sustainability

MDPI

Article

Dynamic Membership Management in Anonymous and Deniable Distance Bounding

Nam-Su Jho [1] and Taek-Young Youn [2,*]

1 Electronics and Telecommunications Research Institute (ETRI), 218 Gajeong-ro, Yuseong-gu, Daejeon 34128, Korea; nsjho@etri.re.kr
2 Department of Industrial Security, Dankook University, 655 Mabuk-dong, Giheung-gu, Yongin-si 16891, Gyeonggi-do, Korea
* Correspondence: taekyoung@dankook.ac.kr; Tel.: +82-31-8005-3253

Received: 22 September 2020; Accepted: 8 December 2020; Published: 10 December 2020

Abstract: For secure location proof in many applications, distance bounding protocols are considered as one of the useful tools that can be used in practice. In distance bounding protocols, a prover and a verifier can measure the distance between them by performing an interactive protocol. In general, the verifier is regarded as an honest service provider, and thus, an adversarial verifier is not considered for security analysis. However, we cannot ignore the possibility of the corruption of the verifier, which can spoil the prover's privacy. To handle the security problem, a prover-anonymous and deniable distance bounding protocol is proposed, which can guarantee the privacy of the prover even though the verifier is corrupted. In this paper, we review the prover-anonymous and deniable distance bounding protocol in terms of the membership management, and we show that the communication overhead in the protocol for each membership change is $\mathcal{O}(n)$ where n is the number of users. Then, we propose an improved membership management technique, which can efficiently support membership change in terms of the communication overhead. The improved technique requires $\mathcal{O}(1)$ for each membership change instead of $\mathcal{O}(n)$, as in the existing protocol.

Keywords: location proof; distance bounding protocol; authentication; dynamic membership management; privacy

1. Introduction

In the beginning, distance bounding (DistB) protocols were devised to counter relay attacks in authentication protocols by allowing a verifier to check that a prover is really within his/her neighborhood or not [1]. By proving the closeness of two parties, a verifier can believe that it is not impossible to relay any information between them since the distance between the two parties is too short to intrude in their communication. Although the primary goal of DistB protocols was to counter relay attacks, currently, they are considered as a practical solution for secure localization in location-based services [2–4] since we can use them to measure the location-related information of a client. Authenticating the exact location-related information of a client makes it possible to provide IT services linked to a specific location or social community. These services are difficult to provide using a general user authentication protocol that is free of physical constraints. In addition, analyzing the possible security and privacy threats of protocols and providing enhancement techniques to cope with them are fundamental requirements ensuring the sustainability of the IT services based on these protocols and the social environment in which these services are used. Thus, until now, a number of distance bounding protocols have been proposed with additional functions and security features [5–7].

To prove the distance between a verifier and a prover in a reliable way, they perform a number of interactive protocols as follows. First, the prover and the verifier exchange some information to share common secret values in a secure and authenticated way. Using the shared secret values, the verifier and the prover perform a number of interactive challenge-response protocols to measure the round trip time between them. In each iteration, the verifier measures the current time and sends a challenge to the prover. Then, the prover responds to the challenge as soon as possible. If the verifier receives the response from the prover, he/she immediately measures the current time to evaluate the round trip time between the two parties. After performing the above-described challenge-response protocols, the verifier obtains a number of samples for the round trip time. The verifier computes the average round trip time from all measured round trip times. We can compute the distance between two parties using the average round trip time and the speed of the signals.

For the reliability of distance bounding protocols, we expect that no one except the proper can correctly respond to challenges sent by the verifier. To achieve the security goal, we use cryptographic techniques to generate the challenges. Generally, the prover is a low-power device, and thus, we cannot use any complicated questions as a challenge since the time to solve the question should not influence on the round trip time. Hence, most of the distance bounding protocols employ symmetric cryptography instead of costly public-key cryptography.

1.1. Related Works: Privacy Issues in Distance Bounding Protocols

Currently, the privacy of legitimate users is very important since the leakage of users' privacy causes unpredictable damage. Therefore, researchers have tried to protect legitimate users' privacy by using cryptographic techniques [5–16].

Since verifiers are generally service providers in distance bounding protocols, the privacy of the prover is an important security condition for DistB protocols [10–12]. For this reason, at first, researchers focused on the privacy of provers under the assumption that verifiers are relatively reliable. However, it is not easy to guarantee the security requirement without public-key cryptography [13].

Some protocols can protect the privacy of provers by using public-key cryptography [14–16] when the assumption that postulates that the verifiers are honesty entities holds. Until now, most of techniques, which have been proposed for the privacy issues, guarantee the privacy of provers only when verifiers are reliable.

Unfortunately, most of the existing techniques have shown that they cannot preserve the privacy of the provers' when verifiers are not reliable. Since we cannot ignore the possibility of the corruption of the verifier or the data leakage from the verifier, it is meaningful to design a secure and efficient DistB protocol that also can preserve the privacy of the prover against malicious or corrupted verifiers. The first solution was proposed by Gambs, Onete, and Robert [9]. In the literature, the prover-anonymous and deniable distance bounding (PA-DistB) protocol proposed in [9] was the first and only DistB protocol that can preserve the privacy of the prover even though the verifier is corrupted or any information is revealed by the verifier.

As a separate research flow, designing efficient DistB protocols by providing a new mechanism with a low false acceptance rate is one of the main research directions in this field [8,17–19]. Some researchers have designed distance bounding protocols that are secure against stronger adversarial provers who perform mafia-fraud attacks or terrorist-fraud attacks [5,6,20]. Guaranteeing stronger security or supporting improved performance is an important research issue in this field, but the main concern of this paper is the privacy of provers against malicious verifiers. Hence, from this viewpoint, we are interested in the technique introduced in [9] as an existing relevant technique since the technique is the only known solution for the same problem.

1.2. Contribution

In this paper, we review the PA-DistB protocol and analyze its performance, especially its membership revocation mechanism. Then, we propose a new membership management technique that requires remarkably less communication overhead than the existing technique in [9]. In the existing technique, for each membership change, the back-end server updates public information and posts it on the public bulletin board. Then, to perform the PA-DistB protocol, each user should download the updated public information (posted on the public bulletin board), the size of which is $\mathcal{O}(n)$ where n is the number clients. Hence, the communication overhead for downloading updated information is $\mathcal{O}(n)$ in the existing technique. For some applications where the number of clients is not small or the changes of membership occur frequently, the inefficiency of membership changes could be a burden on the system. Therefore, it is very important to provide an efficient membership management technique, the cost of which is not influenced by the number of clients. Our technique gives a way to solve the problem since it requires $\mathcal{O}(1)$ communication overhead instead of $\mathcal{O}(n)$, as in the existing PA-DistB protocol. The main contribution of our technique can be summarized as follows:

- Support anonymous and deniable distance bounding.
- Guarantee the privacy of legitimate provers against malicious verifiers.
- Main contribution: support efficient membership update for dynamic membership change.

We want to emphasize that reducing the complexity from $\mathcal{O}(n)$ to $\mathcal{O}(1)$ is very hard to accomplish. Therefore, the proposed technique is very meaningful work from this view point. As we know from the literature, our work is the first and only prover-anonymous distance bounding protocol supporting dynamic membership management with complexity $\mathcal{O}(1)$.

2. Review of the PA-DistB Protocol

In this section, we briefly review the description of the PA-DistB protocol [9]. Before describing the PA-DistB protocol, we want to emphasize that our goal is to give a new membership management technique that can improve the PA-DistB protocol rather than to design a new protocol. Since we did not modify the core algorithms of the PA-DistB protocol, some component algorithms will be described without a detailed explanation if it is possible to explain our idea without them.

We use the following notations throughout the paper (see Table 1).

Table 1. Notations.

Notation	Description
Srv	the server that maintains public information
Pv_i	the i-th prover
q	security parameter that determines the size of other variables [1]
$\mathcal{E}$	an elliptic curve defined over a finite field $\mathbb{F}_q$
P	a point of $\mathcal{E}$, the order of which is prime q
$\mathbb{G} = \langle P \rangle$	P is a generator of the group $\mathbb{G}$
x_i	secret key of the i-th prover Pv_i
P_i	auxiliary public key for the i-th prover Pv_i (used only for the protocol of [9])
$Q_{v,i}$	auxiliary public key for the i-th prover Pv_i in the v-th step (used only for our protocol)
$\mathsf{HEnc}(pk, M)$	encryption of message M under public key pk using additive homomorphic encryption
$\mathsf{Sign}(z, M)$	signature of message M under signing key z
$\mathsf{Prove}(c)$	zero-knowledge proof about ciphertext c
$\mathsf{X}_\ell(P')$	the most significant ℓ bits of the x-coordinate of P', where P' is a point of an elliptic curve $\mathcal{E}$

[1] For example, a 256 bit positive integer is chosen to guarantee the security similar to a 2048 bit RSA cryptosystem.

2.1. Description

2.1.1. Setting

In the PA-DistB protocol, three schemes are used as building blocks. Since the PA-DistB protocol can use any scheme as a building block if the scheme satisfies the pre-defined conditions described in [9], we describe the PA-DistB protocol without giving concrete schemes. Here, we simply remind the reader about some functions of the schemes to help understand the procedure of the PA-DistB protocol.

In the PA-DistB protocol, an additive homomorphic encryption scheme is used as a building block. Let $\mathsf{HEnc}(pk, M)$ be the encryption of a message M under the public key pk. Due to the additive homomorphic property, anyone can compute $\mathsf{HEnc}(pk, aM)$ from $\mathsf{HEnc}(pk, M)$ and an integer a. Note that, this is equal to $a \circ \mathsf{HEnc}(pk, M)$, which denotes the addition of the encryption a times by itself. Note that, in the PA-DistB protocol, the ElGamal encryption scheme defined over an elliptic curve (EC-ElGamal encryption) is considered for the scheme. Refer to Appendix A for the additive homomorphic property of the EC-ElGamal encryption.

For a server Srv that maintains a public bulletin board $\mathcal{B}$, we need a signature scheme to guarantee the information posted on the public bulletin board. Let $\mathsf{Sign}(z, M)$ be the signature of a message M under the server's signing key z.

For a prover, we need a non-interactive zero-knowledge system. Let $\mathsf{Prove}(c)$ be the zero-knowledge proof that proves that the ciphertext c is well formed. The meaning of "well formed" is that the ciphertext is the encryption of one of publicly known value. Refer to Section 2.1.4 for more explanation.

2.1.2. Prover and Verifier Setup

Let $\mathcal{E}$ be an elliptic curve defined over a finite field $\mathbb{F}_q$ and P be a point of $\mathcal{E}$, the order of which is prime q. Let $\mathbb{G} = \langle P \rangle$, which implies that P is a generator of the group $\mathbb{G}$. Let $\mathsf{X}_\ell(P')$ be the function that maps a point $P' \in \mathbb{G}$ to the most significant ℓ bits of the x-coordinate of P'.

For each prover Pv_i, a random value $x_i \in \mathbb{Z}_q$ is chosen as a secret key. Then, the server Srv evaluates an auxiliary public key P_i for Pv_i. Note that the secret key $x_i \in \mathbb{Z}_q$ and the auxiliary public key P_i are stored by both Pv_i and Srv. The auxiliary public key P_i is computed as:

$$P_i = \left(\prod_{j=1, j \neq i}^{k} x_j \right) P$$

for $i = 1, \ldots, k$. The server evaluates one more public key as:

$$Q = \left(\prod_{j=1}^{k} x_j \right) P.$$

Note that these values are updated dynamically as provers are created and revoked.

Let (z, Z) be a signature key pair for the server where z is the signing key and Z is the corresponding verification key. The server generates a signature $\sigma = \mathsf{Sign}(z, m)$, and then, he/she publishes the message m and the signature σ on the public bulletin board $\mathcal{B}$.

The verifier has a private/public key pair (y, Y) where $y \in \mathbb{Z}_q$ and $Y = yP$.

2.1.3. Prover Revocation

When a prover Pv_i is revoked, the server removes the revoked prover's secret key x_i from his/her database and updates all the values in the internal database and on the public board $\mathcal{B}$. For the update, the server removes the corresponding auxiliary public key P_i, regenerates the other prover's auxiliary public key $P_{j(\neq i)}$ and the public key Q, and generates a new signature for the updated information. Note that the updated auxiliary public keys, the public key, and the signature are generated as described in Section 2.1.2. In other words, we have to perform the *prover setup* procedure again whenever any user is revoked.

2.1.4. Protocol Execution

The PA-DistB protocol is composed of the following four phases.

Freshness Test

Before starting the protocol's execution, a prover Pv_i should test the freshness of the information stored in his/her storage by comparing it with the public information published on the public board $\mathcal{B}$. The prover can perform the protocol's execution without changing stored information if it is not updated. However, when the public information has been changed, the prover needs his/her updated auxiliary public key since all auxiliary public keys are updated for each prover revocation. The prover can obtain his/her auxiliary public key by downloading it from the public board. Unfortunately, to obtain his/her updated key, the prover should download all updated information from the public board instead of downloading only his/her updated auxiliary public key. See Section 2.2 for a detailed explanation.

Preparation Phase

The prover chooses three random values $r_1, r_2, sk_e \in \mathbb{Z}_q$ and generates R_1, R_2, pk_e as follows:

$$R_1 = r_1 p, R_2 = r_2 P, \text{ and } pk_e = sk_e P.$$

Then, the prover generates a ciphertext $c = \mathsf{HEnc}(pk_e, P_i)$ and a zero-knowledge proof $\pi = \mathsf{Prove}(c)$. Note that we use the zero-knowledge system to prove that the ciphertext c is the encryption of one of $\{P_j | j = 1, \ldots, k\}$. The verifier checks the given values and stops the protocol's execution if any value is invalid. Otherwise, the verifier chooses $r, r_3 \in \mathbb{Z}_q$, computes:

$$R_3 = r_3 P, c' = r \circ c = \mathsf{HEnc}(pk_e, rP_j),$$

and gives R_3, c' to the prover. Then, the prover tests if $R_3 \neq \mathcal{O}$ and accepts it if the condition holds. Let n be the number of iterations in the fast bit exchange phase. To prepare the fast bit exchange phase, the prover and the verifier generate:

$$p^0 || p^1 = \mathsf{X}_{2\ell}(r_1 R_3) \text{ and } \bar{p}^0 || \bar{p}^1 = \mathsf{X}_{2\ell}(r_3 R_1),$$

respectively.

Fast Bit Exchange Phase

For the fast bit exchange phase, the verifier randomly chooses ℓ challenge bits $c_1, \ldots, c_\ell$. Let $c = c_\ell || \cdots || c_1$. The prover and the verifier perform the following steps ℓ times. For the j-th iteration, two parties perform the following:

1. V starts the timer and sends the j-th challenge c_j to P;
2. P replies by sending $r_j = (1 - c_j) p_j^0 + c_j p_j^1$ to V;
3. V stops the timer and measures the round trip time ΔT_j.

Verification Phase

The verifier sends all challenge bits to the prover. Then, the prover compares this with the challenge received while performing the fast bit exchange phase. If the equality test is passed, the prover computes $D = r_2 Y$ and generates a signature S as follows:

$$S = x_j(rP_j) + eR_1 + R_2 + D.$$

The verifier then evaluates $\tilde{Q} = (S - yR_2) - eR_1 - R_2$ and tests if $\tilde{Q} = rQ$. The verifier checks the correctness of all responses returned by the prover. If all conditions hold, the verifier measures the average round trip time as:

$$\Delta T = \frac{1}{\ell} \cdot \sum_{j=1}^{\ell} \Delta T_j.$$

2.2. Membership Management in the PA-DistB Protocol

For seamless service, we expect that the PA-DistB protocol provides dynamic membership management in the sense that any modification (caused by some membership changes) can be immediately applied to the system. The necessity of the dynamic membership management was already considered and designed in [9]. The technique in [9] uses a public board to announce the quite recently updated public key Q, the auxiliary public keys $\{P_i | i = 1, \ldots, k\}$ for all users, and the signature σ for the information. For each membership change, all values published on the public board are updated by reflecting the membership change, and any user can dynamically follow the change by referring to the posted information.

The membership management technique in [9] works dynamically as claimed in [9], but we need an improved membership management technique since there are some weak points in the existing technique. Recall that each user needs his/her updated auxiliary public key to perform the PA-DistB protocol, and he/she can obtain the updated public key by downloading it from the public board. If the user reddownloads only his/her auxiliary public key, his/her anonymity cannot be guaranteed since anyone can identify the user by seeing the index of the downloaded key. Therefore, the user should download all updated information posted on the public board instead of downloading only his/her key. If there are many clients or the membership changes occur frequently, the cost of downloading updated data could be a burden on the execution of the PA-DistB protocol. Moreover, we cannot use the membership management technique in [9] for some applications where the device used for the protocol cannot access the public board. To use the PA-DistB protocol, devices (used by provers) should be able to access the public board, but some devices such as radio-frequency identification (RFID) tags cannot directly access the public board via the Internet. The verifier can help the devices by downloading the updated information on behalf of the provers. However, in this case, we need additional safeguards to prove the correctness and the freshness of the information given by the verifier.

Here, we do not want to disparage the value of the PA-DistB protocol since it is still useful in the current form. We merely want to emphasize that it is worth providing an efficient membership management technique that can support the PA-DistB protocol to overcome the above described weak points.

3. PA-DistB Protocol with Dynamic Membership Management

Before introducing the PA-DistB protocol with dynamic membership management, we want to emphasize that the main goal of this work is not to design an entirely new distance bounding protocol compared with the previous work in [9], but to give an improved membership management technique for the PA-DistB protocol without spoiling the security of the underlying protocol [9]. To achieve this goal, we will design a novel technique that does not require the public bulletin board since the use of the public board causes the above-described demerits.

3.1. Basic Idea

In this section, we briefly explain the basic idea of our protocol. Recall that the main goal of this work is to give an efficient way to deal with dynamic membership changes. In the existing protocol [9], the existence of a public board causes many problems in achieving the goal since a number of public keys are maintained in the board and the values should be downloaded for each membership change. In the following, we will explain the way to remove the public bulletin board from the protocol and the reason why it is not easy to remove the board in the existing technique. Finally, we will explain a brief strategy of this work in removing the bulletin board.

To remove the use of the public bulletin board, we have to support each prover to follow up all membership changes without access to the public board. To support each prover, we can use the verifier, who is sufficiently stronger than the provers. Since RFID readers, which have sufficient computing and electric power, are verifiers, it is reasonable to assume that the verifier can access the public bulletin board.

Based on the above-described observation, we can give a simple countermeasure as follows. First, the verifier gives the updated information to each prover before performing the protocol execution. In this case, the verifier cannot ask the prover to open the prover's user index to choose one auxiliary public key among many public keys since the user index can be used to break the anonymity of the prover. Then, in the existing technique [9], the only way is to give all updated keys to the prover. In the above-described simple countermeasure, the communication overhead for the update increases along with the number of members for the existing technique [9].

Recall that our goal is to present a way to fix the above discussed problem without sending all public keys. We achieve the goal by introducing the concept of versioned public keys. Instead of downloading all new public keys, in our approach, only the most recent public key of a prover will be computed by the server, and it will be transmitted to the prover. In our protocol, we assume that the verifier is stronger than the prover, and thus, the prover communicates with the server with the help of a verifier. To protect the privacy of the prover, the current public key will be transmitted to the server via the verifier in an encrypted form. To compute the most recent public key in an encrypted form without revealing the identity of the prover, the current public key will be encrypted using a homomorphic encryption scheme. Due to the property of homomorphic encryption schemes, a new public key can be computed from the current public key without revealing the owner of the public key.

3.2. Our Protocol

Now, we describe our technique. To clarify the difference between our construction and the existing construction in [9], we will give a short comment for each algorithm regarding the differences, if any. Recall that our protocol is modified from the work in [9]. Hence, unless otherwise stated, we refer to the protocol in [9] when discussing the previous technique.

3.2.1. Setting

In our protocol, the same setting is used as in the PA-DistB protocol, and thus, we omit the detailed explanation of the system setting procedure.

3.2.2. Prover Setup

In our scheme, some parameters are updated along with the change of membership. Therefore, we use a variable v to indicate the number of updates. Now, we call the state the v-th step if there are v changes of membership. In the initialization step, the state number is set to zero and gradually increased by one if a new member is joining or an existing member is revoked. When the latest version index is v, each prover holds a version index $v'(\leq v)$, and the server maintains the list $\mathsf{L}_{\mathsf{update}}$ of all the history of the membership changes.

Now, we describe the prover setup procedure. First, we setup each user's key pair as follows. Let k be the number of provers in the initialization step and $\mathcal{I}$ be the set of all indexes of valid provers. Then, $\mathcal{I} = \{1, \ldots, k\}$ in the setup phase. Each prover Pv_i chooses a secret key $x_i \in \mathbb{Z}_q$ and securely sends it to the server Srv. Then, the server evaluates the auxiliary public key $Q_{0,i}$ and gives it to the prover Pv_i. Here, the prover's auxiliary public key $Q_{0,i}$ is defined as:

$$Q_{0,i} = \left(\prod_{j=1, j\neq i}^{k} x_j \right) P.$$

Note that the key pair $\{x_i, Q_{0,i}\}$ is shared by Pv_i and Srv. Then, the initial public key Q_0 is defined as:

$$Q_0 = \left(\prod_{j=1}^{k} x_j\right) P.$$

The prover Pv_i's auxiliary public key $Q_{0,i}$ and the public key Q_0 will be updated for each membership change. Let $Q_{v,i}$ and Q_v be the updated auxiliary public key (for the prover Pv_i) and the updated public key for the v-th step, respectively. Let $\mathcal{Q}_v$ be the set of all provers' auxiliary public keys in the v-th step. Initially, $\mathcal{Q}_0 = \{Q_{0,i} | i = 0, \ldots, k\}$. The server generates a signature $\sigma = \mathsf{Sign}(z, m_0)$ where z is the signing key of the server and $m_0 = \{\mathcal{Q}_0, Q_0\}$. Then, the server distributes it to all provers. Using the signature, each prover can verify his/her own information given by the server.

Note that the main difference between the existing techniques is the version index. In [9], each prover did not maintain its version index since all updated public keys were posted on the public bulletin board. In our construction, each prover maintains his/her version number, and this is the main difference of this step. However, the computation of the initial keys is identical to the existing protocol.

3.2.3. Prover Joining or Revocation

Before describing the joining and revocation method, we want to emphasize that all equations described in this section are fully new ones that were not considered in the existing technique. Recall that the joining and revocation of provers are solved by publishing updated public keys using a public bulletin board.

Let v be the latest version index and n be the largest user index in the v-th step. The server maintains a list $\mathsf{L}_{\mathsf{update}}$ where all update details are summarized. For the v-th update, the server adds $\{v, i_v, \widehat{x}_v, \mathsf{flag}_v\}$ to the list where i_v is the user index of the joining/revoked user, $\widehat{x}_v$ is the private key of the user Pv_{i_v}, and flag_v is an indicator. We set $\mathsf{flag}_v = 1$ when the update is caused by the joining of a new member and $\mathsf{flag}_v = -1$ if the update is caused by the revocation of an existing member. In other words, the flag indicates the validity of the user since $\mathsf{flag}_v = 1$ if Pv_{i_v} is a valid user and $\mathsf{flag}_v = -1$ if Pv_{i_v} is an invalid user. Note that $\widehat{x}_v$ is the private key of the user P_{i_v} who is joining or revoked in the v-th step, and thus, we have $\widehat{x}_v = x_{i_v}$.

Recall that n is the largest user index, and thus, the user index for a new joining prover is $n+1$. When a new prover Pv_{n+1} joins the system, the server does the following:

- privately shares a secret key x_{n+1} with the new prover;
- stores $\mathcal{Q}_{v+1}$, Q_{v+1}, and σ_{v+1} instead of $\mathcal{Q}_v$, Q_v, and σ_v where:

$$\mathcal{Q}_{v+1} = x_{n+1}\mathcal{Q}_v \cup \{Q_v\}, Q_{v+1} = x_{n+1}Q_v, \text{ and}$$

$$\sigma_{v+1} = \mathsf{Sign}(z, \{\mathcal{Q}_{v+1}, Q_{v+1}\});$$

- adds $\{v+1, n+1, x_{n+1}, 1\}$ and $n+1$ to the list $\mathsf{L}_{\mathsf{update}}$ and $\mathcal{I}$, respectively;
- updates the version index and the largest user index as $v^* = v+1$ and $n^* = n+1$, respectively.

Suppose that an existing prover Pv_i maintains the information of which version number is v', i.e., his/her private key is x_i, and the corresponding public key is $Q_{v',i}$. To revoke the prover Pv_i, the server does the following:

- stores $\mathcal{Q}_{v+1}$, Q_{v+1}, and σ_{v+1} instead of $\mathcal{Q}_v$, Q_v, and σ_v where:

$$\mathcal{Q}_{v+1} = x_i^{-1}(\mathcal{Q}_v \setminus \{Q_{v,i}\}), Q_{v,i} = \prod_{j=v'+1}^{v} \widehat{x}_j^{\mathsf{flag}_j} Q_{v',i},$$

$$Q_{v+1} = x_i^{-1}Q_v \text{ and } \sigma_{v+1} = \mathsf{Sign}(z, \{\mathcal{Q}_{v+1}, Q_{v+1}\});$$

- adds $\{v+1, i, x_i, -1\}$ to $\mathsf{L}_{\mathsf{update}}$ and removes i from $\mathcal{I}$;
- updates the version index as $v^* = v+1$.

3.2.4. Protocol Execution

Note that most of the procedures of the improved protocol are identical to the PA-DistB protocol except the freshness test step. The only difference is the freshness test step and the key update procedure. In the two steps, the prover checks the version of its current public key and updates his/her public key with the help of the verifier if the version of the current public key is old. Except for the two steps, the protocol's execution is almost identical to the existing technique. The improved protocol works as follows.

Freshness Test

We assume that the verifier maintains the latest version index. When the prover and the verifier have the same index, they go to the preparation phase. Otherwise, the prover performs the key update procedure (as described in Section 3.2.5) to obtain the updated information and then goes to the preparation phase.

Preparation Phase

Here, we assume that the prover and the verifier have the information of the latest version. Let v be the version index and n be the largest user index.

As in the PA-DistB protocol, the prover chooses three random values $r_1, r_2, sk_e \in \mathbb{Z}_q$, computes $R_1 = r_1 p, R_2 = r_2 P, pk_e = sk_e P$, and generates $c = \mathsf{HEnc}(pk_e, Q_{v,i})$ and $\pi = \mathsf{NIZK}(c \text{ well formed})$. Then, the prover sends R_1, R_2, c, and π to the verifier. Then, the verifier chooses $r, r_3 \in \mathbb{Z}_q$, computes:

$$R_3 = r_3 P \text{ and } \widehat{c} = r \circ c = \mathsf{HEnc}(pk_e, rQ_{v,i}),$$

and gives them to the prover. Then, the prover tests if $R_3 \neq \mathcal{O}$ and accepts it if the condition holds. To prepare the fast bit exchange phase, the prover and the verifier generate:

$$p^0 || p^1 = \mathsf{X}_{2\ell}(r_1 R_3)$$

and:

$$\overline{p}^0 || \overline{p}^1 = \mathsf{X}_{2\ell}(r_3 R_1),$$

respectively.

Fast Bit Exchange Phase

For the fast bit exchange phase, the verifier randomly chooses ℓ challenge bits $c_1, \ldots, c_\ell$. Let p_j^b be the j-th bit of p^b. The prover and the verifier perform the following steps ℓ times. For the j-th iteration, the two parties perform the following:

S1. V starts the timer and sends the j-th challenge c_j to P;
S2. P replies by sending $r_j = (1-c_j)p_j^0 + c_j p_i^1$ to V;
S3. V stops the timer and measures the round trip time ΔT_j.

Verification Phase

The verifier sends all challenge bits to the prover. Then, the prover compares this with the challenge received while performing the fast bit exchange phase. If the equality test is passed, the prover computes $D = r_2 Y$ and generates a signature S as follows:

$$S = x_i(rQ_{v,i}) + eR_1 + R_2 + D.$$

The verifier then evaluates $\tilde{Q} = (S - yR_2) - eR_1 - R_2$ and tests if $\tilde{Q} = rQ$. The verifier checks the correctness of all responses returned by the prover. If all conditions hold, the verifier measures the average round trip time as:

$$\Delta T = \frac{1}{\ell} \cdot \sum_{j=1}^{\ell} \Delta T_j.$$

3.2.5. Key Update

Note that the key update method is a fully new technique compared with the existing protocol. In [9], the key update was performed by the server, and provers obtained their new public keys by downloading them from the public bulletin board. In our approach, each prover receives his/her updated public key from the server with the help of the verifier.

Let v be the index for the latest version of the membership state, v' be the version number maintained by the prover, and n be the largest user index in the v-th step. Recall that the key update procedure is performed only if $v' < v$. The prover P_i maintains the information of which version number is v', i.e., his/her private key is x_i, and the corresponding public key is $Q_{v',i}$.

Note that the prover cannot communicate with the server without the help of the verifier. The key update procedure can be performed with the help of the verifier, i.e., the verifier forwards all communicating messages from the prover to the server and vice versa. With the help of the verifier, the prover can obtain the updated key by performing the following procedure with the server:

- The prover Pv_i chooses a value $sk_u \in \mathbb{Z}_q$, $pk_u = sk_u P$, generates $c_u = \mathsf{HEnc}(pk_u, Q_{v',i})$, and gives c_u with the version number v' to the server in a secure and authenticated way (There are several cryptographic tools that can be used for establishing a secure and authenticated communication channel without revealing the privacy of the initiator. For example, we can use the signcryption scheme in [21] for the goal since the scheme permits a sender to share a common session key with the receiver without opening his/her identity to others except for the receiver.).
- To update the prover's key $Q_{v',i}$ in an encrypted form, the server Srv computes:

$$\begin{aligned} \widehat{c}_u &= \left(\prod_{j=v'+1}^{v} \widehat{x}_j^{\,\mathsf{flag}_j} \right) \circ \mathsf{HEnc}(pk_u, Q_{v',i}) \\ &= \mathsf{HEnc}\left(pk_u, \prod_{j=v'+1}^{v} \widehat{x}_j^{\,\mathsf{flag}_j} \cdot Q_{v',i} \right) \\ &= \mathsf{HEnc}(pk_u, Q_{v,i}), \end{aligned}$$

  and gives it with the newest version index v to the prover in a secure and authenticated way.
- The prover obtains the updated public key $Q_{v,i}$ by decrypting $\widehat{c}_u$ and stores it with the corresponding version index v in his/her secure storage.

4. Analysis of the Proposed Technique

In this section, we examine the correctness and security of the prover-anonymous and deniable distance bounding protocol described in Section 3.2. We also compare the proposed technique with existing anonymous and deniable distance bounding techniques.

4.1. Correctness

First, we examine the correctness of the protocol as follows.

Theorem 1. *The proposed prover-anonymous and deniable distance bounding protocol correctly works.*

Proof. Recall that the only difference between the proposed protocol and the underlying PA-DistB protocol is the key update technique. Though there are some insignificant differences, they have no influence on the correctness of the proposed protocol. Hence, it suffices to prove the correctness of the proposed key update technique. Let v be the index for the latest version of the membership state and $v'(< v)$ be the version number maintained by the prover P_i. To prove the correctness of the key update procedure, we have to show that:

$$\prod_{j=v'+1}^{v} \widehat{x}_j^{\,\mathsf{flag}_j} \cdot Q_{v',i} = Q_{v,i}.$$

Recall that $Q_{v',i}$ and $Q_{v,i}$ can be rewritten as follows:

$$Q_{v',i} = \prod_{j \in \mathcal{I}_{v'}, j \neq i} x_j \cdot P \text{ and } Q_{v,i} = \prod_{k \in \mathcal{I}_v, k \neq i} x_k \cdot P$$

where $\mathcal{I}_{v'}$ and $\mathcal{I}_v$ are the set of indexes of all valid provers in the v'-th step and the v-th step, respectively. When the set of indexes of all valid provers is changed from $\mathcal{I}_{v'}$ to $\mathcal{I}_v$ along with the change of membership, some members join and others are revoked. Let $\mathcal{I}_J$ and $\mathcal{I}_R$ be the set of indexes of the joining members and the revoked members from the v'-th step to the v-th step, respectively. Then, we have $\mathcal{I}_{v'} \cup \mathcal{I}_J \setminus \mathcal{I}_R = \mathcal{I}_v$. Note that existing users are not performing the joining procedure again, and only existing users can be revoked; thus, we have $\mathcal{I}_{v'} \cap \mathcal{I}_J = \emptyset$ and $\mathcal{I}_R \subset \mathcal{I}_{v'} \cup \mathcal{I}_J$. There are $v - v'$ updates between the v'-th step and the v-th step, and the recodes for the updates are as follows:

$$\begin{gathered}
\{v'+1, i_{v'+1}, \widehat{x}_{v'+1}, \mathsf{flag}_{v'+1}\} \\
\{v'+2, i_{v'+2}, \widehat{x}_{v'+2}, \mathsf{flag}_{v'+2}\} \\
\vdots \\
\{v, i_v, \widehat{x}_v, \mathsf{flag}_v\}.
\end{gathered}$$

Note that $\mathcal{I}_J \cup \mathcal{I}_R = \{i_{v'+1}, i_{v'+2}, \ldots, i_v\}$. Let:

$$\widetilde{\mathcal{I}}_J = \{k | i_k \in \mathcal{I}_J\} \text{ and } \widetilde{\mathcal{I}}_R = \{k | i_k \in \mathcal{I}_R\}.$$

It is easy to see that $\widetilde{\mathcal{I}}_J \cup \widetilde{\mathcal{I}}_R = \{v'+1, v'+2, \ldots, v\}$. Then, we can obtain the following:

$$\begin{aligned}
Q_{v,i} &= \prod_{j\in\mathcal{I}_v, j\neq i} x_j \cdot P \\
&= \prod_{j\in\mathcal{I}_{v'}\cup\mathcal{I}_J\setminus\mathcal{I}_R, j\neq i} x_j \cdot P \\
&= \prod_{j\in\mathcal{I}_{v'}, j\neq i} x_j \cdot \left(\prod_{j\in\mathcal{I}_J} x_j\right) \cdot \left(\prod_{j\in\mathcal{I}_R} x_j^{-1}\right) \cdot P \\
&= \left(\prod_{j\in\widehat{\mathcal{I}}_J} \widehat{x}_j^{\mathsf{flag}_j}\right) \cdot \left(\prod_{j\in\widehat{\mathcal{I}}_R} \widehat{x}_j^{\mathsf{flag}_j}\right) \cdot \prod_{j\in\mathcal{I}_{v'}, j\neq i} x_j \cdot P \\
&= \left(\prod_{j\in\widehat{\mathcal{I}}_J\cup\widehat{\mathcal{I}}_R} \widehat{x}_j^{\mathsf{flag}_j}\right) \cdot Q_{v',i} \\
&= \left(\prod_{j=v'+1}^{v} \widehat{x}_j^{\mathsf{flag}_j}\right) \cdot Q_{v',i}.
\end{aligned}$$

Hence, the proposed update technique correctly works. □

4.2. Security

It remains to examine the security of the proposed protocol. It is easy to prove the security of the proposed protocol due to the similarity between the proposed protocol and the PA-DistB protocol. Note that the proposed protocol is constructed by giving a new key update mechanism instead of using a public board, and this is the only difference between the two protocols. Therefore, the security of the proposed protocol is identical to the underlying protocol if the proposed key update technique can securely support the key update functionality without using the public board. Recall that the goal of the key update mechanism is to inform about the latest public information in an authenticated way. In the proposed protocol, the server maintains the history of updates and helps each prover update his/her key information. As we prove in Theorem 1, the technique given in Section 3.2.5 correctly updates the key information, and all communicating messages for the update are transmitted in a secure and authenticated way as discussed in Section 3.2.5. Therefore, the update procedure provides the same security functionalities as the PA-DistB protocol.

4.3. Comparison

In this section, we compare the proposed technique introduced in [9], which is known as the only anonymous and deniable distance bounding protocol in the literature. In Figure 1, we compare the main difference of the two protocols, the protocol in [9] and the proposed technique. For performance comparison, we focus on the cost of dynamic membership changes since the two techniques provide the same performance for the other operations. To compare their performance, we measure the number of operations and the size of the messages, and we summarize the result in Table 2.

Table 2. Efficiency comparison (for membership management).

	Computational Cost			**Communication Overhead**		
	Prover	**Verifier**	**Server**	**Prover**	**Verifier**	**Server**
[9]	c_v	-	$n^2 c_{sm} + c_s$	$(n-1)\ell_p + \ell_s$	-	$(n-1)\ell_p + \ell_s$
Ours	$c_{sm} + c_e + c_d + c_v$	-	$c_{sme} + c_{sm} + c_s$	$2\ell_c$	$4\ell_c$	$2\ell_c$

n: number of users, ℓ_p: length of a public key, ℓ_s: length of a signature, ℓ_c: length of a ciphertext. c_s: cost of signing, c_v: cost of signature verification, c_{sm}: cost of ECC scalar multiplication, c_{sme}: cost of the scalar multiplication of an encryption.

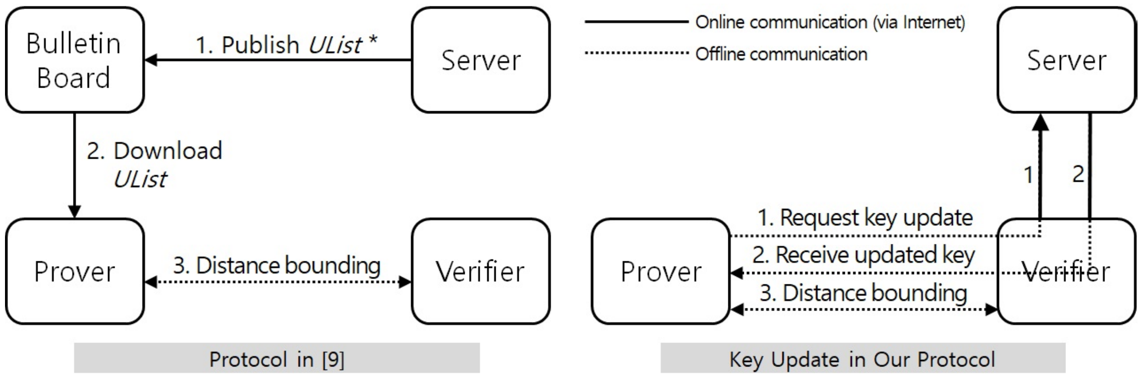

Figure 1. Protocol execution with updated public key. * *Ulist*: Updated list of all public keys.

Note that, in the existing technique [9], the server updates all public keys and publishes them on a public bulletin board. In our approach, the server records the update history as described in Section and helps the prover update his/her public key in the key update step as described in Section 3.2.5.

As summarized in Table 2, the existing technique requires about $\mathcal{O}(n^2)$ computations from the server's viewpoint and $\mathcal{O}(n)$ communication overhead. Though the prover needs more computations to get an updated public key from the server, the server's computational cost is dramatically reduced from $n^2c_{sm} + c_s$ to $c_{sme} + c_{sm} + c_s$. Though the server performs the operations for each prover's update requirements, our approach is more stable than the technique in [9] since the existing technique requires heavy operations instead of a light computational cost as in our construction. Moreover, in our scheme, the prover can update his/her public key with low communication overhead. Though the costs of sending and receiving a message are not the same, we count all messages in Table 2. As seen in Table 2, we need only a few messages to update the key values.

In Table 3, we summarize and compare main features of the two techniques, the technique in [9] and our technique. The two schemes support privacy protection for provers. The two schemes also can support dynamic membership management, but the scheme in [9] has a heavy computational cost and communication overhead for the feature. In our technique, we can support key update without assuming the existence of an additional server, but a bulletin board should be assumed in [9]. As a result, in our technique, a prover can update even though it cannot access the server via the Internet. Therefore, our scheme can support some devices with limited capability in terms of communication.

Table 3. Features.

Supported Feature	**[9]**	**Our**
Privacy protection for provers	O	O
Dynamic membership change	Δ	O
Key update without additional server	X	O
Support offline provers	X	O

5. Conclusions

In this paper, we review the efficiency of the prover-anonymous and deniable distance bounding protocol. Especially, we examine its performance in terms of the membership management and show that we need a somewhat improved membership management technique due to the communication overhead in the existing technique. Then, we propose an improved membership management technique that can improve the performance of the prover-anonymous and deniable distance bounding protocol by efficiently supporting membership change in terms of the communication overhead. The main features guaranteed by the proposed technique can be summarized as follows:

- Security (same a the technique in [9]).
  - Support anonymous and deniable distance bounding.
  - Guarantee the privacy of legitimate provers against malicious verifiers.
- Efficiency (main contribution).
  - Membership update without an additional bulletin board system to publish valid keys.
  - Update with $\mathcal{O}(1)$ messages instead of $\mathcal{O}(n)$ messages.
  - Low computational complexity for membership update (from $\mathcal{O}(n)$ to $\mathcal{O}(1)$).
  - Support dynamic update for offline provers.

As we summarized above, the proposed technique supports efficient membership management different from the existing technique, the complexity of which is $\mathcal{O}(n)$ where n is the number of users. We want to emphasize that our work is the first and only prover-anonymous distance bounding protocol supporting dynamic membership management with complexity $\mathcal{O}(1)$.

Author Contributions: Conceptualization, methodology, and formal analysis, N.-S.J.; validation, formal analysis, and writing, review and editing, T.-Y.Y. All authors read and agreed to the published version of the manuscript.

Funding: This work was supported by the Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government (20ZR1300, Development of core technologies for trust data connectome to realize a safe data society).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Additive Homomorphic Property of the EC-ElGamal Encryption Scheme

The original ElGamal encryption scheme is multiplicative homomorphic, but the ElGamal encryption defined over an elliptic curve (EC-ElGamal encryption) provides the additive homomorphic property with respect to the points of the underlying elliptic curve.

KeyGen: Let $\mathbb{G} = \langle P \rangle$ be a subgroup of prime order q defined by the points on the elliptic curve $\mathcal{E}$ over a finite field $\mathbb{F}_q$. The point P is a generator of the group $\mathbb{G}$. Then, a user chooses random $x \in \mathbb{Z}_q$ as his/her private key and computes $Q = xP$ as the corresponding public key.
Enc: To encrypt a message $M \in \mathcal{E}$, for randomly chosen r, we compute the following:

$$C = M + rQ \text{ and } D = rP.$$

Then, $Ctx = (C, D)$ is the ciphertext for M.
Dec: From the ciphertext Ctx, we can recover the message M as follows:

$$M = C - xD.$$

Add: For $i \in \{1, 2\}$, let $Ctx_i = (C_i, D_i)$ be the ciphertext of M_i where $c_i = M_i + r_iQ$ and $D_i = r_iP$. From Ctx_1 and Ctx_2, we can compute:

$$C' = C_1 + C_2 = M_1 + M_2 + (r_1 + r_2)Q$$

and:

$$D' = D_1 + D_2 = (r_1 + r_2)P.$$

Let $Ctx' = (C', D')$. Then, Ctx' is the encryption of $M_1 + M_2$ since the following equation holds:

$$C' - xD' = M_1 + M_2 + (r_1 + r_2)Q - x(r_1 + r_2)P = M_1 + M_2.$$

Due to the additive homomorphic property of EC-ElGamal encryption, we can compute the encryption of a message $aM \in \mathcal{E}$, from the encryption of the same message $M \in \mathcal{E}$ and any integer a.

Let $Ctx = (C, D)$ be the ciphertext for M where $C = M + rQ$ and $D = rP$ for some r. Then, it is easy to check the correctness of the property as follows:

$$a \circ Ctx = \underbrace{Ctx + \cdots + Ctx}_{a \text{ times}} = (aC, aD) = (aM + r'Q, r'P),$$

where $r' = ar$. Recall that, as defined in Section 2.1, $a \circ Ctx$ denotes the addition of the ciphertext a times itself.

References

1. Brands, S.; Chaum, D. Distance-bounding protocols. In *Advances in Cryptology—EUROCRYPT'93*; Springer: Berlin/Heidelberg, Germany, 1994; Volume 765, pp. 344–359.
2. Abu-Mahfouz, A.; Hancke, G.P. Distance Bounding: A Practical Security Solution for Real-Time Location Systems. *IEEE Trans. Ind. Inform.* **2013**, *9*, 16–27. [CrossRef]
3. Čapkun, S.; Hubaux, J.-P. Secure positioning of wireless devices with application to sensor networks. In Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005; Volume 3, pp. 1917–1928.
4. Nosouhi, M.R.; Sood, K.; Yu, S.; Grobler, M.; Zhang, J. PASPORT: A Secure and Private Location Proof Generation and Verification Framework. *IEEE Trans. Comput. Soc. Syst.* **2020**, *7*, 293–307. [CrossRef]
5. Avoine, G.; Bultel, X.; Gambs, S.; Gérault, D.; Lafourcade, P.; Onete, C.; Robert, J.-M. 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, Abu Dhabi, UAE, 2–6 April 2017; ACM Press: New York, NY, USA, 2017; pp. 800–814.
6. Bultel, X.; Gambs, S.; Gérault, D.; Lafourcade, P.; Onete, C.; Robert, J.-M. A Prover-Anonymous and Terrorist-Fraud Resistant Distance-Bounding Protocol. In Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, Darmstadt, Germany, 18–20 July 2016; ACM Press: New York, NY, USA, 2016; pp. 121–133.
7. Yang, A.; Pagnin, E.; Mitrokotsa, A.; Hancke, G.; Wong, D.S. Two-hop Distance-Bounding Protocols: Keep your Friends Close. *IEEE Trans. Mob. Comput.* **2017**, *17*, 1723–1736. [CrossRef]
8. Hanke, G.; Kuhn, M. An RFID distance boundiing protocol. In Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05), Athens, Greece, 5–9 September 2005; pp. 67–73.
9. Gambs, S.; Onete, C.; Robert, J.-M. Prover Anonymous and Deniable Distance-Bounding Authentication. In Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, Kyoto, Japan, 4–6 June 2014.
10. Juels, A.; Weis, S.A. Defining Strong Privacy for RFID. *ACM Trans. Inf. Syst. Secur.* **2009**, *13*, 7. [CrossRef]
11. Vaudenay, S. On privacy models for RFID. In *Advances in Cryptology—ASIACRYPT 2007, Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, 2–6 December 2007*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 68–87.
12. Paise, R.-I.; Vaudenay, S. Mutual authentication in RFID: Security and privacy. In Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, Tokyo, Japan, 18–20 March 2008; pp. 292–299.
13. Kardaş, S.; Kiraz, M.S.; Bingöl, M.A.; Demirci, H. A Novel RFID Distance Bounding Protocol Based on Physically Unclonable Functions. In *RFID: Security and Privacy, Proceedings of the RFIDSec: International Workshop on Radio Frequency Identification: Security and Privacy, Amherst, MA, USA, 26–28 June 2011*; Springer: Berlin/Heidelberg, Germany, 2011.
14. Reid, J.; Nieto, J.M.G.; Tang, T.; Senadji, B. Detecting relay attacks with timing-based protocols. In Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, Singapore, 20–22 March 2007; ACM Press: New York, NY, USA, 2007; pp. 204–213.
15. Bussard, L.; Bagga, W. Distance-bounding proof of knowledge to avoid real-time attacks. In *Security and Privacy in the Age of Ubiquitous Computing, Proceedings of the IFIP International Federation for Information Processing, Chiba, Japan, 30 May–1 June 2005*; Springer: Boston, MA, USA, 2005; pp. 222–238.

16. Hermans, J.; Peeters, R.; Onete, C. Efficient, secure, private distance bounding without key updates. In Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, Budapest, Hungary, 17–19 April 2013; ACM Press: New York, NY, USA, 2013; pp. 207–218.
17. Kim, C.H.; Avoine, G. RFID distance bounding protocols with mixed challenges. *IEEE Trans. Wirel. Commun.* **2011**, *10*, 1618–1626. [CrossRef]
18. Kim, C.H. Security Analysis of YKHL Distance Bounding Protocol with Adjustable False Acceptance Rate. *IEEE Commun. Lett.* **2011**, *15*, 1078–1080. [CrossRef]
19. Yum, D.H.; Kim, J.S.; Hong, S.J.; Lee, P.J. Distance bounding protocol with adjustable false acceptance rate. *IEEE Commun. Lett.* **2011**, *15*, 434–436.
20. Entezari, R.; Bahramgiri, H.; Tajamolian, M. RFID unilateral distance bounding protocols: A trade-off between mafia and distance fraud. *Comput. Commun.* **2017**, *98*, 97–105. [CrossRef]
21. Youn, T.-Y.; Hong, D. Signcryption with Fast Online Signing and Short Signcryptext for Secure and Private Mobile Communication. *Sci. China Inf. Sci.* **2012**, *55*, 2530–2541. [CrossRef]