. An RFID distance bounding protocol. In *SecureComm 2005*, pages 67–73. IEEE, 2005.
19. J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel. A new RFID privacy model. In *ESORICS*, LNCS 6879, pages 568–587. Springer, 2011.
20. J. Hermans, R. Peeters, and C. Onete. Efficient, secure, private distance bounding without key updates. In *WiSec*, pages 207–218, 2013.
21. D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63, 2001.
22. H. Kılınç and S. Vaudenay. Comparison of public-key distance bounding protocols. *under submission*.
23. H. Kılınç and S. Vaudenay. Optimal proximity proofs revisited. In *ACNS*, LNCS 9092, pages 478–494. Springer, 2015.
24. C. H. Kim and G. Avoine. Rfid distance bounding protocol with mixed challenges to prevent relay attacks. In *CANS*, pages 119–133. Springer, 2009.
25. C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira. The swiss-knife rfid distance bounding protocol. In *ICISC*, pages 98–115. Springer, 2009.
26. H. Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In *CRYPTO*, pages 546–566. Springer, 2005.

27. B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *Provable Security*, pages 1–16. Springer, 2007.
28. K. Lauter and A. Mityagin. Security analysis of kea authenticated key exchange protocol. In *Public Key Cryptography-PKC 2006*, pages 378–394. Springer, 2006.
29. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003.
30. T. Okamoto and D. Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *PKC*, pages 104–118. Springer, 2001.
31. V. Shoup. A proposal for an ISO standard for public key encryption (2.0), 2001.
32. B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, 46(3):329–342, 2008.
33. S. Vaudenay. On privacy models for RFID. In *ASIACRYPT*, LNCS 4833, pages 68–87. Springer, 2007.
34. S. Vaudenay. On modeling terrorist frauds. In *Provable Security*, LNCS 8209, pages 1–20. Springer, 2013.
35. S. Vaudenay. On privacy for RFID. In *Provable Security*, pages 3–20. Springer, 2015.
36. S. Vaudenay. Privacy failure in the public-key distance-bounding protocol. *IET Information Security*, 2015.
37. S. Vaudenay. Private and secure public-key distance bounding: application to NFC payment. In *FC*, volume 8975, pages 207–216. Springer, 2015.
38. S. Vaudenay. Sound proof of proximity of knowledge. In *Provable Security*, pages 105–126. Springer, 2015.

A More results about D-AKA security model

KA Protocol	Efficiency	Security
MQV [29]	2.5	unproven
HMQV [26]	2.5	CK
KEA+ [28]	3	CK
NAXOS [27]	4	eCK
CMQV [32]	3	eCK
Nonce-DH	1	D-AKA

Table 3. Existing KA protocols with their security and efficiency. Efficiency column shows the number of exponentiation done by per party.

The Extended Canetti-Krawczyk (eCK) Security Model [27]

The eCK security model consists of t parties with their certificated public keys. The key exchange protocol is executed between two parties A and B. When A starts a key exchange protocol with B, it is called as a session and A is the owner of the session and B is the peer. A (initiator) starts the protocol by sending a message $\mathsf{M_A}$, then B (responder) responds with a message $\mathsf{M_B}$. The session id *sid* corresponds to an instance of A or B.

There is a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ controlling all communication and some instances. The activation of the parties starts by `Send`$(A, B,$message) (or `Send`$(B, A,$message)). Besides `Send`, $\mathcal{A}$ can do following queries:

– `Long-Term Key Reveal`(A): Outputs the long term public-key of A.
– `Ephemeral Key Reveal(`sid`)` Outputs an ephemeral key of a session sid.
– `Reveal(`sid`)`: Outputs the session key of a completed session sid.
– `Test(`sid`)`: If sid is clean then outputs $s \leftarrow \texttt{Reveal}(sid)$ if $b = 1$, outputs $s \leftarrow \{0, 1\}^{\lambda}$ if $b = 0$ (λ is the size of the session key).
The advantage is the difference of the probability that $\mathcal{A}$ gives 1 for $b = 0$ and $b = 1$.

A clean session is basically a session where winning the game for $\mathcal{A}$ is not trivial. See [27] for more details.

Theorem 8. *If a key agreement protocol is eCK secure [27], then it is D-AKA secure.*

Proof. Let's assume that there is an adversary $\mathcal{A}$ playing D-AKA game. We construct an adversary $\mathcal{B}$ simulating the D-AKA game and playing the eCK game. $\mathcal{B}$ receives all the public keys in the eCK game. $\mathcal{B}$ first picks two parties A and B. Then, he creates a session sid between them by sending the query Send(A,B, message) and he assigns the ephemeral public key of B as a nonce N. Then, he sends the query Test(sid) and receives s_b. Finally, he sends $s_b, N, \mathsf{pk}_B, \mathsf{pk}_A$ to $\mathcal{A}$. Whenever $\mathcal{A}$ calls the oracle $\mathcal{O}_B(\mathsf{pk}_{A'})$, $\mathcal{B}$ creates a new session sid' with A' on behalf of B as explained above. Similarly, he assigns the ephemeral public key of B as a nonce N'. After, he sends the query Reveal(sid') and receives the session key s'. As a response of $\mathcal{O}_B(\mathsf{pk}_{A'})$, he sends s', N' to $\mathcal{A}$. In addition, whenever $\mathcal{A}$ calls the oracle $\mathcal{O}_A(\mathsf{pk}_{B'}, N'')$, first, $\mathcal{B}$ checks if $(\mathsf{pk}_{B'}, N'')$ equals (pk_B, N). If it is not equal, he creates a new session sid'' on behalf of B' with the ephemeral public key N'' and calls the oracle Reveal(sid'') to receive the session key s''. Then, he responds to $\mathcal{A}$ with s''. In the end, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. The simulation of D-AKA game is perfect. So the advantage of $\mathcal{B}$ equals to the advantage of $\mathcal{A}$. Therefore, since the advantage of $\mathcal{B}$ is negligible, the advantage of $\mathcal{A}$ is negligible as well. □

As a result of Theorem 8, we can conclude any eCK secure key agreement protocol can be used in Eff-pkDB. However, we suggest using D-AKA secure key agreement protocols since they may require less public-key operations.

Lemma 1. *We consider D-AKA secure key agreement protocol* $(\mathsf{Gen}_A, \mathsf{Gen}_B, D, A, B)$. *We define the random variables* $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow \mathsf{Gen}_A(1^n)$, $(\mathsf{sk}_B, \mathsf{pk}_B) \leftarrow \mathsf{Gen}_B(1^n)$, *and* $(s, N) \leftarrow \mathcal{O}_B(\mathsf{pk}_A)$ *and* $(s', N') \leftarrow \mathcal{O}_B(\mathsf{pk}_A)$. *We have that* $\Pr[N = N']$ *is negligible. Furthermore, for all values u which could depend on* $\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{sk}_B, \mathsf{pk}_B$, $\Pr[N = u]$ *is negligible.*

Proof. We define an adversary $\mathcal{A}$ playing the D-AKA game as follows:

```
$\underline{\mathcal{A}}$
    receive $s_b, N, \mathsf{pk}_B, \mathsf{pk}_A$
    $(s', N') \leftarrow \mathcal{O}_B(\mathsf{pk}_A)$
    if $N' = N$
        if $s' = s_b$
            output $0$
        else:
            output $1$
    else:
        output $b' \leftarrow_r \{0, 1\}$
```

In this strategy, $\mathcal{A}$ wins if $N = N'$ (except $s_1 = s_0$ and $b = 1$). Otherwise, he wins with $\frac{1}{2}$ probability.

$$\begin{aligned}\Pr[\mathcal{A}\,\mathsf{win}] =& \frac{1}{2}(1 - \Pr[N = N']) + \Pr[N = N'] - \Pr[N = N', s_1 = s_0, b = 1] \\ =& \frac{1}{2} + \frac{1}{2}\Pr[N = N'] - \Pr[N = N', s_1 = s_0, b = 1]\end{aligned}$$

We know from the D-AKA security that $\Pr[\mathcal{A}\,\mathsf{win}] - \frac{1}{2}$ is negligible. $\Pr[s_1 = s_0] = 2^{-n}$ is negligible as well. So, $\Pr[N = N']$ is negligible. Now, we need to show that it holds for all values u.

Let v be the most probable value for N. We have

$$\begin{aligned}\Pr[N = N'] =& \sum_w \Pr[N = N' = w] \\ =& \sum_w \Pr[N = w]^2 \\ \geq& \Pr[N = v]^2\end{aligned}$$

So, we have the following inequality in the end:

$$\Pr[N = u] \leq \Pr[N = v] \leq \sqrt{\Pr[N = N']}$$

We know that $\Pr[N = N']$ is negligible so $\Pr[N = u]$ is negligible.

□

B Mafia and Distance Fraud Secure Public Key DB

We consider the Simp-pkDB protocol in Figure 6. In Simp-pkDB the prover P selects a nonce $N \in \{0, 1\}^n$ where n is security parameter and sends it to the verifier together with pk. Then verifier V selects a secret $s \in \{0, 1\}^n$, encrypts it with N by the public key pk of the prover and sends the encryption e to P. After receiving e, P decrypts it with the secret key sk and gets s, N. If the N is the nonce by P, then they run one-time secure $\mathsf{symDB}(s)$.

We show that this protocol is MiM-secure but not DH-secure. Simp-pkDB requires only one operation which is IND-CCA decryption.

$$\begin{array}{lcr}
\mathsf{V} & & \mathsf{P}(\mathsf{pk},\mathsf{sk}) \\
\text{pick } s \in \{0,1\}^{\ell} & \xleftarrow{\quad N,\mathsf{pk}\quad} & \text{pick } N \in \{0,1\}^{\ell} \\
e = \mathsf{Enc}_{\mathsf{pk}}(s||N) & \xrightarrow{\quad e \quad} & s, N = \mathsf{Dec}_{\mathsf{sk}}(e) \\
 & & \mathsf{Verify}(N) \\
 & \xleftrightarrow{\quad \mathsf{symDB}(s) \quad} & \\
 & \xrightarrow{\quad \mathsf{Out}_V \quad} &
\end{array}$$

Fig. 6. Simp-pkDB

Theorem 9. *If symDB is DF-secure then Simp-pkDB is DF-secure.*

Proof. It is trivial.

Theorem 10. *If symDB is one-time MiM-secure and the cryptosystem resists chosen-ciphertext attacks (IND-CCA secure) then Simp-pkDB is MiM-secure.*

Proof. Γ_i is a game and p_i denotes the probability that Γ_i succeeds.

Γ_0 : Adversary plays MiM game in the protocol in Figure 6 with the verifier with its instances, the prover with its instances and other actors. Let's assume that the number of prover instances is k where k is polynomially bounded.

Let s, pk, N and e be the values seen by the distinguished instance $\mathcal{V}$ of the verifier. Here $e = \mathsf{Enc}_{\mathsf{pk}}(s||N)$. We group the prover's instances as the following:

1. The provers seeing N and e,
2. The provers seeing e but another nonce N'.
3. The provers not seeing e (see a ciphertext e' which is not e).
   The probability that an adversary succeeds in Γ_0 is p_0.

Γ_1 : We reduce Γ_0 to Γ_1 where the first group has up to one prover instance P. We call $\mathcal{V}$ and P the matching instances. The probability that more than one prover picks same N is bounded by $\binom{k}{2}2^{-\ell}$ which is negligible. So, $p_1 - p_0$ is negligible.

Γ_2 : We reduce Γ_1 to Γ_2 where the matching P receives e after $\mathcal{V}$ has released e which means that e which is encryption of $s||N$ is only sent by the verifier. In Γ_1, the probability that $\mathcal{V}$ selects s after P has received e so that $\mathsf{Dec}_{\mathsf{sk}}(e) = s$ is $\frac{1}{2^{\ell}}$ which means that $p_2 - p_1$ is negligible.

Γ_3 : We reduce Γ_2 to Γ_3 where the provers are simulated as below:

The prover in the first group after receiving e run symDB(s) without decrypting e. Since e was released before, the value of s is already defined. The provers in the second group, abort the protocol after receiving e. The provers in the third group, call decryption oracle $\mathsf{Dec}_{\mathsf{sk}}(.)$ after receiving e' and check if the nonce is the same nonce that was chosen by them. Then they run symDB(s') with s' obtained from the decryption oracle.

The simulation gives identical result so the success probabilities in Γ_3 and Γ_2 are the same.

Γ_4 : We reduce Γ_3 to Γ_4. We simulate $\mathcal{V}$ in Γ_4. The simulation of $\mathcal{V}$ after selecting s encrypts a random plaintext instead of $s||N$.

Γ_3 and Γ_4 are indistinguishable because of the IND-CCA security of the encryption scheme. We construct an adversary $\mathcal{B}$ playing IND-CCA game and simulating MiM game against the adversary $\mathcal{A}$.

$\mathcal{B}$ receives pk from the IND-CCA game challenger and then $\mathcal{B}$ forwards it to $\mathcal{A}$. Firstly, $\mathcal{B}$ picks $N, s \in \{0,1\}^\ell$ and $r \in \{0,1\}^{2\ell}$ and assigns $m_0 = s||N, m_1 = r$. Then he sends m_0 and m_1 to IND-CCA game challenger and receives the response e_b where $e_b = \mathsf{Enc}_{\mathsf{pk}}(m_0)$ or $\mathsf{Enc}_{\mathsf{pk}}(m_1)$. If $\mathcal{A}$ interacts with $\mathcal{V}$ then $\mathcal{B}$ sends e_b, if $\mathcal{A}$ interacts with P, then $\mathcal{B}$ sends N. For the simulation of other prover instances P' (controlled by $\mathcal{A}$), when P' asks for the decryption of e', $\mathcal{B}$ sends e' to IND-CCA game challenger and receives decryption of e' to send P'. In the end, if $\mathcal{A}$ succeeds then $\mathcal{B}$ outputs 0, otherwise he outputs 1. If $\mathcal{A}$ succeeds given $b = 0$, then it means that he succeeds Γ_3 and if $\mathcal{A}$ succeeds given $b = 1$ then it means that he succeeds Γ_4. Therefore we have the following success probability of $\mathcal{B}$.

$$\mathsf{Adv}(\mathcal{B}) = \mathsf{Pr}[\mathcal{B} \to 1|b = 0] + \mathsf{Pr}[\mathcal{B} \to 1|b = 1] = p_3 - p_4$$

Since we know that the advantage of $\mathcal{B}$ is negligible, we can deduce that $p_3 - p_4$ is negligible (if we multiply negligible function with a polynomial we still have a negligible function).

Γ_5 : Now in Γ_5 we have at most two matching instances and they both run $\mathsf{symDB}(s)$ with the same and fresh random s. In Γ_5, The rest of the game (including the selection of pk and sk and the the decryption oracle $\mathsf{Dec}_{\mathsf{sk}}(.)$) is simulated by the adversary, Γ_4 and Γ_5 work the same. So $p_4 = p_5$. So they run symDB(s). The success probability p_5 of Γ_5 is negligible because of the security of OT-MiM-security of symDB.

As a conclusion, since $p_1 - p_0 = \mathsf{negl}$, $p_2 - p_1 = \mathsf{negl}$, $p_2 - p_3 = 0$, $p_4 - p_3 = \mathsf{negl}$, $p_5 - p_4 = 0$ and $p_5 = \mathsf{negl}$, we deduce that p_0 is negligible.

DH-Security: The protocol in Figure 6 is not secure against DH because of the attack in Figure B. In this attack, the malicious and far away prover P uses honest and close prover P' so that in the end V accepts P.

Basically, P chooses the same nonce that P' chose. Then V encrypts $s||N$ with the public key pk_P of P and then sends it to P. P decrypts e with his own secret key sk_P and then behaves as if he is the verifier and prepares encryption $e' = \mathsf{Enc}_{\mathsf{pk}_{\mathsf{P}'}}$ with using P''s public key $\mathsf{pk}_{\mathsf{P}'}$ and sends it to P'. Since e' is valid encryption for P', he continues by executing $\mathsf{symDB}(s)$ with V. In the end of the protocol, V accepts P since V has the P's public key. P' is used by P only to be able to pass the distance bounding phase of $\mathsf{symDB}(s)$ protocol.

C Security of Nonce-DH

Definition 15 (Gap Diffie-Hellman (GDH) [30]). *Let $\mathbb{G}$ be a prime order group and $g \in \mathbb{G}$ be a generator. We have the following problems:*

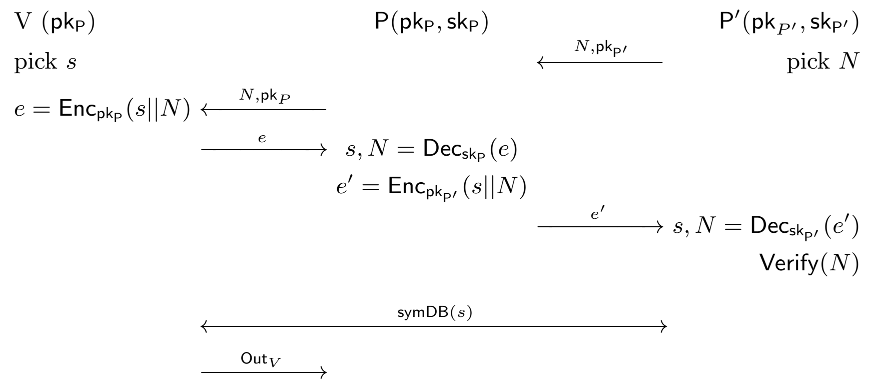

Fig. 7. DH attack on Simp-pkDB.

- ***Computational Diffie-Hellman Problem (CDH):*** *Given $g, X, Y \in \mathbb{G}$ compute $Z = g^{log_g X . log_g Y}$.*
- ***Decisional Diffie-Hellman Problem (DDH):*** *Given $g, X, Y, Z \in \mathbb{G}$, decide if $Z = g^{log_g X . log_g Y}$ or $Z = g^r$ where r is a random element.*

The GDH problem is solving the CDH given (g, X, Y) with the help of a DDH oracle which answers whether a given quadruple is a Diffie-Hellman quadruple.

Theorem 11. *Assuming that the GDH problem is hard and $\ell = \Omega(n)$, Nonce-DH is D-AKA secure in the random oracle model.*

Proof. The game Γ_0 is the D-AKA game. The challenger works as follows: He picks q and g as described in Nonce-DH. He randomly picks $\mathsf{sk}_A, \mathsf{sk}_B \in \mathbb{Z}_q$, and computes $\mathsf{pk}_A = g^{\mathsf{sk}_A}$, $\mathsf{pk}_B = g^{\mathsf{sk}_B}$. He picks randomly $s_1 \in \{0,1\}^n$ and then he assigns $(s_0, N) \leftarrow \mathcal{O}_B(\mathsf{pk}_A)$. Then, he picks $b \in \{0,1\}$ and gives $g, q, \mathsf{pk}_A, \mathsf{pk}_B, N, s_b$ to the adversary $\mathcal{A}$. $\mathcal{A}$ has access to the oracle H, $\mathcal{O}_A(.,.)$ (with the restriction not asking for pk_B, N) and $\mathcal{O}_B(.)$ defined below.

```
O_A(.,.)
Input: $\mathsf{pk}'_B, N'$
    if $(\mathsf{pk}'_B, N')$ equals $(\mathsf{pk}_B, N)$
        send $\perp$
    else:
        $s \leftarrow H(g, \mathsf{pk}'_B, \mathsf{pk}_A, \mathsf{pk}'^{\mathsf{sk}_A}_B, N')$
        send $s$
```

```
O_B(.)
Input: $\mathsf{pk}'_A$
    pick $N' \in \{0,1\}^\ell$
    $s \leftarrow H(g, \mathsf{pk}_B, \mathsf{pk}'_A, \mathsf{pk}'^{\mathsf{sk}_B}_A, N')$
    send $(s, N')$
```

```
H(.)
Input: $U$
    if $(U, .) \in T$
        send $V$ where $(U, V) \in T$
    else:
        pick $V \in \{0,1\}^n$
        save $(U, V)$ to $T$
        send V
```

```
H'(.)
Input: $(w, x, y, z, N')$
    if $w = g$ and $1 \leftarrow \mathsf{DDH}(g, x, y, z)$ :
        $z \leftarrow \perp$
    send $\mathrm{H}(w, x, y, z, N')$
```

We let $\perp$ be a special symbol which is unavailable to $\mathcal{A}$. The success probability of $\mathcal{A}$ in Γ_0 is p_0.

We reduce Γ_0 to Γ_1 where the oracle $\mathcal{O}_B$ never selects again the nonce N (which is obtained by the first call). Since a nonce in Γ_0 is equal to N with the probability $\frac{1}{2^\ell}$, $|p_1 - p_0| \leq \frac{q_B}{2^\ell}$ where q_B is the number of queries to $\mathcal{O}_B$. Due to $\ell = \Omega(n)$, $p_1 - p_0$ is negligible.

We reduce Γ_1 to Γ_2 where we replace H with H'. H' is defined with access to a DDH oracle (as Definition 15) as the following:

Since there is one-to-one mapping in the transformation of (g, x, y, z, N'), the success probability of Γ_2 remains the same which means $p_2 = p_1$.

We define another game Γ_3 where the only difference from Γ_2 is that we replace the oracle $\mathcal{O}_B$ with the oracle $\mathcal{O}'_B$.

```
$\underline{\mathcal{O}'_B(.)}$
**Input:** $\mathsf{pk}'_A$
    **pick** $N' \in \{0,1\}^\ell$
    $s \leftarrow H(g, \mathsf{pk}_B, \mathsf{pk}'_A, \perp, N')$
    **send** $(s, N')$
```

Note that $\mathcal{O}'_B$ queries H instead of H' and $N' \neq N$ due to the reduction to Γ_1. Γ_3 is exactly same with Γ_2 so the success probabilities p_3 and p_2 are the same as well.

Now in Γ_3, sk_B is used only by the DDH oracle.

We reduce Γ_3 to Γ_4 where $\mathcal{A}$ does not make the query $H'(g, \mathsf{pk}_B, \mathsf{pk}_A, z, N)$ with $z = \mathsf{pk}_A^{\mathsf{sk}_B}$. Indeed, any such query can be filtered using the DDH oracle and stopped to solve the GDH problem. Since the GDH problem is hard, $\mathcal{A}$ in Γ_3 selects $z = \mathsf{pk}_A^{\mathsf{sk}_B}$ given $(\mathsf{pk}_A, \mathsf{pk}_B)$ with negligible probability. Therefore, $p_4 - p_3$ is negligible.

In Γ_4, $H(g, \mathsf{pk}_B, \mathsf{pk}_A, \perp, N)$ is queried only once and this query is done by the challenger. Lastly, we reduce Γ_4 to Γ_5 where the challenger picks a random s_0 instead of picking $s_0 = H(g, \mathsf{pk}_B, \mathsf{pk}_A, \perp, N)$.

Γ_4 and Γ_5 are the same because if $(g, \mathsf{pk}_B, \mathsf{pk}_A, \perp, N)$ is never being queried again, it is not necessary that H stores $((g, \mathsf{pk}_B, \mathsf{pk}_A, \perp, N), s_0)$ in T. So, $p_4 = p_5$.

In Γ_5, s_0 and s_1 play a symmetric role and could be erased with b from the game after s_b is released. So, the state of the game after erasure of b, s_0 and s_1 are independent from b. Hence, $p_5 = \frac{1}{2}$ leading to $p_0 - \frac{1}{2}$ is negligible.

□

Theorem 12. *Assuming that $\ell = \Omega(n)$, Nonce-DH is D-AKA^p private in the random oracle model.*

Proof. The game Γ_0 is D-AKA^p game. The challenger works as follows: He picks q and g as described in Nonce-DH. He selects $\mathsf{sk}_A, \mathsf{sk}_{B_1} \in \mathbb{Z}_q$, and computes $\mathsf{pk}_A = g^{\mathsf{sk}_A}$ and $\mathsf{pk}_{B_1} = g^{\mathsf{sk}_{B_1}}$. Then, he sends $\mathsf{pk}_A, \mathsf{pk}_{B_1}$ and sk_{B_1} to $\mathcal{A}$. $\mathcal{A}$ selects sk_{B_0} and pk_{B_0} and sends them to the challenger. Next, the challenger picks $b \in \{0,1\}$, $N \in \{0,1\}^\ell$, queries $(g, \mathsf{pk}_{B_b}, \mathsf{pk}_A, \mathsf{pk}_A^{\mathsf{sk}_{B_b}}, N)$ to H and receives s. He sends s to $\mathcal{A}$. $\mathcal{A}$ has access to the oracle H as defined in the proof of Theorem 11, and to the oracle $\mathcal{O}_A(.,.)$.

We reduce Γ_0 to Γ_1 where $\mathcal{A}$ never selects the same nonce with N in the query of the oracle H or $\mathcal{O}_A$. The probability that he selects N is $\frac{1}{2^\ell}$ so $p_2 - p_1$ is negligible.

We reduce Γ_1 to Γ_2 where $\mathcal{O}_B$ picks s at random instead of a response from H. Since, the query $(g, \mathsf{pk}_{B_b}, \mathsf{pk}_A, \mathsf{pk}_A^{\mathsf{sk}_{B_b}}, N)$ by the challenger is never done again, we have $p_1 = p_2$. Now, b is never used in Γ_2. It means that s is independent from b, so $p_2 = \frac{1}{2}$. Therefore, $p_0 - \frac{1}{2}$ is negligible.

□