7-2017

Privacy-preserving k-time authenticated secret handshakes

Yangguang TIAN

Shiwei ZHANG

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Yi MU

Yong YU

Privacy-Preserving k-time Authenticated Secret Handshakes

Yangguang Tian[1(✉)], Shiwei Zhang[1], Guomin Yang[1], Yi Mu[1], and Yong Yu[2]

[1] School of Computing and Information Technology,
Institute of Cybersecurity and Cryptology, University of Wollongong,
Wollongong, NSW 2522, Australia
{yt412,sz653,gyang,ymu}@uow.edu.au
[2] School of Computer Science, Shaanxi Normal University, Xi'an 710062, China
yuyong@snnu.edu.cn

Abstract. Secret handshake allows a group of authorized users to establish a shared secret key and at the same time authenticate each other anonymously. A straightforward approach to design an unlinkable secret handshake protocol is to use either long-term certificate or one-time certificate provided by a trusted authority. However, how to detect the misusing of certificates by an insider adversary is a challenging security issue when using those approaches for unlinkable secret handshake. In this paper, we propose a novel k-time authenticated secret handshake (k-ASH) protocol where each authorized user is only allowed to use the credential for k times. We formalize security models, including session key security and anonymity, for k-ASH, and prove the security of the proposed protocol under some computational problems which are proved hard in the generic bilinear group model. The proposed protocol also achieved public traceability property if a user misuses the k-time credential.

Keywords: Unlinkable secret handshake · Insider adversary · k-time authentication · Public traceability · Generic bilinear group model

1 Introduction

Secret handshake is a useful cryptographic primitive and has been extensively studied in the literature. It allows an authorized user to share a secret key with others without revealing their real identities. The following scenario can clarify its practicality. A FBI agent wants to contact with another agent, and both of them do not want to disclose their identity information during interaction. The only information they need to know is the peer belongs to the same agent system. There have been two types of unlinkable secret handshake system in the literature: one is based on the long-term certificate (e.g., [14,15]), and the other is based on one-time certificate (e.g., [16]). In the former type, the authorized user generates the shared secret value using the secret long-term certificate given by a trusted authority (TA) of the organization. In the latter type, the long-term

J. Pieprzyk and S. Suriadi (Eds.): ACISP 2017, Part II, LNCS 10343, pp. 281–300, 2017.
DOI: 10.1007/978-3-319-59870-3_16

secret value will be replaced by a set of one-time certificates and the authorized user will use one of them for unlinkable secret handshake in each session.

For the long-term certificate, an authorized user is allowed to reuse the given certificate when establishing a secret value with another authorized user. For example, the given secret certificate is blended with Diffie-Hellman key exchange, in order to generate a secret key with forward secrecy (e.g., [14,15]). Since the same certificate is used everytime, how to ensure the unlinkability is the major challenge in the protocol design. On the other hand, the one-time certificate approach (e.g., [16]) can address the unlinkability easily since each certificate is supposed to be used only once. Nevertheless, none of the previous approaches has considered the issue of misusing of certificates. We should note that for the one-time certificate schemes, the user is supposed to use each certificate once. However, reusing the given one-time certificates is a security issue that has not been formally considered in the previous works.

We give an example where misusing of the certificates (or credentials) should be prevented in secret handshake in some scenarios. Suppose there are n players subscribed to a real-time gaming system. Each user will obtain a set of k credentials from the game server after paying a subscription fee that is proportional to k. The players can form ad-hoc groups to play the game and a player can join a gaming session using one credential at a time. In order to ensure that only registered players are eligible to communicate with the peers, the players should generate a common session key to protect the communication. Also, it is desirable that the players cannot recognize each other except the fact that they are all legitimate subscribers of the system. Therefore, we may use a multi-party secret handshake protocol to achieve the security and privacy goals. However, in this example, a malicious player may try to reuse his credentials to continue playing the game without topping up extra money after all the credentials are used up. Therefore, it is important to identify such cheating players who reuse their one-time credentials. However, we found that the misusing of credentials has not been formally addressed in the previous secret handshake schemes. In this paper, we focus on addressing the credential misusing problem under the one-time certificate setting, and leave the task of designing such a scheme under the long-term certificate setting as our future work.

1.1 This Work

In this paper, we introduce the notion of k-time authenticated secret handshake (k-ASH), allowing all authorized users in a system to agree on a common secret value anonymously while preventing them from misusing their credentials issued by a trusted party of the system. Our contributions can be summarized as follows:

1. We present the formal security definition for k-time ASH protocol. In particular, we extend the eCK model [21] to define session key security and a variant of Juels-Weis privacy model [17] to define user anonymity.

2. We present a new unlinkable k-time ASH using anonymized Schnorr signature [22] and tag bases [25] to trace the cheating users who reuse their one-time credentials.
3. We prove a variant of the Computational Diffie-Hellman problem (VoCDH) and an extension of Decisional Combined Bilinear Diffie-Hellman problem (EVoDCBDH) [27] in the generic bilinear group model, and prove the security of the new k-time ASH protocol under these assumptions.

1.2 Related Work

Key Exchange. Bellare and Rogaway [6] introduced the first complexity-theoretic security model for key exchange under the symmetric-key setting. The model was later extended and enhanced under different contexts [2,5,7]. Canetti and Krawczyk [11] later refined the previous models and proposed a new model, known as the CK model, which is widely used in the analysis of many well-known key exchange protocols. Some variants [20,21] of CK model were also proposed to allow an adversary to obtain either long-term secret key or ephemeral secret key of the challenge session. Burmester and Desmedt [10] (BD) introduced several key exchange protocols in the multi-party setting, including star-based, broadcast-based, tree-based, and cyclic-based protocols. Later, a few generic transformations [8,18,19] were proposed to convert passive-secure group key exchange protocols into active-secure ones.

Secret Handshakes. Balfanz et al. [1] introduced the concept of secret handshake that allows any users in the same group to generate a shared value secretly using the long-term certificate approach. Afterwards, Castelluccia et al. [12] constructed a more efficient scheme than [1] under the standard Computational Diffie-Hellman Assumption. But both schemes did not provide the unlinkability property. In [26], Xu and Yung provided an unlinkable scheme but with weaker anonymity, named k-unlinkability, which means in the worst case, an adversary can infer that a participant is one out of certain k users. For achieving the full anonymity, Jarecki et al. [16] proposed two group secret handshake protocols using the BD group key agreement protocol (e.g., [10]). In particular, the second construction in [16] used one-time certificate to achieve full anonymity under the Gap Diffie-Hellman Assumption. Meanwhile, several secret handshake protocols have been proposed in the literature (e.g., [14,15]) which achieved full anonymity without using one-time certificate. The protocol in [15] and the improvement protocol in [14] are long-time certificate based, and both of them are allowed to reuse the given certificate with unlimited number of times.

2 Security Model

In this section, we present the security models for k-ASH. As mentioned in the introduction, a secure k-ASH protocol should achieve both session key security and anonymity. Below we present the corresponding security models to capture the above requirements. Specifically, the session key security model is a *modified*

version of eCK model [21], which is an extension of CK model [11] in the secret handshake setting, while the anonymity model is extended from the privacy models ([17,24]) for RFID authentication protocols.

States. We define a system user set $\mathcal{U}$ with n users, i.e. $|\mathcal{U}| = n$. We say an oracle Π_U^i may be *used* or *unused*. The oracle is considered as unused if it has never been initialized. Each unused oracle Π_U^i can be initialized with a secret key x. The oracle is initialized as soon as it becomes part of a group. After the initialization the oracle is marked as used and turns into the *stand-by* state where it waits for an invocation to execute a protocol operation. Upon receiving such invocation the oracle Π_U^i learns its partner identifier pid_U^i and turns into a *processing* state where it sends, receives and processes messages according to the description of the protocol. During that stage, the internal state information $state_U^i$ is maintained by the oracle. The oracle Π_U^i remains in the processing state until it collects enough information to compute the session key K_U^i. As soon as K_U^i is computed Π_U^i *accepts* and *terminates* the protocol execution meaning that it would not send or receive further messages. If the protocol execution fails then Π_U^i terminates without having accepted.

Partnering. We denote the i-th session established by a user U by Π_U^i, and identities of all the users recognized by Π_U^i during the execution of that session by pid_U^i. We define sid_U^i as the unique session identifier belonging to the session i established by the user U. Specifically, $\mathsf{sid}_U^i = \{m_j\}_{j=1}^n$, where $m_j \in \{0,1\}^*$ is the message transcript among users. We say two instance oracles Π_U^i and $\Pi_{U'}^j$ are *partners* if and only if $\mathsf{pid}_U^i = \mathsf{pid}_{U'}^j$ and $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j$.

2.1 System Model

We define a k-time authenticated secret handshake protocol consists of the following algorithms:

- Setup: The algorithm takes the security parameter λ as input, outputs the master public parameters mpk (including the k-time tag bases) and the master secret keys msk.
- KeyGen: The algorithm takes the master public key mpk as input, outputs a public/secret key pair (X, x).
- Register: This is an interactive algorithm that executed between the user and the TA. TA takes the master secret key msk and a public key X of one user as input, outputs a set of credentials $\{s_i\}_{i=1}^k$ on X. The user will become a registered user after interaction with TA.
- Handshake: This is an interactive algorithm that executed by registered users. Each user takes his/her secret key x, one of his/her credentials $\{s_i\}_{i=1}^k$ and mpk as input, outputs a shared secret key K if and only if his/her counterparts are registered users.
- Tracing: The algorithm takes two handshake transcripts of one user and one of tag bases as input, outputs the user's public key X.

2.2 Session Key Security

We define the session key security model for k-ASH protocols, in which each user obtains a set of credentials associated with his/her public key from the TA, and establishes a session key using one of the given secret credentials in one session. The model is defined via a game between a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a simulator $\mathcal{S}$. $\mathcal{A}$ is an active attacker with full control of the communication channel among all the users.

- Setup: $\mathcal{S}$ first generates master public/secret key pair (mpk, msk) for the TA and long-term secret keys $\{x_i\}_{i=1}^{n}$ for n users by running the corresponding KeyGen algorithms, where x_i denotes the secret key of user i. In addition, $\mathcal{S}$ generates a set of secret credentials $\{s_{i,j}\}_{j=1}^{k}$ for user i by running the Register algorithm. $\mathcal{S}$ also tosses a random coin b which will be used later in the game. Let $\mathcal{U}$ denote all the registered users.
- Training: $\mathcal{A}$ can make the following queries in arbitrary sequence to simulator $\mathcal{S}$.
  - Establish: $\mathcal{A}$ is allowed to register a user U' with public key X_i'. If a user is registered by $\mathcal{A}$, then we call this user *dishonest*; Otherwise, it is *honest*.
  - Send: If $\mathcal{A}$ issues send query in the form of (U, i, m) to simulate a network message for the i-th session of user U, then $\mathcal{S}$ would simulate the reaction of instance oracle Π_U^i upon receiving message m, and returns to $\mathcal{A}$ the response that Π_U^i would generate; If $\mathcal{A}$ issues send query in the form of $(U,' start')$, then $\mathcal{S}$ creates a new instance oracle Π_U^i and returns to $\mathcal{A}$ the first protocol message.
  - Session key reveal: $\mathcal{A}$ can issue reveal query to an accepted instance oracle Π_U^i. If the session is accepted, then $\mathcal{S}$ will return the session key to $\mathcal{A}$; Otherwise, a special symbol '$\perp$' is returned to $\mathcal{A}$.
  - Ephemeral secret key reveal: If $\mathcal{A}$ issues an ephemeral secret key reveal query to (possibly unaccepted) instance oracle Π_U^i, then $\mathcal{S}$ will return all ephemeral secret values contained in Π_U^i at the moment the query is asked.
  - long term secret key reveal: If $\mathcal{A}$ issues a long term secret key reveal (or corrupt, for short) query to user i, then $\mathcal{S}$ will return both the long term secret key and the secret credential set $(x_i, \{s_{i,j}\}_{j=1}^{k})$ to $\mathcal{A}$.
  - Master secret key reveal: If $\mathcal{A}$ issues a master secret key reveal query to TA, then $\mathcal{S}$ will return the master secret keys msk to $\mathcal{A}$.
  - Test: This query can only be made to an accepted and *fresh* (as defined below) session i of a user U. Then $\mathcal{S}$ does the following:
    * If the coin $b = 1$, $\mathcal{S}$ returns the real session key to the adversary;
    * Otherwise, a random session key is drawn from the session key space and returned to the adversary.

    Note that $\mathcal{A}$ can generate a set of secret credentials $\{s_{i,j}\}_{j=1}^{k}$ of user i after issuing Master secret key reveal query to TA. It is also worth noting that $\mathcal{A}$ can continue to issue other queries after the Test query. However, the test session must maintain fresh throughout the entire game.

Finally, $\mathcal{A}$ outputs b' as its guess for b. If $b' = b$, then the simulator outputs 1; Otherwise, the simulator outputs 0.

Freshness. We say an *accepted* instance oracle Π_U^i is fresh if $\mathcal{A}$ does not perform any of the following actions during the game:

- $\mathcal{A}$ issues Session key reveal query to Π_U^i or its accepted partnered instance oracle $\Pi_{U'}^j$;
- $\mathcal{A}$ issues both Long term secret key reveal query to U' s.t. $U' \in \mathsf{pid}_U^i$ and Ephemeral secret key reveal query for an instance $\Pi_{U'}^j$ partnered with Π_U^i;
- $\mathcal{A}$ issues Long term secret key reveal query to user U' s.t. $U' \in \mathsf{pid}_U^i$ prior to the acceptance of instance Π_U^i and there exists no instance oracle $\Pi_{U'}^j$ partnered with Π_U^i.
  Note that the Master key reveal query to TA is equivalent to the Long term secret key reveal to all users in pid_U^i.

We define the advantage of an adversary $\mathcal{A}$ in the above game as

$$\mathtt{Adv}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{S} \to 1] - 1/2. \tag{1}$$

Definition 1. *We say a k-ASH protocol has* session key security *if for any PPT* $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ *is a* negligible *function of the security parameter* λ.

2.3 Anonymity

Informally, an adversary is not allowed to identify who are the handshake users, with the condition that honest users authenticate with each other within k times. We define a game between an *insider* adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ as follows:

- Setup: $\mathcal{S}$ generates master public/secret key pairs (mpk, msk) for the TA and long term secret keys $\{x_i\}_{i=1}^n$ for n users by running the corresponding KeyGen algorithms. In addition, $\mathcal{S}$ generates a set of secret credentials $\{s_{i,j}\}_{j=1}^k$ for user i by running the Register algorithm. $\mathcal{S}$ also tosses a random coin b which will be used later in the game. We denote the original n users set as $\mathcal{U}$.
- Training: $\mathcal{A}$ is allowed to issue Establish, Send, Ephemeral secret key reveal, Session key reveal and at most n-2 Long term secret key reveal queries to $\mathcal{S}$. We denote the honest (i.e., uncorrupted) user set as $\mathcal{U}'$.
- Challenge: $\mathcal{A}$ randomly selects two users $U_i, U_j \in \mathcal{U}'$ as challenge candidates, then $\mathcal{S}$ remove them from $\mathcal{U}'$ and simulates U_b^* to $\mathcal{A}$ by either $U_b^* = U_i$ if $b = 1$ or $U_b^* = U_j$ if $b = 0$.
  Let $\mathcal{A}$ interact with U_b^*. Note that $\mathcal{A}$ is allowed to activate at most k sessions for U_i, U_j throughout the entire game.

$$\mathcal{A} \Leftrightarrow U_b^* = \begin{cases} U_i & b = 1 \\ U_j & b = 0 \end{cases}$$

Finally, $\mathcal{A}$ outputs b' as its guess for b. If $b' = b$, then the simulator outputs 1; Otherwise, the simulator outputs 0.

We define the advantage of $\mathcal{A}$ in the above game as

$$\mathtt{Adv}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{S} \rightarrow 1] - 1/2. \tag{2}$$

Definition 2. *We say a k-ASH protocol has* anonymity *if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ is a* negligible *function of the security parameter λ.*

3 Our Construction

3.1 Preliminaries

Bilinear Map. The bilinear map $\hat{\mathsf{e}} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ has the following properties:

1. Bilinearity: $\hat{\mathsf{e}}(g^{\alpha_i}, g^{\alpha_j}) = \hat{\mathsf{e}}(g, g)^{\alpha_i \cdot \alpha_j} : \forall \alpha_i, \alpha_j \in \mathbb{Z}_q, g \in \mathbb{G}$.
2. Non-degeneracy: $\hat{\mathsf{e}}(g, g) \neq 1$.
3. Computable: There exists an efficient algorithm for computing the bilinear map.

Note that the map $\hat{\mathsf{e}}$ is symmetric since $\hat{\mathsf{e}}(g^{\alpha_i}, g^{\alpha_j}) = \hat{\mathsf{e}}(g, g)^{\alpha_i \cdot \alpha_j} = \hat{\mathsf{e}}(g^{\alpha_j}, g^{\alpha_i})$.

3.2 Modified Computational Diffie-Hellman Problem

Definition 3 ***Computational Diffie-Hellman (CDH) Assumption*** *[20]: Given $g, g^a, g^b \in \mathbb{G}$ where $a, b \in_R \mathbb{Z}_q$, we define the advantage of the adversary in solving the CDH problem as*

$$\mathtt{Adv}_{\mathcal{A}}^{CDH}(\lambda) = \Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} \in \mathbb{G}]$$

We say a CDH assumption holds in group $\mathbb{G}$ if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ is a negligible *function of the security parameter λ.*

We propose a variant of computational diffie-hellman problem (VoCDH) below.

Definition 4. *Given $g, g^a, g^{1/a}, g^b \in \mathbb{G}$ where $a, b \in_R \mathbb{Z}_q$, we define the advantage of the adversary in solving the VoCDH problem as*

$$\mathtt{Adv}_{\mathcal{A}}^{VoCDH}(\lambda) = \Pr[\mathcal{A}(g, g^a, g^{1/a}, g^b) = g^{ab} \in \mathbb{G}]$$

We prove the above VoCDH problem is hard in $\mathbb{G}$ with a bilinear map $\hat{\mathsf{e}} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ in the generic bilinear group model [9,23].

Theorem 1. *Let $\epsilon_1, \epsilon_2 : \mathbb{F}_p \rightarrow \{0,1\}^*$ be two random encodings (injective functions) where $\mathbb{F}_p$ is a prime field and $\mathbb{G} = \{\epsilon_1(a) | a \in \mathbb{F}_p\}, \mathbb{G}_1 = \{\epsilon_2(a) | a \in \mathbb{F}_p\}$. If a, b are uniformly and independently chosen from $\mathbb{F}_p$ and encodings ϵ_1, ϵ_2 are randomly chosen, we then define the advantage of the adversary in solving the VoCDH with at most q, q_1 queries to the group operation oracles $\mathcal{O}, \mathcal{O}_1$ and $q_{\hat{\mathsf{e}}}$ queries to the bilinear pairing oracle $\mathcal{O}_{\hat{\mathsf{e}}} : \epsilon_1 \times \epsilon_1 \rightarrow \epsilon_2$ as*

$$\begin{aligned} \mathtt{Adv}_{\mathcal{A}}^{VoCDH}(\lambda) &= \Pr[\mathcal{A}(\epsilon_1(1), \epsilon_1(a), \epsilon_1(b), \epsilon_1(a^{-1})) \\ &= \epsilon_1(a \cdot b)] \leq \frac{4(q + q_1 + q_{\hat{e}} + 4)^2}{p} \end{aligned}$$

Proof. Let $\mathcal{S}$ be the simulator to simulate the entire game for $\mathcal{A}$. $\mathcal{S}$ maintains two polynomial sized dynamic lists: $L_1 = \{(p_i, \epsilon_{1,i})\}, L_2 = \{(q_i, \epsilon_{2,i})\}$, the $p_i \in \mathbb{F}_p[X_1, X_2]$ are 2-variate polynomials over $\mathbb{F}_p$, such that $p_0 = 1, p_1 = X_1, p_2 = X_2, p_3 = X_1^{p-2}$, and $\{\epsilon_{1,i}\}_{i=0}^{3} \in_R \{0,1\}^*$ are corresponding arbitrary strings, $\mathcal{S}$ then sets those pairs $(p_i, \epsilon_{1,i})$ as L_1. Therefore, the two lists are initialised as $L_1 = \{(p_i, \epsilon_{1,i})\}_{i=0}^{3}, L_2 = \emptyset$.

At the beginning of the game, $\mathcal{S}$ sends $\{\epsilon_{1,i}\}_{i=0,\cdots,3}$ to $\mathcal{A}$. After this, $\mathcal{S}$ simulates the group operation oracle $\mathcal{O}, \mathcal{O}_1$ and the bilinear pairing oracle $\mathcal{O}_{\hat{\mathbf{e}}}$ as follows. We assume that all requested operands are obtained from $\mathcal{S}$.

- $\mathcal{O}$: The group operation involves two operands $\epsilon_{1,i}, \epsilon_{1,j}$. Based on these operands, $\mathcal{S}$ searches the list L_1 for the corresponding polynomials p_i and p_j. Then $\mathcal{S}$ perform the polynomial addition or subtraction $p_l = p_i \pm p_j$ depending on whether multiplication or division is requested. If p_l is in the list L_1, then $\mathcal{S}$ returns the corresponding ϵ_l to $\mathcal{A}$. Otherwise, $\mathcal{S}$ uniformly chooses $\epsilon_{1,l} \in_R \{0,1\}^*$, where $\epsilon_{1,l}$ is unique in the encoding string L_1, and appends the pair $(p_l, \epsilon_{1,l})$ into the list L_1. Finally, $\mathcal{S}$ returns $\epsilon_{1,l}$ to $\mathcal{A}$ as the answer. Group operation queries in $\mathbb{G}_1$ (i.e., $\mathcal{O}_1$) is treated similarly.
- $\mathcal{O}_{\hat{\mathbf{e}}}$: The group operation involves two operands $\epsilon_{1,i}, \epsilon_{1,j}$. Based on these operands, $\mathcal{S}$ searches the list L_1 for the corresponding polynomials p_i and p_j. Then $\mathcal{S}$ perform the polynomial multiplication $p_l = p_i \cdot p_j$. If p_l is in the list L_2, then $\mathcal{S}$ returns the corresponding $\epsilon_{2,l}$ to $\mathcal{A}$. Otherwise, $\mathcal{S}$ uniformly chooses $\epsilon_{2,l} \in_R \{0,1\}^*$, where $\epsilon_{2,l}$ is unique in the encoding string L_2, and appends the pair $(p_l, \epsilon_{2,l})$ into the list L_2. Finally, $\mathcal{S}$ returns $\epsilon_{2,l}$ to $\mathcal{A}$ as the answer.

After querying at most q, q_1, q_e times of corresponding oracles, $\mathcal{A}$ terminates and outputs $\epsilon_1(x_1 \cdot x_2)$. At this point, $\mathcal{S}$ chooses random $a, b \in_R \mathbb{F}_p$ and sets $X_1 = a, X_2 = b$. The simulation by $\mathcal{S}$ is perfect unless the `abort` event happens. Thus, we bound the probability of event `abort` by analyzing the following cases:

1. $p_i(a,b) = p_j(a,b)$: Since $p_i \neq p_j$ as the method of L_1 is generated, $p_i - p_j$ is a non-zero polynomial of degree 0, 1, or $p-2$ where $p-2$ is produced by X_1^{p-2}. Since $X_1 \cdot X_1^{p-2} = X_1^{p-1} \equiv 1 \pmod{p}$, we have $X_1(p_i - p_j)$ is a non-zero polynomial of degree 0, 1, or 2. Therefore, the maximum degree of $X_1(p_i - p_j)$ is 2. By using lemma 1 in [23], we have $\Pr[(X_1(p_i - p_j))(a,b) = 0] \leq \frac{2}{p}$ and thus $\Pr[p_i(a,b) = p_j(a,b)] \leq \frac{2}{p}$. As there are $\binom{q+4}{2}$ pairs of (p_i, p_j), we have the `abort` probability is $\Pr[\mathsf{abort}_1] \leq \binom{q+4}{2} \cdot \frac{2}{p}$.
2. $q_i(a,b) = q_j(a,b)$: Since $q_i \neq q_j$ as the method of L_2 is generated and q_i, q_j are in the form of $\sum a_{k,l} p_k p_j$ for some constants $a_{k,l}$, $q_i - q_j$ is a non-zero polynomial of degree 0, 1, 2, $p-1$, $p-2$, or $2p-4$. Similar to above case, we have $X_1^2 \cdot X_1^{p-1} \equiv X_1^2$, $X_1^2 \cdot X_1^{p-2} \equiv X_1$, and $X_1^2 \cdot X_1^{2p-4} = (X_1^{p-1})^2 \equiv 1 \pmod{p}$. Therefore, $X_1^2(q_i - q_j)$ is a non-zero polynomial of degree ranging from 0 to 4. Since the maximum degree of $X_1^2(q_i - q_j)$ is 4, we have $\Pr[(X_1^2(q_i - q_j))(a,b) = 0] \leq \frac{4}{p}$ and thus $\Pr[q_i(a,b) = q_j(a,b)] \leq \frac{4}{p}$. As there are $\binom{q_1+q_{\hat{e}}}{2}$ pairs of (q_i, q_j), we have the `abort` probability is $\Pr[\mathsf{abort}_2] \leq \binom{q_1+q_{\hat{e}}}{2} \cdot \frac{4}{p}$.

3. $p_i(a,b) = ab$: Since the degree of p_1 is 0, 1, or $p-2$, and the degree of X_1X_2 is 2, we have that $p_i - X_1X_2$ is a non-zero polynomial of degree 2 or $p-2$. Similar to the case 1, we have $X_1(p_i - X_1X_2)$ is a non-zero polynomial of maximum degree of 3. Therefore, we have $\Pr[(X_1(p_i - X_1X_2))(a,b) = 0] \leq \frac{3}{p}$ and thus $\Pr[p_i(a,b) = ab] \leq \frac{3}{p}$. As there are $q+4$ polynomials in L_1, we have the abort probability is $\Pr[\mathsf{abort}_3] \leq \frac{3(q+4)}{p}$.

By combining all above cases, we have the abort probability is

$$\begin{aligned}\Pr[\mathsf{abort}] &= \Pr[\mathsf{abort}_1] + \Pr[\mathsf{abort}_2] + \Pr[\mathsf{abort}_3] \\ &\leq \binom{q+4}{2} \cdot \frac{2}{p} + \binom{q_1 + q_{\widehat{e}}}{2} \cdot \frac{4}{p} + \frac{3(q+4)}{p} \\ &< \frac{(q+4)^2 + 2(q_1 + q_{\widehat{e}})^2 + 3(q+4)}{p} \\ &< \frac{4(q + q_1 + q_{\widehat{e}} + 4)^2}{p}\end{aligned}$$

3.3 Modified Decisional Combined Bilinear Diffie-Hellman Problem

Definition 5. ***Variant of Decisional Combined Bilinear Diffie-Hellman Problem:*** *Given $g, g^a, g^b, h^c, h^d, h^{1/d} \in \mathbb{G}$ where $a, b, c, d \in_R \mathbb{Z}_q$ and $h = g^e$, we define the advantage of the adversary in solving the VoDCBDH problem as*

$$\begin{aligned}\mathtt{Adv}_{\mathcal{A}}^{VoDCBDH}(\lambda) = \Pr[w = \mathcal{A}(g, g^a, g^b, g^{ec}, g^{ed}, g^{e/d}, \\ T_0, T_1, w \in_R \{0,1\}) : T_w = g^{ab+ecd}, T_{w-1} = Z].\end{aligned}$$

The VoDCBDH problem is a variant of Decisional Combined Bilinear Diffie-Hellman Problem [27]. We prove the VoDCBDH problem is hard in $\mathbb{G}$ with a bilinear map $\hat{\mathsf{e}} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ in the generic bilinear group model [9,23].

Theorem 2. *The lower bound of the complexity of the VoDCBDH problem is stated as follows, querying the group operations and bilinear pairing operations at most q times.*

$$\mathtt{Adv}_{\mathcal{A}}^{VoDCBDH}(\lambda) \leq \frac{3(q+9)^2}{p}.$$

To prove this theorem, we introduce an intermediate problem (see Lemma 2), and we prove that the hardness of intermediate problem implies the hardness of the VoDCBDH problem. After that, we prove the intermediate problem is intractable (see Lemma 1) and then the theorem follows.

Definition 6. *Given $g, g^d, g^{cd}, g^{d^2}, g^e, g^{ae}, g^{be} \in \mathbb{G}$ where $a, b, c, d, e \in_R \mathbb{Z}_p$ and $g \in_R \mathbb{G}$, the modified problem is to distinguish g^{abe+cd^2} from a random element*

$Z \in_R \mathbb{G}$. The advantage of an adversary $\mathcal{A}$ to solve the modified problem is defined as

$$\mathtt{Adv}_{\mathcal{A}}^{Modified}(\lambda) = \Pr[w = \mathcal{A}(g, g^d, g^{cd}, g^{d^2}, g^e, g^{ae}, g^{be}, T_0, T_1, w \in_R \{0,1\}) : T_w = g^{abe+cd^2}, T_{w-1} = Z]$$

Lemma 1. *If an algorithm $\mathcal{A}$ can solve the VoDCBDH problem with the advantage $\mathtt{Adv}_{\mathcal{A}}^{VoDCBDH}(\lambda)$, then we can built an algorithm $\mathcal{S}$ to solve the modified problem with the advantage $\mathtt{Adv}_{\mathcal{S}}^{Modified}(\lambda)$ such that*

$$\mathtt{Adv}_{\mathcal{A}}^{VoDCBDH}(\lambda) \leq \mathtt{Adv}_{\mathcal{S}}^{Modified}(\lambda).$$

Proof. The simulator $\mathcal{S}$ obtains an instance $\hat{\theta} = (\hat{g}, \hat{g}^{\hat{d}}, \hat{g}^{\hat{c}\hat{d}}, \hat{g}^{\hat{d}^2}, \hat{g}^{\hat{e}}, \hat{g}^{\hat{a}\hat{e}}, \hat{g}^{\hat{b}\hat{e}}, T_0, T_1)$. Then $\mathcal{S}$ checks whether $\hat{g}^{\hat{d}} = 1$ or not. If $\hat{g}^{\hat{d}} = 1$, that is $\hat{d} = 0$, the simulator $\mathcal{S}$ returns $w = 0$ if $e(\hat{g}^{\hat{a}\hat{e}}, \hat{g}^{\hat{b}\hat{e}}) = e(T_0, \hat{g}^{\hat{e}})$ or returns $w = 1$ otherwise, and solves $\hat{\theta}$ with the probability of 1. If $\hat{g}^{\hat{d}} \neq 1$, the simulator $\mathcal{S}$ continues and sets $\theta = (g, g^a, g^b, h, h^c, h^d, h^{\frac{1}{a}}, T_0, T_1) = (\hat{g}^{\hat{e}}, \hat{g}^{\hat{a}\hat{e}}, \hat{g}^{\hat{b}\hat{e}}, \hat{g}^{\hat{d}}, \hat{g}^{\hat{c}\hat{d}}, \hat{g}^{\hat{d}^2}, \hat{g}, T_0, T_1)$, it implicitly sets $g = \hat{g}^{\hat{e}}$, $h = \hat{g}^{\hat{d}}$, $a = \hat{a}$, $b = \hat{b}$, $c = \hat{c}$, and $d = \hat{d}$. After that, $\mathcal{S}$ sends θ to $\mathcal{A}$. At some point, the adversary $\mathcal{A}$ outputs a bit w, indicating $T_w = g^{ab}h^{cd}$. Since $T_w = g^{ab}h^{cd} = (\hat{g}^{\hat{e}})^{\hat{a}\hat{b}}(\hat{g}^{\hat{d}})^{\hat{c}\hat{d}} = \hat{g}^{\hat{a}\hat{b}\hat{e}+\hat{c}\hat{d}^2}$, the simulator $\mathcal{S}$ wins with the probability $\mathtt{Adv}_{\mathcal{A}}^{\mathrm{VoDCBDH}}(\lambda)$. Therefore, we have

$$\begin{aligned}\mathtt{Adv}_{\mathcal{S}}^{\mathrm{Modified}}(\lambda) &\geq \Pr[\hat{g}^{\hat{d}} = 1] + \Pr[\hat{g}^{\hat{d}} \neq 1] \cdot \mathtt{Adv}_{\mathcal{A}}^{\mathrm{VoDCBDH}}(\lambda) \\ &\geq \frac{1}{p} + \frac{p-1}{p}\mathtt{Adv}_{\mathcal{A}}^{\mathrm{VoDCBDH}}(\lambda) \geq \mathtt{Adv}_{\mathcal{A}}^{\mathrm{VoDCBDH}}(\lambda).\end{aligned}$$

Lemma 2. *The lower bound of the complexity of the modified problem is stated as follows, querying the group operations and bilinear pairing operations at most q times.*

$$\mathtt{Adv}_{\mathcal{S}}^{Modified}(\lambda) \leq \frac{3(q+9)^2}{p}.$$

Proof. The modified problem is an instance of Decisional Bilinear (P, f)-Diffie-Hellman problem family [27] where $P = (p_1, \ldots, p_7) = (1, d, cd, d^2, e, ae, be)$ and $f = abe + cd^2$. We show that f is not dependent on P by contradiction.

Assume f is dependent on P that by definition in [27] there exists 57 constants $a_{i,j}$, b_k, and c that

$$Q = cf^2 + \sum_{k=1}^{7} b_k p_k f + \sum_{i=1}^{7}\sum_{j=1}^{7} a_{i,j} p_i p_j = 0$$

where at least one of b_k or c is non-zero. We analyze the above equation in two cases.

1. $c \neq 0$: In this case, there is a term $f^2 = a^2b^2e^2 + 2abcd^2e + c^2d^4$ in Q. Furthermore, the term $a^2b^2e^2$ is not in any combination of $p_k f$ or $p_i p_j$, then f^2 cannot be canceled out. Hence, we have $Q \neq 0$ if $c \neq 0$.

2. $c = 0$: In this case, we have $Q = cf^2 + \sum_{k=1}^{7} b_k p_k f + \sum_{i=1}^{7}\sum_{j=1}^{7} a_{i,j} p_i p_j$ where at least one of b_k is non-zero. In other words, Q has at least a term $p_k f = p_k(abe + cd^2) = p_k abe + p_k cd^2$. As $Q = 0$, both two terms $p_k abe$ and $p_k cd^2$ should be canceled out. In the first step, we focus on the term $p_k abe$. There are two methods to cancel the term $p_k abe$.
   (a) To cancel with $p_{k'} f = p_k abe + p_{k'} cd^2$ where $k \neq k'$, we have $p_k abe = p_{k'} cd^2$, that is, $p_k = \theta cd^2$ and $p_{k'} = \theta abe$ for some polynomial θ. Since no such pair of p_k and $p_{k'}$ in P, we cannot cancel $p_k abe$ via $p_{k'} f$.
   (b) To cancel with $p_i p_j$, we have $p_k abe = p_i p_j$. By observing P, the only polynomial which has a is $p_6 = ae$. Thus we have $p_k abe = p_6 p_j \iff p_k b = p_j$. By observing P again, the only polynomial which has b is $p_7 = be$. Thus we have $p_k = e = p_5$.

   Therefore, $p_k abe$ can be canceled out when $k = 5$. To further cancel out $p_5 f$, the term $p_5 cd^2 = cd^2 e$ has to be canceled out. As before, there are two methods to cancel the term $cd^2 e$.
   (a) To cancel with $p_k f = p_k abe + p_k cd^2$ where $k \neq 5$, we have $p_k abe = cd^2 e$. Since the term $\frac{cd^2}{ab}$ is not in P, we cannot cancel out the term $cd^2 e$.
   (b) To cancel with $p_i p_j$, we have $p_i p_j = cd^2 e$. By observing P, the only polynomial, which has c is $p_3 = cd$. Thus we have $p_i p_3 = cd^2 e \iff p_i = de$. Since the term de is not in P, we cannot cancel out the term $cd^2 e$.

   Since it is impossible to cancel out any term $p_k f$, we have $Q \neq 0$ if $c = 0$.

To sum up, it is impossible to make $Q = 0$, which contradicts the assumption. Therefore, we have f is not dependent on P. By the theorem 1 in [27], we directly have the lemma.

By combining the Lemmas 1 and 2, we have

$$\mathtt{Adv}_{\mathcal{A}}^{\mathrm{VoDCBDH}}(\lambda) \leq \mathtt{Adv}_{\mathcal{S}}^{\mathrm{Modified}}(\lambda) \leq \frac{3(q+9)^2}{p}.$$

3.4 Extended Decisional Combined Bilinear Diffie-Hellman Problem

We propose an extension of variant of Decisional Combined Bilinear Diffie-Hellman Problem below.

Definition 7 ***Extended variant of Decisional Combined Bilinear Diffie-Hellman (EVoDCBDH) Assumption:*** *Given* $g, g^a, g^b, g^e, g^f, h^c, h^d, h^{1/d}, h^l \in \mathbb{G}$ *where* $a, b, c, d, e, f, l \in_R \mathbb{Z}_q$ *and* $h = g^e$*, we define the advantage of the adversary in solving the EVoDCBDH problem as*

$$\mathtt{Adv}_{\mathcal{A}}^{EVoDCBDH}(\lambda) = \Pr[w = \mathcal{A}(g, g^a, g^b, g^f, h^c, h^d, h^{1/d}, h^l, T_0, T_1, w \in_R \{0,1\}) : T_w = g^{ab+ecd}, T_{w-1} = g^{bf+edl}]$$

Theorem 3. *We say a EVoDCBDH assumption holds in group* $\mathbb{G}$ *if for any PPT* $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ *is a* negligible *function of the security parameter* λ.

Proof. Let $\mathcal{S}$ denote the VoDCBDH problem solver, who is given $(g^a, g^b, g^e, g^f, h^c, h^d, h^{1/d}, h^l)$, and aims to distinguish $T = g^{ab} \cdot h^{cd}$ from another value $g^{bf} \cdot h^{dl}$. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

- Setup: $\mathcal{S}$ chooses $f, l \in_R \mathbb{Z}_q$ and computes g^f, h^l, then generates other public parameters using the given instances and sends them to $\mathcal{A}$. $\mathcal{S}$ also tosses a random coin w which will be used later in the game.
- Challenge stage: $\mathcal{S}$ returns the challenge T if $b = 0$; Otherwise, returns the value $g^{bf} \cdot h^{dl}$ to $\mathcal{A}$. Note that the value T comes from his own challenger. Finally, $\mathcal{A}$ outputs w' as its guess for w. If $w' = w$, then $\mathcal{S}$ outputs 1; Otherwise, $\mathcal{S}$ outputs 0.
  Probability analysis: Since the value T from its challenger can be either $g^{ab} \cdot h^{cd}$ or R, thus we have

$$\begin{aligned}
\mathtt{Adv}_{\mathcal{S}}^{VoDCBDH} &= \Pr[\mathcal{A} \to 1 \mid T = g^{ab} \cdot h^{cd}] - \Pr[\mathcal{A} \to 1 \mid T = R] \\
&= [\mathtt{Adv}_{\mathcal{A}}^{EVoDCBDH} + 1/2] - [\mathtt{Adv}_{\mathcal{S}}^{VoDCBDH} + 1/2] \\
&= \mathtt{Adv}_{\mathcal{A}}^{EVoDCBDH} - \mathtt{Adv}_{\mathcal{S}}^{VoDCBDH} \\
&\Rightarrow \mathtt{Adv}_{\mathcal{A}}^{EVoDCBDH} = 2 \cdot \mathtt{Adv}_{\mathcal{S}}^{VoDCBDH}.
\end{aligned}$$

3.5 Exponent Challenge Response Signature

We firstly review the Exponent Challenge-Response signature, which will be used in our k-ASH protocol.

Definition 8 ***The Exponential Challenge-Response (XCR) signature scheme*** *[20]: The signer possess a public/secret key pair (g^a, a) $(a \in \mathbb{Z}_q)$. A verifier provides a message m together with a challenge $g^{w'}$ $(w' \in \mathbb{Z}_q$ is chosen by verifier). The signature produced by signer using challenge $g^{w'}$ is defined as $(g^w, g^{w'(w+a \cdot \mathtt{H}(g^w||m))})$ $(w \in \mathbb{Z}_q$ is chosen by signer). Then the verifier accepts a signature pair (g^w, σ) as valid iff $g^w \neq 0$ and $\sigma = (g^w \cdot g^{a \cdot \mathtt{H}(g^w||m)})^{w'}$.*

3.6 Our k-ASH Protocol

Now we present our proposed unlinkable secret handshake with k-time authentication protocol in the two party setting (without loss of generality, we use user $\widehat{A}$ and user $\widehat{B}$ here). It works as follows:

- Setup: TA takes the security parameter λ and the number of handshakes k as input, outputs the master public key $mpk = (g, h, \{g^{t_i}\}_{i=1}^{i=k}, h^\alpha, h^{1/\alpha})$, and the master secret key $msk = (\{t_i\}_{i=1}^{i=k}, \alpha)$. TA also generates four hash functions $\mathtt{H}_1 : \mathbb{G} \times \mathbb{G}_1 \to \mathbb{Z}_q, \mathtt{H}_2 : \{0,1\}^* \to \mathbb{Z}_q, \mathtt{H}_3 : \mathbb{G} \to \mathbb{Z}_q, \mathtt{H}_4 : \mathbb{G} \to \mathbb{Z}_q$ and denotes the bilinear pairing $\hat{\mathtt{e}} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$.
- KeyGen: User $\widehat{A}$ chooses $x_a \in \mathbb{Z}_q$ and computes g^{x_a} as his/her public key.

– Register: User $\widehat{A}$ submits his/her public key g^{x_a} to TA. TA then chooses $w_{a_i} \in \mathbb{Z}_q$ and computes $s_{a_i} = w_{a_i} + \alpha \cdot \mathtt{H}_1(h^{w_{a_i}}||\hat{\mathtt{e}}(g^{x_a}, h^{\alpha})^{t_i})$ and returns a credential set $\{h^{w_{a_i}}\}_{i=1}^{i=k}, \{s_{a_i}\}_{i=1}^{i=k}$ to user $\widehat{A}$. While user $\widehat{A}$ can verify them using the following equations: $\{h^{s_{a_i}} \stackrel{?}{=} h^{w_{a_i}} \cdot h^{\alpha \cdot \mathtt{H}_1(h^{w_{a_i}}||\hat{\mathtt{e}}(h^{\alpha}, g^{t_i})^{x_a})}\}_{i=1}^{i=k}$.
– Handshake (Fig. 1):
  - User $\widehat{A}$ chooses the ephemeral secret key $r_a \in_R \mathbb{Z}_q$, computes $R_a = h^{r'_a} = h^{\mathtt{H}_2(r_a||x_a||s_{a_i})}$ and sends it to user $\widehat{B}$;
  - User $\widehat{B}$ performs the following.
    * Choose the ephemeral secret key $r_b \in_R \mathbb{Z}_q$, computes $R_b = h^{r'_b} = h^{\mathtt{H}_2(r_b||x_b||s_{b_i})}$;
    * Compute $C_{b_i} = \hat{\mathtt{e}}(h^{\alpha}, g^{t_i})^{x_b}$;
    * Compute $\widehat{C_{b_i}} = g^{t_i \cdot x_b} \cdot h^{s_{b_i} \cdot e_b/\alpha}$, where $e_b = \mathtt{H}_3(R_a^{r'_b})$;
    * Send $R_b, g^{t_i}, h^{w_{b_i}}, C_{b_i}, \widehat{C_{b_i}}, e_b$ to user $\widehat{A}$.
  - User $\widehat{A}$ receives the incoming message from user $\widehat{B}$, then performs the following.
    * Verify $e_a = \mathtt{H}_3(R_b^{r'_a}) \stackrel{?}{=} e_b$. If verification fails, reject the session; Otherwise, proceeds;
    * Verify $\hat{\mathtt{e}}(\widehat{C_{b_i}}, h^{\alpha}) \stackrel{?}{=} C_{b_i} \cdot \hat{\mathtt{e}}(h^{w_{b_i}} \cdot h^{\alpha \cdot e_{b_i}}, h^{e_a})$, where $e_{b_i} = \mathtt{H}_1(h^{w_{b_i}}||C_{b_i})$. If verification fails, reject the session; Otherwise, proceed to the next step;
    * Compute the session key $K = \mathtt{H}_4((h^{s_{b_i} \cdot e_b^*} \cdot R_b)^{s_a^*})$, where $e_b^* = \mathtt{H}_3(R_b||e_{b_i}), s_a^* = s_{a_i} \cdot e_a^* + r'_a, e_a^* = \mathtt{H}_3(R_a||e_{a_i}), e_{a_i} = \mathtt{H}_1(h^{w_{a_i}}||C_{a_i})$;
    * Send $g^{t_i}, h^{w_{a_i}}, C_{a_i}, \widehat{C}_{a_i}, e_a$ to user $\widehat{B}$. Note that the computation of $C_{a_i}, \widehat{C}_{a_i}$ by user $\widehat{A}$ follows the same procedures as above.
  - User $\widehat{B}$ verifies the received message using the same method as user $\widehat{A}$, and computes the session key $K = \mathtt{H}_4((h^{s_{a_i} \cdot e_a^*} \cdot R_a)^{s_b^*})$, where $e_a^* = \mathtt{H}_3(R_a||e_{a_i})$, $s_b^* = s_{b_i} \cdot e_b^* + r'_b, e_b^* = \mathtt{H}_3(R_b||e_{b_i})$.

  Note that the computation of session key used the XCR signature from [20].
– Tracing
  If user $\widehat{A}$ used the same credential twice, e.g., $(\widehat{C_{a_i}}, e_{a_i})$ and $(\widehat{C'_{a_i}}, e'_{a_i})$, then anyone can compute $g^{t_i \cdot x_a} = [(g^{t_i \cdot x_a} \cdot h^{s_{a_i} \cdot e_{a_i}/\alpha})^{e'_{a_i}} / (g^{t_i \cdot x_a} \cdot h^{s_{a_i} \cdot e'_{a_i}/\alpha})^{e_{a_i}}]^{1/(e'_{a_i} - e_{a_i})}$, where $e_{a_i} = \mathtt{H}_3(R^{r_a}), e'_{a_i} = \mathtt{H}_3(R'^{r_a})$. That means if user $\widehat{A}$ reused a credential, then user $\widehat{A}$' identity can be revealed since $\hat{\mathtt{e}}(g^{t_i \cdot x_a}, g) = \hat{\mathtt{e}}(g^{t_i}, g^{x_a})$ for public key g^{x_a}.

4 Security Analysis

4.1 Session Key Security

Theorem 4. *The proposed k-ASH protocol achieves* session key security *(*Definition *1) in the random oracle model if the VoCDH assumption is held in the underlying group* $\mathbb{G}$.

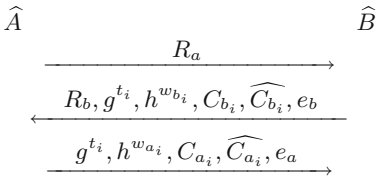

Fig. 1. Handshake

Proof. We define a sequence of games G_i, $i = 0, \cdots, 3$ and let $\mathtt{Adv}_i^{k-ASH}$ denote the advantage of the adversary in game G_i. Assume that $\mathcal{A}$ activates at most m (perhaps $m \geq k$) sessions in each game.

- G_0 This is original game for session key security.
- G_1 This game is identical to game G_0 except that $\mathcal{S}$ will output a random bit if the nonce R_i is used twice by two different instance oracles. Therefore, we have:
$$\left|\mathtt{Adv}_0^{k-ASH} - \mathtt{Adv}_1^{k-ASH}\right| \leq m^2/2^{\lambda} \tag{3}$$
- G_2 This game is identical to game G_1 except that $\mathcal{S}$ will output a random bit if **Forge** event happens where $\mathcal{A}$ made a send query in the form of $(h^{r_0}, g^{t_i}, h^{w_0}, \hat{\mathtt{e}}(h^{\alpha}, g^{t_i})^{x_i}, g^{t_i \cdot x_i} \cdot h^{s_0 \cdot \mathtt{H}_3(R^{*r_0})/\alpha}, \mathtt{H}_3(R^{*\cdot r_0}))$ and an $\mathtt{H}_4$ query with a valid forgery $\sigma = R^{*s_0^*} = R^{*[s_0 \cdot \mathtt{H}_3(h^{r_0} || \mathtt{H}_1(h^{w_0} || \hat{\mathtt{e}}(h^{\alpha}, g^{t_i})^{x_i})) + r_0]}$ for challenge R^*, such that user i is not corrupted (i.e., no Long term secret key reveal query to user i or Master secret key reveal query to TA) when the hash query is made. Then we have:
$$\left|\mathtt{Adv}_1^{k-ASH} - \mathtt{Adv}_2^{k-ASH}\right| \leq \Pr[\mathbf{Forge}] \tag{4}$$

Lemma 3. *The **Forge** event happens only with a negligible probability when the VoCDH assumption is held in* $\mathbb{G}$.

Let $\mathcal{S}$ denote the VoCDH problem solver, who is given $h^a, h^{1/a}, h^b$, and aims to compute h^{ab}. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows:

- Setup stage: $\mathcal{F}$ sets up the game for $\mathcal{A}$ by creating n users (set $\mathcal{U}$) with the corresponding public/secret key pairs $\{X_i, x_i\}_{i=1}^n$. $\mathcal{F}$ randomly selects an index i and guesses that the **Forge** event will happen with regard to user i and session i. $\mathcal{S}$ then sets the mpk as $h^{\alpha} = h^a, h^{1/\alpha} = h^{1/a}$ and generates other public parameters honestly. In addition, $\mathcal{S}$ sets the challenge as $R^* = h^b$ in the guessed session i, $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.
- $\mathcal{S}$ answers $\mathcal{A}$'s queries as follows:
  * If $\mathcal{A}$ issues establish query in the form of (U', X'), such that $U' \notin \mathcal{U}$, then user U' with public key X' will be added to the system.
  * If $\mathcal{A}$ issues a send query in the form of $(h^{r'}, g^{t_i}, h^{w'}, \hat{\mathtt{e}}(h^a, g^{t_i})^{x'}, g^{t_i \cdot x'} \cdot h^{(s' \cdot e')/a})$ to user i, then $\mathcal{S}$ verifies it successfully (notice that $\mathcal{A}$ may corrupt a user with secret key x' and secret signature pair $(h^{w'}, s')$), and next to generating the signatures (h^{w_i}, s_i) as follows:

1. Chooses $s_i, e_i \in_R \mathbb{Z}_q$;
2. Sets $h^{w_i} = h^{s_i}/h^{a \cdot e_i}$;
3. Sets $\mathtt{H}_1(h^{w_i}||C_i) = e_i$, where $C_i = \hat{\mathtt{e}}(h^a, g^{t_i})^{x_i}$.

Then, $\mathcal{S}$ chooses $r_i' \in \mathbb{Z}_q$ and computes $e = \mathtt{H}_3(h^{r' \cdot r_i'})$. Eventually, $\mathcal{S}$ generates the message $(h^{r_i'}, g^{t_i}, h^{w_i}, C_i, g^{t_i \cdot x_i} \cdot h^{(s_i \cdot e)/a}, e)$ and sends it to $\mathcal{A}$.

* If $\mathcal{A}$ issues an ephemeral secret key reveal query to instance oracle $\Pi^i_{U_i}$, then $\mathcal{S}$ returns the ephemeral value r_i ($r_i' = \mathtt{H}_2(r_i||x_i||s_i)$) to $\mathcal{A}$.
* If $\mathcal{A}$ issues a long term secret key reveal query to user j ($\neq i$), then $\mathcal{S}$ returns x_j and secret signatures $\{s_j\}_{j=1}^k$ to $\mathcal{A}$. Note that $\mathcal{S}$ can simulate secret signatures (h^{w_j}, s_j) of user j ($\neq i$) using the same method that described above. If $\mathcal{A}$ issues a long term secret key reveal key query to user i or a master secret key reveal key query to TA, then `abort`.
* Session key reveal query and Test query: $\mathcal{S}$ answers the session key reveal query and the test query by using the session key it has derived during the protocol simulation described above.

- When **Forge** event occurs (i.e., $\mathcal{A}$ outputs: $h^{r_0}, g^{t_i}, h^{w_0}, \hat{\mathtt{e}}(h^a, g^{t_i})^{x_i}$, $g^{t_i \cdot x_i} \cdot h^{(s_0 \cdot \mathtt{H}_3(h^{b \cdot r_0}))/a}, \mathtt{H}_3(h^{b \cdot r_0})$), $\mathcal{S}$ checks whether:
  1. The **Forge** event with respect to user i on challenge h^b;
  2. Verifies:

$$\hat{\mathtt{e}}(g^{t_i \cdot x_i} \cdot h^{(s_0 \cdot \mathtt{H}_3(h^{b \cdot r_0}))/a}, h^a) \stackrel{?}{=} \hat{\mathtt{e}}(h^a, g^{t_i})^{x_i} \cdot \hat{\mathtt{e}}(h^{s_0}, h^{e^*})$$

  Note that $h^{s_0} = h^{w_0} \cdot h^{a \cdot e_1}$, $s_0 = w_0 + a \cdot e_1, e_1 = \mathtt{H}_1(h^{w_0}||\hat{\mathtt{e}}(h^a, g^{t_i})^{x_i})$, $e^* = \mathtt{H}_3(h^{b \cdot r_0})$.

  3. Verifies:

$$\begin{aligned} \hat{\mathtt{e}}(D, h) &= \hat{\mathtt{e}}((h^{s_i \cdot \mathtt{H}_3(h^b||e_i)} \cdot h^b)^{s_0 \cdot \mathtt{H}_3(h^{r_0}||e_1)+r_0}, h) \\ &\stackrel{?}{=} \hat{\mathtt{e}}(h^{s_i \cdot \mathtt{H}_3(h^b||e_i)} \cdot h^b, h^{s_0 \cdot e_0} \cdot h^{r_0}) \end{aligned}$$

  Note that the value D is used to compute session key $K(= \mathtt{H}_4(D))$, $e_0 = \mathtt{H}_3(h^{r_0}||e_1)$.

If all the above conditions hold, $\mathcal{S}$ confirms it as a successful forgery from $\mathtt{H}_4$ and proceeds:

$$\begin{aligned} \sigma_1 &= \frac{D}{(h^{s_0 \cdot e_0} \cdot h^{r_0})^{s_i \cdot \mathtt{H}_3(h^b||e_i)}} \\ &= (h^b)^{s_0 \cdot e_0 + r_0} = h^{b[(w_0 + a \cdot e_1)e_0 + r_0]} \end{aligned}$$

According to the forking lemma [4], by rewinding the adversary twice, $\mathcal{S}$ would obtain four forgeries from $\mathtt{H}_4$, which will be listed below.

$$\begin{aligned} \sigma_1 &= h^{b[(w_0 + a \cdot e_1)e_0 + r_0]}, e_0 = \mathtt{H}_3(h^{r_0}||e_1); \\ \sigma_2 &= h^{b[(w_0 + a \cdot e_1)e_0' + r_0]}, e_0' = \mathtt{H}_3(h^{r_0}||e_1); \\ \sigma_3 &= h^{b[(w_0 + a \cdot e_1')\widehat{e_0} + r_0']}, \widehat{e_0} = \mathtt{H}_3(h^{r_0'}||e_1'); \\ \sigma_4 &= h^{b[(w_0 + a \cdot e_1')\widehat{e_0'} + r_0']}, \widehat{e_0'} = \mathtt{H}_3(h^{r_0'}||e_1'); \end{aligned}$$

Therefore, $\mathcal{S}$ can perform the computation below to obtain a solution to VoCDH.

$$D_1 = (\frac{\sigma_1}{\sigma_2})^{1/(e_0 - e_0')} = h^{b \cdot w_0} \cdot h^{ab \cdot e_1}$$
$$D_2 = (\frac{\sigma_3}{\sigma_4})^{1/(\widehat{e_0} - \widehat{e_0'})} = h^{b \cdot w_0} \cdot h^{ab \cdot e_1'}$$
$$h^{ab} = (\frac{D_1}{D_2})^{1/(e_1 - e_1')}.$$

The simulation performed by $\mathcal{S}$ is perfect. Since at most n users and m sessions in the game, we have:

$$\Pr[\textbf{Forge}] \leq n \cdot m \cdot \mathtt{Adv}_{\mathcal{S}}^{VoCDH}(\lambda) \tag{5}$$

– G_3: This game is identical to game G_2 except that in the test session, we replace the session key $K = \mathtt{H}_4(h^{s_i^* \cdot s_j^*})$ by a random value $r \in \mathbb{Z}_q$. Since we model $\mathtt{H}_4$ as a random oracle, if the event **Forge** does not happen, then we have

$$\mathtt{Adv}_2^{k-ASH} = \mathtt{Adv}_3^{k-ASH} \tag{6}$$

It is easy to see that in game G_3, $\mathcal{A}$ has no advantage, i.e.,

$$\mathtt{Adv}_3^{k-ASH} = 0 \tag{7}$$

Combining the above results together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{k-ASH}(\lambda) \leq m^2/2^{\lambda} + n \cdot m \cdot \mathtt{Adv}_{\mathcal{A}}^{VoCDH}(\lambda)$$

4.2 Anonymity

Theorem 5. *The proposed k-ASH protocol achieves* anonymity (Definition 2) *in the random oracle model if the EVoDCBDH Assumption is held in the underlying group* $\mathbb{G}$.

Proof. Let $\mathcal{S}$ denote a EVoDCBDH problem distinguisher, who is given $(g, h, g^a, g^b, g^f, h^c, h^d, \mathtt{H}^{1/d}, h^l)$, and aims to distinguish $g^{ab} \cdot h^{cd}$ and $g^{bf} \cdot h^{dl}$. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

– Setup: $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating n users. $\mathcal{S}$ sets $h^{\alpha} = h^{1/d}, h^{1/\alpha} = h^d$ (the $msk = (\alpha, 1/\alpha)$ are implicitly set as $(1/d, d)$ respectively), and randomly selects one tag base $g^{t^*} = g^b$ and generates other tag bases honestly (i.e., g^{t_i}, $t_i \in \mathbb{Z}_q$ is chosen by $\mathcal{S}$). In addition, $\mathcal{S}$ randomly chooses users i, j from user set $\mathcal{U}$ and sets $g^{x_i} = g^a, g^{x_j} = g^f$ (the secret keys (x_i, x_j) are implicitly set as (a, f) respectively), and generates public/secret key pair for other users honestly.
– If $\mathcal{A}$ issues a send query in the form of $(R', g^{t_i}, h^{w'}, C_{b'}, \widehat{C_{b'}})$ to user i, then $\mathcal{S}$ performs the simulation as follows.

- $\mathcal{S}$ simulates the signature pair (h^{w_i}, s_i) using the same method that described in Lemma 3;
- $\mathcal{S}$ computes $\widehat{C_i} = g^{a \cdot t_i} \cdot h^{d \cdot s_i \cdot e'_i}$, and $C_i = \hat{\mathrm{e}}(h^{1/d}, g^a)^{t_i}$, where $e'_i = \mathtt{H}_3(R'^{r_i})$, $r_i \in \mathbb{Z}_q$;
- $\mathcal{S}$ generates $R_i = h^{r_i}$ and sets $e_i = \mathtt{H}_1(h^{w_i}||C_i)$;
- $\mathcal{S}$ returns $(R_i, g^{t_i}, h^{w_i}, C_i, \widehat{C_i}, e'_i)$ to user $\mathcal{A}$ as the response.

Note that $\mathcal{S}$ can simulate the response of user j using the same method as above.

– It is easy to see that all queries to other users can be simulated perfectly using the user secret keys, and $\mathcal{S}$ can simulate secret credentials using the same method as described in Lemma 3.

– Challenge: If $\mathcal{A}$ issues a send query in the form of $(R, g^{t_i}, h^{w'}, C'_i, \widehat{C'_i})$ to user i, then $\mathcal{S}$ computes $\widehat{C_i} = (g^{ba} \cdot h^{dc})^{e^*}$ and $C_i = \hat{\mathrm{e}}(\widehat{C_i}, h^{1/d}) / \hat{\mathrm{e}}(h^c, h^{e^*})$, where $e^* = \mathtt{H}_3(R^{r^*})$. Eventually, $\mathcal{S}$ returns $(R^*, g^b, h^{w_i}, C_i, \widehat{C_i}, e^*)$ to $\mathcal{A}$ as the response. Similarly, if $\mathcal{A}$ issues a send query to user j, then $\mathcal{S}$ computes $\widehat{C_j} = (g^{bf} \cdot h^{dl})^{e^*}$ and $C_j = \hat{\mathrm{e}}(\widehat{C_j}, h^{1/d}) / \hat{\mathrm{e}}(h^l, h^{e^*})$, where $e^* = \mathtt{H}_3(R^{r^*})$. Eventually, $\mathcal{S}$ returns $(R^*, g^b, h^{w_j}, C_j, \widehat{C_j}, e^*)$ to $\mathcal{A}$ as the response. Note that $\mathcal{S}$ can perfectly simulate the value $h^{w_i} = h^c / h^{e_i/d}$, and sets $e_i = \mathtt{H}_1(h^{w_i}||C_i)$ for user i, $\mathcal{S}$ also can simulate the value h^{w_j} of user j using the same method.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ can break the EVoDCBDH problem. Hence, we have

$$\mathtt{Adv}_{\mathcal{A}}^{k-ASH} \leq \mathtt{Adv}_{\mathcal{S}}^{EVoDCBDH}(\lambda) \tag{8}$$

5 Extension

We can extend the above k-time ASH protocol in the two party setting to the multiple party setting using the classic BD broadcasting protocol [10]. The Setup, KeyGen, Register and Tracing algorithms are same as the two party setting, except the Handshake algorithm, which will be described below. Note that we suppose at most n users in the multiple party setting.

– **Round 1:** User i computes $R_i = h^{r'_i} = h^{\mathtt{H}_2(r_i||x_i||s_i)}, r_i \in_R \mathbb{Z}_q$ and **broadcasts** $(R_i, g^{t_i}, h^{w_i}, C_i)$. Note that x_i, s_i denote the secret key and the secret credential value of user i, and $C_i = \hat{\mathrm{e}}(h^\alpha, g^{t_i})^{x_i}$. Also notice that the indices are taken module n so that user 0 is user n and user i+1 is user 1.

– **Round 2:** After receiving $n-1$ messages in Round 1, then user i computes $\{\widehat{C_j} = g^{t_i \cdot x_i} \cdot h^{s_i \cdot e_j/\alpha}, e_j = \mathtt{H}_3(R_j^{r'_i})\}_{j=1,j\neq i}^{n-1}$ and $\{h^{s_j} = h^{w_j} \cdot h^{\alpha \cdot \mathtt{H}_1(h^{w_j}||C_j)}\}_{j=1,j\neq i}^{n-1}$. Eventually, user i computes the intermediate key $K_i = \frac{\mathtt{H}_4(h^{s^*_{i+1} \cdot s^*_i})}{\mathtt{H}_4(h^{s^*_{i-1} \cdot s^*_i})}$ and **broadcasts** $(K_i, \{\widehat{C_j}, e_j\}_{j=1,j\neq i}^{n-1})$.

Note that $s^*_i = s_i \cdot e^*_i + r'_i, e^*_i = \mathtt{H}_3(R_i||\mathtt{H}_1(h^{w_i}||C_i))$,
$h^{s^*_{i+1}} = (h^{w_{i+1}} \cdot h^{\alpha \cdot \mathtt{H}_1(h^{w_{i+1}}||C_{i+1})})^{e^*_{i+1}} \cdot R_{i+1}, e^*_{i+1} = \mathtt{H}_3(R_{i+1}||\mathtt{H}_1(h^{w_{i+1}}||C_{i+1}))$,
$h^{s^*_{i-1}} = (h^{w_{i-1}} \cdot h^{\alpha \cdot \mathtt{H}_1(h^{w_{i-1}}||C_{i-1})})^{e^*_{i-1}} \cdot R_{i-1}, e^*_{i-1} = \mathtt{H}_3(R_{i-1}||\mathtt{H}_1(h^{w_{i-1}}||C_{i-1}))$.

– **Key Derivation:** User i verifies the received messages $\{\widehat{C_j}\}_{j\neq i}$ from $n-1$ users (it supports batch verification, see below), if either of them fail, then `abort`; Otherwise, computes the final session key $sk_i = \mathtt{H}_4(h^{s^*_{i-1}\cdot s^*_i})^n \oplus K_i^{n-1} \oplus K_{i+1}^{n-2} \cdots \oplus K_{i-2})$.

1. Batch Verification. User i is able to batch verify the received n-1 messages from n-1 users using the small exponents test in [3,13].

$$\begin{aligned}\hat{\mathtt{e}}(\prod_{j=1}^{n-1}\widehat{C_j}^{\delta_j}, h^{\alpha}) &= \hat{\mathtt{e}}(\prod_{j=1}^{n-1} g^{t_i\cdot x_j\cdot\delta_j}\cdot h^{s_j\cdot e_j\cdot\delta_j/\alpha}, h^{\alpha})\\ &= \prod_{j=1}^{n-1}\hat{\mathtt{e}}(g^{t_i\cdot x_j\cdot\delta_j}, h^{\alpha})\cdot\hat{\mathtt{e}}(\prod_{j=1}^{n-1} h^{s_j\cdot e_j\cdot\delta_j}, h)\\ &\stackrel{?}{=} \prod_{j=1}^{n-1} C_j^{\delta_j}\cdot\hat{\mathtt{e}}(\prod_{j=1}^{n-1} h^{s_j\cdot\delta_j}, h^{e_j}).\end{aligned}$$

where $\delta_j \in \mathbb{Z}_q, e_j = \mathtt{H}_3(R_j^{r'_i})$ and $j \in [1, j\neq i, \cdots, n-1]$. If batch verification fail, then `abort`; Otherwise, proceeds.

2. Correctness Check.

$$\begin{aligned}sk_i &= \mathtt{H}_4(h^{s^*_{i-1}\cdot s^*_i})^n \oplus K_i^{n-1} \oplus K_{i+1}^{n-2}\cdots\oplus K_{i-2}\\ &= \mathtt{H}_4(h^{s^*_{i-1}\cdot s^*_i})^n \oplus \frac{\mathtt{H}_4(h^{s^*_{i+1}\cdot s^*_i})^{n-1}}{\mathtt{H}_4(h^{s^*_{i-1}\cdot s^*_i})^{n-1}}\\ &\oplus \frac{\mathtt{H}_4(h^{s^*_{i+2}\cdot s^*_{i+1}})^{n-2}}{\mathtt{H}_4(h^{s^*_i\cdot s^*_{i+1}})^{n-2}}\cdots\oplus\frac{\mathtt{H}_4(h^{s^*_{i-1}\cdot s^*_{i-2}})}{\mathtt{H}_4((h^{s^*_{i-3}\cdot s^*_{i-2}})}\\ &= \mathtt{H}_4(h^{s^*_{i-1}\cdot s^*_i})\oplus\mathtt{H}_4((h^{s^*_i\cdot s^*_{i+1}})\cdots\oplus\mathtt{H}_4((h^{s^*_{i-2}\oplus s^*_{i-1}}).\end{aligned}$$

It is easy to see that all users compute the same key.

The k-time ASH protocol in the multiple party setting also achieved session key security, anonymity and public traceability. In particular, the security analysis (including session key security and anonymity) in the two party setting can be extended to the multiple party setting.

6 Conclusion

In this paper, we proposed a k-time authenticated secret handshake protocol based on the k-time tag bases and anonymized Schnorr signature. We also defined the formal security models for session key security and (full) anonymity, and proved the security of the proposed k-ASK protocol under our proposed complexity assumptions which have been proved hard in the generic bilinear group model.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (61602396, 61572303), the Fundamental Research Funds for the Central Universities under Grant GK201702004.

References

1. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, J.: Secret handshakes from pairing-based key agreements. In: 2003 IEEE Symposium on Security and Privacy (S&P 2003), 11–14, Berkeley, CA, USA, pp. 180–196, May 2003
2. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In: Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, pp. 419–428 (1998)
3. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998). doi:10.1007/BFb0054130
4. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: ACM, CCS 2006, pp. 390–399 (2006)
5. Park, S.B., Kang, M.S., Lee, S.J.: Authenticated key exchange protocol secure against offline dictionary attack and server compromise. In: Li, M., Sun, X.-H., Deng, Q., Ni, J. (eds.) GCC 2003. LNCS, vol. 3032, pp. 924–931. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24679-4_154
6. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). doi:10.1007/3-540-48329-2_21
7. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, pp. 57–66 (1995)
8. Bohli, J., Vasco, M.I.G., Steinwandt, R.: Secure group key establishment revisited. Int. J. Inf. Sec. **6**(4), 243–254 (2007)
9. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005). doi:10.1007/11426639_26
10. Burmester, M., Desmedt, Y.G.: Efficient and secure conference-key distribution. In: Lomas, M. (ed.) Security Protocols 1996. LNCS, vol. 1189, pp. 119–129. Springer, Heidelberg (1997). doi:10.1007/3-540-62494-5_12
11. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). doi:10.1007/3-540-44987-6_28
12. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret handshakes from CA-oblivious encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30539-2_21
13. Ferrara, A.L., Green, M., Hohenberger, S., Pedersen, M.Ø.: Practical short signature batch verification. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 309–324. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00862-7_21
14. Gu, J., Xue, Z.: An improved efficient secret handshakes scheme with unlinkability. IEEE Commun. Lett. **15**(2), 259–261 (2011)
15. Huang, H., Cao, Z.: A novel and efficient unlinkable secret handshakes scheme. IEEE Commun. Lett. **13**(5), 363–365 (2009)
16. Jarecki, S., Kim, J., Tsudik, G.: Group secret handshakes or affiliation-hiding authenticated group key agreement. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 287–308. Springer, Heidelberg (2006). doi:10.1007/11967668_19

17. Juels, A., Weis, S.A.: Defining strong privacy for RFID. ACM Trans. Inf. Syst. Secur. **13**(1), 7 (2009)
18. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: ACM, CCS 2005, pp. 180–189 (2005)
19. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45146-4_7
20. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). doi:10.1007/11535218_33
21. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75670-5_1
22. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). doi:10.1007/0-387-34805-0_22
23. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_18
24. Sun, D., Cao, Z.: On the privacy of Khan et al.'s dynamic id-based remote authentication scheme with user anonymity. Cryptologia **37**(4), 345–355 (2013)
25. Teranishi, I., Furukawa, J., Sako, K.: k-times anonymous authentication (extended abstract). In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 308–322. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30539-2_22
26. Xu, S., Yung, M.: k-anonymous secret handshakes with reusable credentials. In: ACM, CCS 2004, pp. 158–167 (2004)
27. Zhang, S., Yang, G., Mu, Y.: Linear encryption with keyword search. In: Liu, J.K., Steinfeld, R. (eds.) ACISP 2016. LNCS, vol. 9723, pp. 187–203. Springer, Cham (2016). doi:10.1007/978-3-319-40367-0_12