4-2020

A new construction for linkable secret handshake

Yangguang TIAN
Singapore Management University, ygtian@smu.edu.sg

Yingjiu LI

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Nan LI

Guomin YANG

See next page for additional authors

Citation

1

Published in Computer Journal, 2020 April, 63 (4), 536-548.
https://doi.org/10.1093/comjnl/bxz095

A New Construction for Linkable Secret Handshake

YANGGUANG TIAN[1], YINGJIU LI[1], ROBERT H. DENG[1], NAN LI[2], GUOMIN YANG[3] AND ZHENG YANG*[4]

[1]*School of Information Systems, Singapore Management University, Singapore*
[2]*School of Electrical Engineering and Computing, University of Newcastle, Australia*
[3]*School of Computing and Information Technology, University of Wollongong, Australia*
[4]*Information Systems Technology and Design, Singapore University of Technology and Design, Singapore*
Email: zheng_yang@sutd.edu.sg

In this paper, we introduce a new construction for linkable secret handshake that allows authenticated users to perform handshake anonymously within an allowable times. We define formal security models for the new construction, and prove that it can achieve session key security, anonymity, untraceability and linkable affiliation hiding. In particular, the proposed construction ensures that: 1) anyone can trace the real identities of dishonest users who perform handshakes for more than k times; and 2) an optimal communication cost between authorized users is achieved by exploiting the proof of knowledges.

Keywords: Linkable Secret Handshake, Anonymity, Public Tracing, Communication Cost

1. INTRODUCTION

Secret handshake (SH) [1] is an important privacy-enhancing technique, and has been recommended by some standardized cryptographic protocols, such as TLS and IKE [2]. Specifically, SH allows authorized users of the same organization to establish a shared key in an *anonymous* manner, and the only information they are able to know is the peer belongs to the same organization (i.e., membership). To ensure that authorized users authenticate each other's membership simultaneously, the shared key must be derived from a *credential* (or group token), which is issued by the same authority of the organization.

There are two types of SH based on the usage of credentials: linkable and unlinkable. Linkable SH (LSH) means that an authorized user performs multiple handshakes using the same credential. In particular, user's anonymity remains secure even if multiple handshakes from the same user can be linked. While the unlinkable SH may assume that an authorized user is given a set of (one-time) credentials and each credential is supposed to be used once for each handshake. The unlinkable SH from one-time credential can easily guarantee that multiple handshakes from the same user cannot be linked, however, it is unscalable (e.g., a user may easily deplete her credentials). In this work, we focus on LSH [3, 4, 5, 6, 7, 8], which is particularly useful if authorized users wish to be recognized across different handshake sessions.

The LSH protocols can be further investigated. First, the credential issuer is given too much power as the revocation is totally controlled by him (i.e., he can arbitrarily revoke anonymity). By contrast, public traceability is a desirable property [9], which is required in many anonymous authentication systems such as e-cash [10] and cryptocurrencies [11]. For example, e-cash scheme allows one to trace a cheating user (or double-spender) without help from the credential issuer. Second, while reusing the credentials is desired [12], none of LSH protocols was ever considered the issue of restricting the issued credentials. Third, an optimal communication overhead (or bandwidth) is essential in real-time communication applications as it will determine the execution time between authorized users [8]. Therefore, the main goal of this work is to design a linkable SH with a k-time reusable credential (kLSH) while achieving optimal communication overhead. In particular, user's anonymity will be publicly revoked if her credential is used beyond k times.

The kLSH can be applied to various collaborative and group-based applications, such as ad-hoc mobile networks. Let us consider an intelligent transport system where a number of users wish to share real-time traffic information (or path optimization) among themselves. First, each user obtains a credential from a group authority after paying a registration fee that is proportional to k. Later, the mobile users can form ad-hoc groups using the k-time credential, and exchange the relevant traffic information between them. Such transportation system enjoys the security and privacy guarantees as the LSH protocol provided. In addition, it can detect any dishonest user who tries to misuse the k-time credential and continues to form a group without topping up. Furthermore, the kLSH can also be implemented in the information-sharing scenarios, such as covert communications [13] and flow watermarks [14]. Specifically, the kLSH can provide an enhanced security/privacy guarantee to the existing end-to-end communications in the internet.

Our contributions. The main contributions of this work are summarized as follows.

- *New Construction.* We propose a new kLSH construction for linkable secret handshakes such that anyone is able to trace a dishonest user who uses her credential beyond the allowed number of times.
- *Communication Cost.* The proposed kLSH achieves an *optimal* communication cost between authorized users through a variant of the proof of equality of two discrete logarithms [15]. Optimal means that a minimal communication cost is achieved to meet our design goal, which will be shown in the efficiency analysis.
- *Security Models.* We present the formal security definition for linkable secret handshakes with public tracing. We prove that the proposed kLSH can achieve session key security, anonymity, untraceability and linkable affiliation-hiding.

Overview of Techniques. We now explain our key technical issues. First, to ensure untraceability against credential issuer (or group authority) [16, 6, 7], we use a blind signature scheme proposed in [10], which is a blind version of Schnorr signature scheme [17]. The derived message-signature pair (m, s) after registration has the following features: 1) the signature generation requires the involvement of both user and group authority (i.e., neither user nor group authority have exclusive control over the generation of s); 2) the message m implicitly includes user's public key $\mathtt{pk}$, so the user must use the correct secret key $\mathtt{sk}$ when performing handshakes between authorized users (due to the unforgeability of blind signature scheme); and 3) the signature s acts as a secret credential which is later used to derive a shared session key between authorized users. Once a user outputs a message-signature pair (m, s), the registration terminates. Thus, the message-signature pair can be regarded as public pseudonym id and secret credential $id.cred$ of an authorized user. We stress that group authority *cannot* link user's public pseudonym id to her real public key $\mathtt{pk}$ due to the blindness of blind signature scheme.

Second, a shared session key is established between authorized users after a successful handshake. In particular, the generation of session key is based on the implicitly authenticated key exchange such as HMQV protocol (see Section 3.4). Specifically, if a user i later receives a message together with a MAC tag that verifies with respect to the key generated from the session, then user i is assured that the MAC tag must have been generated by another authorized user j (with pseudonym id_j) who is also registered to the same group authority. Since the public pseudonym id_i and secret credential $id_i.cred$ pair can be regarded as long-term public/secret key pair (g^{s_i}, s_i), the correctness of session key via the HMQV protocol is held.

Moreover, we use Shamir's secret sharing scheme to ensure k-time (untraceable) handshakes. The secret to be shared is a blind factor x (*not* user's secret key $\mathtt{sk}$) which was used in the generation of blind signature. Two types of insider attacks needed to be addressed: 1) user i uses the incorrect secret for sharing; or 2) user i uses the inconsistent secret shares (denote one secret share as $f(x)$) in a handshake session. We use user i and user j as an example to illustrate. In the first case, we let user j verify user i's secret shares by inputting user i's public pseudonym id_i for each handshake session. So user i must use the correct x for secret sharing due to the unforgeability of blind signature (i.e., g^x is also included in the pseudonym id_i). For the second case, we use the proof of equality of discrete logarithms [18] to ensure the consistency of secret shares. That is, user i should not only release her shares but also provide a proof of correctness for each share released. We note that multiple handshake sessions from the same user can be linked, *but* her real public key (or identity) $\mathtt{pk}$ cannot be traced/identified within k-time handshakes.

Third, to trace a dishonest user (e.g., user i) who uses the secret credential $id.cred$ beyond k times, we let user i release the share $g^{f(x)}$ instead of the exponent $f(x)$, then user j can reconstruct the secret g^x if k+1 related shares $\{g^{f(x)}\}$ are accumulated. As a result, a dishonest user i's real public key can be identified by using the bilinear maps. That is, user i's public key $\mathtt{pk}_i$ is linked to her pseudonym id_i. We stress that the bilinear maps is solely used for public tracing, and all security guarantees we considered in this work remain secure under the bilinear map due to the SXDH assumption (see Section 3.1).

1.1. Related Work

Key Exchange. Burmester and Desmedt (BD) [19] introduced several key exchange protocols in the multi-party setting, including star-based, tree-based, broadcast-based and cyclic-based protocols. Later, a few generic transformations [20, 21] were proposed to convert passive-secure group key exchange protocols into active-secure ones. On the security models for key exchange protocols, Bellare and Rogaway [22] introduced the first complexity-theoretic security model for key exchange in the symmetric-key setting. Canetti and Krawczyk [23] later 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 [24, 25, 26] of CK model have also been proposed to allow the adversary to obtain either long-term secret key or ephemeral secret key of the challenge session.

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 credential approach. Afterwards, Castelluccia et al. [3] constructed a more efficient scheme under the standard computational Diffie-Hellman assumption. Both schemes did not provide the unlinkability property. Then, Xu and Yung [2] provided an unlinkable scheme with a weaker anonymity: k-anonymity (including k-unlinkability). That is, an adversary can infer that a session participant is one out of certain k users in the worst case.

For achieving the full anonymity, Jarecki et al. [4] proposed two group secret handshake protocols using BD protocol [19]. In particular, their second construction can achieve full unlinkability using a one-time credential. To prevent any dishonest user from misusing the one-time credential, Tian et al. [27] proposed a k-time unlinkable secret handshake protocol. Specifically, the proposed solution achieves full unlinkability based on a k-size one-time credentials set, but the group authority is fully trusted.

The untraceablility against group authority is another privacy guarantee required in secret handshakes. Kawai et al. [16] split the group authority into (non-colluding) authorities: one is responsible for registration and issuing credential, the other is responsible for tracing users based on protocol transcript. Meanwhile, Manulis et al. [6] addressed the issue of untraceability against a group authority. Specifically, they use blinded RSA signature scheme at Add (or register) stage, and their solution is based on the construction in [5]. Similarly, Manulis et al. [7] proposed a discrete-logarithm based construction, and used the blinded Schnorr signature to tackle a group authority.

An independent research line was considering a practical application, where each user has multiple membership credentials (i.e., group discovery). In particular, the linkable secret handshake proposed by Manulis and Pottering [8] has a log-linear time complexity $\mathcal{O}(n \cdot \log n)$ (where n is the number of group affiliations per user), in contrast to the existing solutions with linear complexity [28, 29]. It is claimed to be suitable for resources-constraint devices, but the expensive operations involved in the handshake algorithm, since the underlying non-interactive key distribution scheme is based on pairings.

To highlight our distinction, we compare our construction with some existing linkable/unlinkable secret handshake protocols in Table 1: it shows that our proposed construction has session key security with forward secrecy, anonymity against insiders, untraceability against group authority, public tracing and linkable affiliation-hiding. We stress that the proposed construction can be regarded as a step forward from linkable secret handshakes [5, 6, 7, 8] in a single group setting.

2. SECURITY MODEL

In this section, we present the security models for linkable secret handshakes. As mentioned in the introduction, a secure linkable secret handshake protocol should achieve session key security, anonymity, untraceability and linkable affiliation hiding.

States. We define a system user set $\mathcal{U}$ with n users, i.e. $|\mathcal{U}| = n$. We say an instance oracle Π^i_{id} (e.g., session i of user id) may be used or unused, and a user has unlimited number of instances called oracles. The oracle is considered as unused if it has never been initialized. Each unused oracle Π^i_{id} can be initialized with a secret key $\mathtt{sk}$. 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^i_{id} learns its partner id pid^i_{id} 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 $state^i_{id}$ is maintained by the oracle. The oracle Π^i_{id} remains in the processing state until it collects enough information to compute the session key SK^i_{id}. As soon as SK^i_{id} is computed Π^i_{id} accepts and terminates the protocol execution meaning that it would not send or receive further messages. If the protocol execution fails then Π^i_{id} terminates without having accepted. We assume that each user can only sequentially execute the protocol (i.e., a new session of a user is activated if and only if the last session of that user is terminated).

Partnering. We denote the i-th session of a user id by Π^i_{id}. Let the partner identifier pid^i_{id} include the identities of participating users (including id) in the i-th session of user with the condition that $\forall id_j \in \mathsf{pid}^i_{id}$. In other words, pid^i_{id} is a collection of recognized participants by the instance oracle Π^i_{id}. Note that the recognized participants mean that authorized users are communicating with user at some sessions. We also define sid^i_{id} as the unique session identifier belonging to user pk of session i. Specifically, $\mathsf{sid}^i_{id} = \{m_j\}^n_{j=1}$, where $m_j \in \{0,1\}^*$ is the message transcript among users in pid^i_{id}. We say two instance oracles Π^i_{id} and $\Pi^j_{id'}$ are partners if and only if $\mathsf{pid}^i_{id} = \mathsf{pid}^j_{id'}$ and $\mathsf{sid}^i_{id} = \mathsf{sid}^j_{id'}$.

2.1. System Model

A linkable secret handshake protocol consists of the following algorithms:

- Setup: A group authority (GA) takes security parameter λ as input, outputs a master public/secret

Scheme	LIN	AKE	LAH	UT	HEA	ECC	**TR**
XY [2]	×	×	✓	×	×	×	×
JKT [5]	✓	✓	✓	×	×	×	×
MPT [6]	✓	✓	✓	✓	×	×	×
MPT [7]	✓	✓	✓	✓	×	✓	×
MP [8]	✓	✓	✓	×	✓	✓	×
TZYMY [27]	×	✓	×	×	✓	✓	✓
Ours	✓	✓	✓	✓	×	✓	✓

TABLE 1. The comparison between different functionality of linkable/unlinkable secret handshake protocols. LIN means linkable SH. AKE means the session key security that includes forward secrecy. LAH means the linkable affiliation hiding (i.e., group membership is hidden from outsiders). UT means untraceable towards an untrusted group authority. HEA means the handshake algorithm involves expensive operations (e.g., pairing). ECC means the protocol supports ECC group setting w.r.t. bandwidth complexity. **TR** means tracing any dishonest user in public.

key pair ($\mathtt{msk}, \mathtt{mpk}$). The GA indicates an untrusted authority in this work.

- KeyGen: A user takes master public key $\mathtt{mpk}$ as input, outputs a secret/public key pair ($\mathtt{sk}, \mathtt{pk}$). GA outputs an empty pseudonym revocation list Ł.
- Register: This is an interactive algorithm that executed between a user and the GA. It takes the master secret key $\mathtt{msk}$ and a public key $\mathtt{pk}$ of one user as input, outputs a public pseudonym and secret credential pair $(id, id.cred)$ on $\mathtt{pk}$. Note that the user will become an authorized user after registration, and the interaction between a user and the TA is assumed to be authentic.
- Handshake: This is an interactive algorithm that executed by authorized users. Each user takes her secret key $\mathtt{sk}$, her credential $id.cred$, the master public key $\mathtt{mpk}$ and Ł as input, outputs a shared secret key SK if and only if her counterparts are non-revoked and authorized users.
- Tracing: The algorithm takes two handshake transcripts of one user as input, outputs user's public key $\mathtt{pk}$. GA updates the pseudonym revocation list Ł by adding a pseudonym id associated with $\mathtt{pk}$.

2.2. Session Key Security

We define the session key security model for LSH protocols, in which each user obtains a credential associated with her public key from GA, and establishes a session key using the given credential. The model is defined via a game between a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ (i.e., challenger). $\mathcal{A}$ is an active attacker with full control of the communication channel among all the users.

- Setup: $\mathcal{S}$ first generates a master public/secret key pair ($\mathtt{mpk}, \mathtt{msk}$) for GA and long-term secret/public key pairs $\{(\mathtt{sk}_i, \mathtt{pk}_i)\}_{i=1}^{n}$ for n users by running the corresponding KeyGen algorithms. In addition, $\mathcal{S}$ generates pseudonym and credential pairs $\{id_j, id_j.cred\}_{j=1}^{n}$ for all authorized users by running the Register algorithms. Eventually, $\mathcal{S}$ returns all users' long-term public key and pseudonym to $\mathcal{A}$. $\mathcal{S}$ also tosses a random coin b which will be used later in the game.
- Training: $\mathcal{A}$ can make the following queries in arbitrary sequence to $\mathcal{S}$.
  - Establish: $\mathcal{A}$ is allowed to register a user with public key $\mathtt{pk}_i'$ and pseudonym id_i' (id_i' cannot be identical to any existing identities). If a user is registered by $\mathcal{A}$, then we call this user as *dishonest*; Otherwise, it is *honest*.
  - Send: If $\mathcal{A}$ issues a send query in the form of (id, i, m) to simulate a network message for the i-th session of user id, then $\mathcal{S}$ would simulate the reaction of instance oracle Π_{id}^{i} upon receiving message m, and return to $\mathcal{A}$ the response that Π_{id}^{i} would generate; If $\mathcal{A}$ issues a send query in the form of $(id,' start')$, then $\mathcal{S}$ creates a new instance oracle Π_{id}^{i} and returns to $\mathcal{A}$ the first protocol message.
  - Session key reveal: $\mathcal{A}$ can issue a session key reveal query to an accepted instance oracle Π_{id}^{i}. If the session is accepted, then $\mathcal{S}$ returns 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_{id}^{i}, then $\mathcal{S}$ returns all ephemeral secret values contained in Π_{id}^{i} at the moment the query is asked.
  - Long-term secret key reveal: If $\mathcal{A}$ issues a long-term secret key reveal query to user i, then $\mathcal{S}$ returns the long-term secret key $\mathtt{sk}_i$ to $\mathcal{A}$.
  - Long-term secret credential reveal: If $\mathcal{A}$ issues a long-term secret credential reveal query to user i, then $\mathcal{S}$ returns the secret credential $id_i.cred$ to $\mathcal{A}$.
  - Master secret key reveal: If $\mathcal{A}$ issues a master secret key reveal query to GA, then $\mathcal{S}$ returns the master secret keys $\mathtt{msk}$ to $\mathcal{A}$.

- Test: This query can only be made to an accepted and *fresh* (as defined below) session i of a user. 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 $\mathcal{A}$.

  $\mathcal{A}$ can generate the secret credential $id_i.cred$ of user i after issuing the Master secret key reveal query to GA. *$\mathcal{A}$ can continue to issue other queries after* 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 $\mathcal{S}$ outputs 1; Otherwise, $\mathcal{S}$ outputs 0.

Freshness. We say an *accepted* instance oracle Π^i_{id} 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^i_{id} or its accepted partnered instance oracle $\Pi^j_{id'}$;
- $\mathcal{A}$ issues both Long-term secret credential reveal query to id' s.t. $id' \in \mathsf{pid}^i_{id}$ and Ephemeral secret key reveal query for an instance $\Pi^j_{id'}$ partnered with Π^i_{id};
- $\mathcal{A}$ issues Long-term secret credential reveal query to user id' s.t. $id' \in \mathsf{pid}^i_{id}$ prior to the acceptance of instance Π^i_{id} and there exists no instance oracle $\Pi^j_{id'}$ partnered with Π^i_{id}. Note that the Master secret key reveal query to GA is equivalent to the Long-term secret credential reveal to all users in pid^i_{id}.

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

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

Definition 2.1. We say a LSH 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 insider adversary is not allowed to identify who are the handshake users, with the condition that the authorized users authenticate with each other within k times. The formal anonymity game between an insider adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ is defined as follows:

- Setup: $\mathcal{S}$ first generates a master public/secret key pair $(\mathtt{mpk}, \mathtt{msk})$ for GA and long-term secret/public key pairs $\{(\mathtt{sk}_i, \mathtt{pk}_i)\}^n_{i=1}$ for n users by running the corresponding KeyGen algorithms. In addition, $\mathcal{S}$ generates a set of pseudonym and credential $\{id_j, id_j.cred\}^{\mathrm{k}}_{j=1}$ for each user by running the Register algorithms. Eventually, $\mathcal{S}$ returns all users' long-term public key and pseudonyms to $\mathcal{A}$. $\mathcal{S}$ also tosses a random coin b which will be used later in the game. We denote the original n authorized users set as $\mathcal{U}$.
- Training: $\mathcal{A}$ is allowed to issue at most n-2 Long term secret key reveal queries and all other queries to $\mathcal{S}$. We denote the honest (i.e., uncorrupted) user set as $\mathcal{U}'$.
- Challenge: $\mathcal{A}$ randomly selects two users $\mathtt{pk}_i, \mathtt{pk}_j \in \mathcal{U}'$ as challenge candidates, then $\mathcal{S}$ removes them from $\mathcal{U}'$ and simulates $\mathtt{pk}^*_b$ to $\mathcal{A}$ by either $\mathtt{pk}^*_b = \mathtt{pk}_i$ if $b = 1$ or $\mathtt{pk}^*_b = \mathtt{pk}_j$ if $b = 0$.

  Let $\mathcal{A}$ interact with $\mathtt{pk}^*_b$. *$\mathcal{S}$ is allowed to execute at most k handshake sessions for* $\mathtt{pk}_i, \mathtt{pk}_j$ *throughout the entire game.*

$$\mathcal{A} \Leftrightarrow \mathtt{pk}^*_b = \begin{cases} \mathtt{pk}_i & b = 1 \\ \mathtt{pk}_j & b = 0 \end{cases}$$

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

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

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

Definition 2.2. We say a LSH protocol has *anonymity* if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ is a *negligible* function of the security parameter λ.

2.4. Untraceability

Informally, untraceability requires that an adversary (e.g., untrusted GA) is not able to trace the real identities of handshake users, with the condition that the handshake users authenticate with each other within k times even if the linkable credentials (e.g., pseudonyms) are used. We stress that the untrusted GA is not allowed to enroll a phantom or revoked user during registration. The formal untraceability game between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ is defined as follows.

- Setup: $\mathcal{S}$ first generates a master public/secret key pair $(\mathtt{mpk}, \mathtt{msk})$ for GA and long-term secret/public key pairs $\{(\mathtt{sk}_i, \mathtt{pk}_i)\}^n_{i=1}$ for n users by running the corresponding KeyGen algorithms. In addition, $\mathcal{S}$ generates public pseudonym and secret credential $(id_i, id_i.cred)$ for n users by running the corresponding Register algorithms. Eventually, $\mathcal{S}$ returns all public keys and pseudonyms to $\mathcal{A}$. $\mathcal{S}$ also tosses a random coin b which will be used later in the game.
- Training: $\mathcal{A}$ interacts with all users via a set of oracle queries (as defined in the session key security model), and outputs two public keys $(\mathtt{pk}_0, \mathtt{pk}_1)$, where $(\mathtt{pk}_0, \mathtt{pk}_1)$ denotes two new distinct and non-revoked users. In particular, $\mathcal{A}$ runs the Register algorithms on these two users $(\mathtt{pk}_0, \mathtt{pk}_1)$ which are admitted to become authorized users, outputs the public pseudonyms (id_0, id_1), respectively. *The game does not proceed until the corresponding* Register *algorithms on behalf of* $\mathtt{pk}_0$ *and* $\mathtt{pk}_1$ *outputs the secret credentials* $(id_0.cred, id_1.cred)$.

- Challenge: $\mathcal{A}$ is given the challenge public pseudonym id_b, and $\mathcal{A}$ continues to interact with all users via all oracle queries until it terminates and outputs the bit b'. In particular, $\mathcal{S}$ *is allowed to execute at most k handshake sessions with regard to the challenge pseudonym id_b*. We define the advantage of $\mathcal{A}$ in the above game as

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

Definition 2.3. We say a LSH protocol has *untraceability* if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ is a *negligible* function of the security parameter λ.

Remark. Untraceability means that the real identities of authorized users cannot be recognized by any parties (including untrusted GA) within k times handshake sessions. That is, authorized user's real identities are not linked to their credentials within k times handshake sessions, even if their long-term secret keys are leaked to adversaries (this is prohibited in the anonymity model).

2.5. Linkable Affiliation Hiding

Informally, an outsider adversary aims to learn the handshake users' affiliation by corrupting users and by learning certain handshake sessions were successful. The linkable affiliation hiding (LAH) model is simulation-based [5, 6]. That is, the real protocol execution is indistinguishable from an idealized one performed by a simulator $\mathcal{SIM}$ that simulates handshake executions without knowing participants' affiliation. We formally define a linkable affiliation-hiding LAH security game between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ (with help of another simulator $\mathcal{SIM}$) as follows.

- Setup: The simulator $\mathcal{S}$ first chooses a secret bit b at random, and answers $\mathcal{A}$'s queries according to the bit b. We call a group as *trivially intrudable* is the group was not Setup honestly or if $\mathcal{A}$ corrupted some pseudonyms id generated in response to some Register queries. When $b = 1$, $\mathcal{S}$ answers $\mathcal{A}$'s queries by following the protocol specification faithfully. Below we show that the $\mathcal{SIM}$ helps the simulator $\mathcal{S}$ to answer queries made by $\mathcal{A}$ when $b = 0$, such that simulating the handshake sessions on behalf of honest pseudonyms only if they are not trivially intrudable.
  - Register and Long-term secret key reveal: These queries are answered honestly without involving $\mathcal{SIM}$ *unless* there exists some still running handshake session sid which is not trivially intrudable. In this later case, Register and Long-term secret key reveal queries are ignored if their input is such that after processing there queries the group becomes trivially intrudable.
  - Handshake: If the group is trivially intrudable, then $\mathcal{S}$ answers the query to the invoked session correctly (denote the invokes session as case-1 session). If the group is not trivially intrudable, then $\mathcal{S}$ invokes $\mathcal{SIM}$ and relays its reply (denote the invoked session as case-2 session).
  - Send: If the send query involves a case-1 session sid, then $\mathcal{S}$ correctly answers its query. $\mathcal{S}$ simulates session key SK as follows: If there exists a partnering session sid$'$, then the session key is set as SK' (i.e., session key of sid$'$); Otherwise, if sid is fresh (as defined in the session key security), then the session key is set as $SK \in \{0,1\}^\lambda$ (a random key is drawn from the session key space); Otherwise, $\mathcal{S}$ sets the session key of sid according to the protocol specification. If sid is a case-2 session, then $\mathcal{S}$ invokes $\mathcal{SIM}$ and relays its reply.
  - Session key reveal: If the session key reveal query to a case-1 session sid and the session is accepted, then $\mathcal{S}$ returns the session key correctly; if session sid is not accepted, then $\mathcal{S}$ returns '$\perp$'. If the session key reveal query to a case-2 session sid, the session is accepted and there exists a partnering session sid$'$, then $\mathcal{S}$ performs as follows: if session sid is not set but sid$'$ is, then returns SK' to $\mathcal{A}$; If both sessions sid and sid$'$ are not set, then returns $SK \in \{0,1\}^\lambda$ to $\mathcal{A}$.

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

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

Definition 2.4. We say a LSH protocol has *linkable affiliation hiding* if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ is a *negligible* function of the security parameter λ.

3. PRELIMINARIES

3.1. Complexity Assumptions

Decisional Diffie-Hellman (DDH) Problem. Let $\mathbb{G}$ denote a cyclic group with prime order q and generator g. Given g, g^a, g^b and a random element $Z \in \mathbb{G}$, where (randomly chosen) $a, b \in \mathbb{Z}_q$, decide $Z \overset{?}{=} g^{ab}$.

Bilinear Symmetric External Diffie-Hellman (SXDH) Assumption [30]: Given two generators g_1, g_2 and a bilinear maps $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_{\mathbb{T}}$, the advantage of the adversary $\mathcal{A}$ in solving a bilinear SXDH problem is defined as

$$\begin{aligned}\mathtt{Adv}_{\mathcal{A}}(\lambda) \quad = \quad & \Pr[w \in \{0,1\}, \mathcal{A}(g_i, g_i^a, g_i^b, T_w = g_i^{ab}, \\ & T_{1-w} \in \mathbb{G}_i) = w]\end{aligned}$$

where $a, b \in \mathbb{Z}_q$ and $i \in [1, 2]$. The bilinear SXDH assumption holds if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ is a negligible function of the security parameter λ.

Knowledge of Exponent Assumption (KEA) [31]. Given a cyclic group $\mathbb{G}$ of prime order q, for any adversary $\mathcal{A}$ that takes input (q, g, g^a) and returns (B, Z) with $Z = B^a$, there exists an "extractor" $\mathcal{A}'$, which given the same inputs as $\mathcal{A}$ returns b such that $g^b = B$.

3.2. Equality of Discrete Logarithms

First, we present the proof of knowledge of equality of representation module two primes, which is introduced by Chaum and Pederson [15], the goal is to prove that $\log_{g_1} h_1 = \log_{g_2} h_2$, for generators $g_1, g_2, h_1, h_2 \in \mathbb{G}$. We denote it as $DLEQ(g_1, h_1, g_2, h_2)$, and the interaction between prover and verifier is described as follows.

- The prover sends $R_1 = g_1^w$ and $R_2 = g_2^w$, where $w \in \mathbb{Z}_q$.
- The verifier sends a challenge $e \in \mathbb{Z}_q$ to the prover.
- The prover sends $p = w - x \cdot e$, where $x = \log_{g_1} h_1$.
- The verifier checks that $R_1 \overset{?}{=} g_1^p \cdot h_1^e$ and $R_2 \overset{?}{=} g_2^p \cdot h_2^e$. If the verification passes, it outputs "1", otherwise, it outputs "0".

Non-Interactive Version. We then show the non-interactive version of $DLEQ(g_1, h_1, g_2, h_2)$ by applying the Fiat-Shamir's technique [32].

- The prover sends a challenge-response value pair (e, p) to the verifier, where $e = \mathtt{H}(h_1, h_2, R_1, R_2)$.
- The verifier checks that $e \overset{?}{=} \mathtt{H}(h_1, h_2, R_1', R_2')$, where $R_1' = g_1^p \cdot h_1^e$ and $R_2' = g_2^p \cdot h_2^e$.

3.3. Blind Signature

The blind signature (BS) scheme is executed between a user and a singer, and the output of BS scheme includes a message/signature pair, which will be used in the Register algorithm. The message and signature will be regarded as authorized user's pseudonym and credential, respectively.

Definition 3.1 (Blindness [33]). The game is involved two users and a (adversary controlled) signer which is running as follows. Firstly, the singer is running the KeyGen algorithm to obtain $(\mathtt{pk}, \mathtt{sk})$; Secondly, the signer outputs a pair of messages (m_0, m_1); Thirdly, the signer engages in two parallel (and arbitrarily interleaved) interactive protocols with two users, with inputs m_b and m_{1-b} respectively (b is a random bit which is hidden from the signer). If both users output valid signatures on their respective messages, then the signer is also given these two signatures; Otherwise, output *fail* (we do not insist this happens). Eventually, the signer outputs a bit b' as its guess for b. We define the advantage of user in the above game as

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

A BS scheme is said to have *blindness* if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(\lambda)$ is a *negligible* function of the security parameter λ. Meanwhile, the formal definition on the *unforgeability* of BS scheme is referred to [33], we omit it here.

3.4. Implicitly Authenticated Key Exchange

We assume that user i (with pseudonym id_i) and user j (with pseudonym id_j) wish to establish a shared secret key, *implicit authentication* means that user i knows that the only other party who could compute the shared secret key as her is user j (and vice versa). It is worth noting that, the implicit authentication is a novel and efficient solution to the Diffie-Hellman protocol which secures against man-in-the-middle attacks, in contrast to the method that signs all protocol messages using digital signatures [20]. We do not give a survey on the implicitly AKE protocols, we just present a variant of well-known HMQV protocol [24] below. It takes user i's ephemeral secret key and long-term secret key, and user j's ephemeral public key and long-term public key as input, outputs a shared secret key between user i and user j.

- User i and user j exchange ephemeral public key $R_i = g^{t_i}$ and $R_j = g^{t_j}$, respectively;
- User i computes $\sigma_i = (R_j \cdot \mathtt{pk}_j^e)^{(t_i + d \cdot \mathtt{sk}_i)}$, user j computes $\sigma_j = (R_j \cdot \mathtt{pk}_i^d)^{(r_j + e \cdot \mathtt{sk}_j)}$, where $d = \mathtt{H}(R_i, id_j)$, $e = \mathtt{H}(R_j, id_i)$;
- Both users generate: $SK_{i/j} \xleftarrow{\text{AKE}} \mathtt{H}'(\mathsf{sid}_{i/j}, \sigma_{i/j})$, where $\mathsf{sid}_{i/j} = (id_i, id_j, R_i, R_j)$. Here $\mathsf{sid}_{i/j}$ denote user i or user j's session identifier; the same for $\sigma_{i/j}$ and $SK_{i/j}$.

Remark. The NAXOS technique may be implemented to the HMQV protocol, so that achieving an extended CK (eCK) security [25]. Specifically, user i sends an ephemeral public key in the form of $R_i = g^{\mathtt{H}(t_i, \mathtt{sk}_i)}$ to user j (and the same method is applied to user j). The detailed description of eCK model and NAXOS technique are referred to [25].

4. THE PROPOSED CONSTRUCTION

In this section, we present the detailed construction of kLSH.

- Setup: GA takes security parameter λ as input, outputs master secret key $\mathtt{msk} = (y, f, h)$ and master public key $\mathtt{mpk} = (g^y, g_1, g_2, g_1^y, g_2^y)$, where $g_1 = g^f, g_2 = g^h$. GA also generates the hash functions $\mathtt{H}_1 : \{0,1\}^* \to \mathbb{G}, \mathtt{H}_2 : \mathbb{G} \to \mathbb{Z}_q, \mathtt{H}_3 : \{0,1\}^* \times \mathbb{G} \to \{0,1\}^{2\lambda}$. Let $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_{\mathbb{T}}$ be the bilinear maps.
- KeyGen: User i generate a long-term secret/public key pair $(\mathtt{sk}_i, \mathtt{pk}_i) = (\mathtt{x}_i, g_1^{\mathtt{x}_i})$. GA publishes an empty pseudonym revocation list Ł.

- Register: A user i and GA perform an interactive algorithm that described in Figure. 1. Eventually, user i outputs a public pseudonym $id_i \leftarrow (\alpha_i, \beta_i, r_i)$ and a secret credential $id_i.cred \leftarrow s_i$ (underline part).

- Handshake: We use two authorized users (i, j) to illustrate our handshake algorithm. The description of algorithm is shown in Figure. 2.

- Tracing: Suppose a public pseudonym $id_i = (\alpha_i, \beta_i, r_i)$ is used by a dishonest user for k+1 times. Then, anyone can get k+1 correct shares about the secret $\underline{g_2^{x_{(i,1)}}}$. That means if user i misused her credential, then user i's real identity can be revealed (i.e., her real identity is linked to her public pseudonym). Specifically, $\hat{e}(g_1^{\mathbf{x}_i}, g_2^{x_{(i,1)}}) \overset{?}{=} \hat{e}(\underline{g_1^{\mathbf{x}_i \cdot x_{(i,1)}}}, g_2)$, where $\underline{g_1^{\mathbf{x}_i \cdot x_{(i,1)}}} = \beta_i / g_2^{x_{(i,1)}}$. Eventually, GA updates the pseudonym revocation list Ł by including id_i.

Correctness. The verification of the blind signature s_i in the Register algorithm is shown as follows.

$$\begin{aligned} \mathtt{H}_1(\alpha_i, \beta_i) &= \beta_i^{-s_i} \cdot \alpha_i^{\mathtt{H}_2(\mathtt{H}_1(\alpha_i,\beta_i),r_i)} \cdot r_i \\ &= (g_1^{\mathbf{x}_i} \cdot g_2)^{x_{(i,1)} \cdot (-b \cdot y \cdot m'_i - b \cdot k - a)} \\ &\quad \cdot (g_1^{\mathbf{x}_i} \cdot g_2)^{y \cdot x_{(i,1)} \cdot b \cdot m'_i} \cdot m_i \cdot (g_1^{\mathbf{x}_i} \cdot g_2)^{x_{(i,1)} \cdot a} \\ &\quad \cdot (g_1^{\mathbf{x}_i} \cdot g_2)^{k \cdot b \cdot x_{(i,1)}} = \underline{m_i} \end{aligned}$$

where $\alpha_i \leftarrow (g_1^{\mathbf{x}_i} \cdot g_2)^{y \cdot x_{(i,1)}} = (g_1^y)^{\mathbf{x}_i \cdot x_{(i,1)}} \cdot (g_2^y)^{x_{(i,1)}}$.

Next, we show the detailed techniques used in the Handshake algorithm, including the proof of knowledges (PoK) and authenticated key exchanges (AKE).

1. PoK: The proof of knowledge is used to guarantee the correctness and the authenticity of secret shares.

    - User i chooses a random polynomial f of degree at most k with coefficients $\{a_l\} \in \mathbb{Z}_q$, for $1 \le l \le$ k, and publishes the related k-size commitments $C_l = (g_1^{\mathbf{x}_i} \cdot g_2)^{a_l}$, for $1 \le l \le$ k. Specifically, $f(x) \overset{\mathrm{k}}{\leftarrow} a_0 + \sum_{l=1}^{\mathrm{k}} a_l \cdot x^l$.

    - User i (prover) runs the non-interactive equality of discrete logarithms to obtain a proof: $F_i = (e_{(i,1)}, p_{(i,0)}, p_{(i,1)}) \leftarrow DLEQ((g_1^{\mathbf{x}_i} \cdot g_2), (g_1^{\mathbf{x}_i} \cdot g_2)^{f(x_i)}, g_2, g_2^{f(x_i)})$. Specifically, $e_{(i,1)} = \mathtt{H}_2(X_i, Y_i, a_i, b_i)$, where $a_i \leftarrow (g_1^{\mathbf{x}_i} \cdot g_2)^{w_i}, b_i \leftarrow R_j^{w_i}, p_{(i,1)} \leftarrow w_i - f(x_i) \cdot e_{(i,1)}, p_{(i,0)} = p_{(i,1)}/x_{(i,1)}, w_i \in \mathbb{Z}_q$ and $f(x_i) \overset{\mathrm{k}}{\leftarrow} \underline{x_{(i,1)}} + a_1 \cdot x_i + a_2 \cdot x_i^2 + \cdots + a_{\mathrm{k}} \cdot x_i^{\mathrm{k}}$, $x_i = \mathtt{H}_2(g_2^{t_i \cdot t_j})$.

    - User j (verifier) computes $a'_i \leftarrow \beta_i^{p_{(i,0)}} \cdot \underline{X_i}^{e_{(i,1)}}, b'_i \leftarrow R_j^{p_{(i,1)}} \cdot Y_i^{e_{(i,1)}}$ and verifies that $e_{(i,1)} \overset{?}{=} \mathtt{H}_2(X_i, Y_i, a'_i, b'_i)$. It outputs "1" if the verification passes (we denote it as Verify$(F_i) = 1$); Otherwise, it outputs "0". In particular, the value $X_i \leftarrow (g_1^{\mathbf{x}_i} \cdot g_2)^{f(x_i)}$ is computed from public commitments $\{C_l\}$ and $\beta_i \in id_i$), more concretely

$$\begin{aligned} \underline{\beta_i} \prod_{l=1}^{\mathrm{k}} C_l^{x_i^l} &= \underline{(g_1^{\mathbf{x}_i} \cdot g_2)^{x_{(i,1)}}} \cdot (g_1^{\mathbf{x}_i} \cdot g_2)^{a_1 \cdot x_i} \\ &\quad \cdot (g_1^{\mathbf{x}_i} \cdot g_2)^{a_2 \cdot x_i^2} \cdots (g_1^{\mathbf{x}_i} \cdot g_2)^{a_{\mathrm{k}} \cdot x_i^{\mathrm{k}}} \\ &= (g_1^{\mathbf{x}_i} \cdot g_2)^{f(x_i)}. \end{aligned}$$

2. AKE: Authenticated users rely on the implicitly authenticated key exchange protocol to derive a shared session key. The correctness of the shared session key $SK_{(i,j)}$ is shown below.

$$\begin{aligned} (SK_{(i,j)}, \mu_i) &= \mathtt{H}_3[\mathsf{sid}_i, \sigma_i] \\ &= \mathtt{H}_3[\mathsf{sid}_i, (R_j \cdot \mathtt{H}_1(\alpha_j, \beta_j)^{e_i})^{t_i + d_i \cdot s_i}] \\ &= \mathtt{H}_3[\mathsf{sid}_i, (g_2^{t_j} \cdot m_j^{e_j})^{t_i + d_i \cdot s_i}] \\ &= \mathtt{H}_3[\mathsf{sid}_i, g_2^{(t_j + e_j \cdot s_j) \cdot (t_i + d_i \cdot s_i)}] \end{aligned}$$

where $e_j = \mathtt{H}_2(R_j, id_i), d_i = \mathtt{H}_2(R_i, id_j)$ and $m_j = g_2^{s_j}$.

5. SECURITY ANALYSIS AND EFFICIENCY ANALYSIS

THEOREM 5.1. *The proposed kLSH protocol achieves* session key security *in the random oracle model if the underlying HMQV protocol is session key secure.*

Proof. We define a sequence of games $\mathbb{G}_i$, $i = 0, \cdots, 3$ and let $\mathtt{Adv}_i^{LSH}$ denote the advantage of the adversary in game $\mathbb{G}_i$. Assume that $\mathcal{A}$ activates at most $n(\lambda)$ sessions in each game. For simplicity, we ignore the key confirmation for perfect forward security (PFS) in the following and subsequent proofs.

- $\mathbb{G}_0$ This is original game for session key security.

- $\mathbb{G}_1$ This game is identical to game $\mathbb{G}_0$ except that $\mathcal{S}$ will output a random bit if user i and user j accept, but $\mathsf{pid}_i \neq \mathsf{pid}_j, \mathsf{sid}_i \neq \mathsf{sid}_j$. Since n users involved in this game, we have:

$$\left|\mathtt{Adv}_0^{kLSH} - \mathtt{Adv}_1^{kLSH}\right| \le n \cdot n(\lambda)^2 / 2^\lambda \tag{1}$$

- $\mathbb{G}_2$ This game is identical to game $\mathbb{G}_1$ except the following difference: $\mathcal{S}$ randomly chooses $g \in [1, m]$ as a guess for the index of the test session. $\mathcal{S}$ will output a random bit if $\mathcal{A}$'s test query does not occur in the g-th session. Therefore we have

$$\mathtt{Adv}_1^{kLSH} = n(\lambda) \cdot \mathtt{Adv}_2^{kLSH} \tag{2}$$

- $\mathbb{G}_3$ This game is identical to game $\mathbb{G}_2$ except that in the g-th session, $\mathcal{S}$ replaces the real session key by a random key. Below we show the difference between $\mathbb{G}_2$ and $\mathbb{G}_3$ is negligible under the assumption that HMQV is eCK secure (with weak PFS).

User i		GA
Input :$(\mathtt{sk}_i, \mathtt{pk}_i)$		**Input** :$(\mathtt{msk}, \mathtt{mpk})$ $(\delta_1, \delta_2) = (g^k, (g_1^{\mathbf{x}_i} \cdot g_2)^k)$ where $k \in \mathbb{Z}_q$
	$\xleftarrow{\delta_1, \delta_2}$	
$x_{(i,1)}, a, b \in \mathbb{Z}_q$ $\alpha_i \leftarrow (g_1^{\mathbf{x}_i} \cdot g_2)^{y \cdot x(i,1)}, \beta_i \leftarrow (g_1^{\mathbf{x}_i} \cdot g_2)^{x(i,1)}$ $m_i \leftarrow \mathtt{H}_1(\alpha_i, \beta_i), r_i \leftarrow m_i \cdot \beta_i^a \cdot \delta_2^{b \cdot x(i,1)}$ $m_i' = \mathtt{H}_2(m_i, r_i)/b$		
	$\xrightarrow{m_i'}$	$s_i' = k + y \cdot m_i'$
	$\xleftarrow{s_i'}$	
If $g^{s_i'} \neq g^{y \cdot m_i'} \cdot \delta_1$, then reject $s_i = s_i' \cdot b + a$ **Output** : $id_i \leftarrow \alpha_i, \beta_i, r_i$ $\underline{id_i.cred} \leftarrow \underline{s_i}$		

FIGURE 1. Register.

User i		User j
Input :$(id_i, id_i.cred)$ $t_i \in \mathbb{Z}_q, R_i \leftarrow g_2^{t_i}$		**Input** :$(id_j, id_j.cred)$ $t_j \in \mathbb{Z}_q, R_j \leftarrow g_2^{t_j}$
	$\xleftarrow{R_j, id_j}$	
$f(x_i) \xleftarrow{\text{k}} x_{(i,1)}$ $X_i \leftarrow (g_1^{\mathbf{x}_i} \cdot g_2)^{f(x_i)}, Y_i \leftarrow R_j^{f(x_i)}$ $F_i \xleftarrow{\text{PoK}} \{(f(x_i)) : X_i \wedge Y_i\}$ $(SK_{(i,j)}, \mu_i) \xleftarrow{\text{AKE}} \mathtt{H}_3(\mathsf{sid}_i, \sigma_i)$		
	$\xrightarrow{R_i, id_i, Y_i, F_i, \mu_i}$	
		$(SK_{(i,j)}, \mu_j) \xleftarrow{\text{AKE}} \mathtt{H}_3(\mathsf{sid}_j, \sigma_j)$ If Verify$(F_i) \neq$ "1" or $\mu_i \neq \mu_j$, then abort $p(x_j) \xleftarrow{\text{k}} x_{(j,1)}$ $X_j \leftarrow (g_1^{\mathbf{x}_j} \cdot g_2)^{f(x_j)}, Y_j \leftarrow R_i^{f(x_j)}$ $F_j \xleftarrow{\text{PoK}} \{(f(x_j)) : X_j \wedge Y_j\}$
	$\xleftarrow{Y_j, F_j, \mu_j}$	
If Verify$(F_j) \neq$ "1" or $\mu_i \neq \mu_j$, then abort		

FIGURE 2. Handshake. $id_{i/j} \notin \text{Ł}$, $\mathsf{sid}_{i/j} = (id_i, id_j, R_i, R_j, Y_i, Y_j)$ and $x_{i/j} = \mathtt{H}_2(g_2^{t_i \cdot t_j})$. Here $id_{i/j}$ denote user i or user j's session identifier; the same for $\mathsf{sid}_{i/j}$ and $x_{i/j}$; $SK_{(i,j)}$ denote the shared session key between user i and user j.

Let $\mathcal{S}$ denotes an attacker against the HMQV protocol, who is given corresponding oracles in the sense of eCK security, aims to distinguish a real session key from a random key. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

– Setup: $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating n users with the corresponding public/secret key pairs $\{(\mathtt{pk}_i, \mathtt{sk}_i)\}$ from his oracle queries (i.e., long-term secret key query). $\mathcal{S}$ honestly generates master public/secret key pair $(\mathtt{mpk}, \mathtt{msk})$ for GA. In addition, $\mathcal{S}$ generates user's pseudonym and credential by running the Register algorithms honestly. Eventually, $\mathcal{S}$ sends all public keys and pseudonyms to $\mathcal{A}$.

– $\mathcal{S}$ answers $\mathcal{A}$'s queries as follows.

  ∗ If $\mathcal{A}$ issues a send query in the form of (R_j, id_j) to user i, then $\mathcal{S}$ will perform the simulation as follows.

    · Forward R_j to his challenger and obtain an ephemeral public key R_i from his send oracle.

    · Generate a proof F_i (as the Handshake algorithm described) on a secret share which is derived from the simulated credential.

    · Return (R_i, id_i, Y_i, F_i) to $\mathcal{A}$ as the response.

    · Simulate the final session key using either session key reveal oracle or test oracle w.r.t. g-th session, and $\mathsf{sid} = (id_i, id_j, R_i, R_j, Y_i, Y_j)$.

  Note that if $\mathcal{A}$ issues a send query in the form of $(id_i,' start')$ to user i, then $\mathcal{S}$ returns (R', id_i) to $\mathcal{A}$, where R' is randomly chosen by $\mathcal{S}$.

  ∗ If $\mathcal{A}$ issues an ephemeral key reveal query to $\mathcal{S}$, then $\mathcal{S}$ makes an ephemeral key reveal query to its own oracle to get t_i and returns it to $\mathcal{A}$.

  ∗ If $\mathcal{A}$ issues a long-term key reveal query to user i, then $\mathcal{S}$ returns $\mathtt{sk}_i$ to $\mathcal{A}$. If $\mathcal{A}$ issues a master secret key reveal query to $\mathcal{S}$, then $\mathcal{S}$ returns $\mathtt{msk}$

to $\mathcal{A}$.

  * $\mathcal{S}$ answers the session key reveal query and test query by using the session key SK_i it has derived during the protocol simulation described above.

Note that in the test session $\mathcal{S}$ will get either a real session key or a random key from his own test oracle. If it is the real session key, then the simulation is consistent with $\mathbb{G}_2$; Otherwise, the simulation is consistent with $\mathbb{G}_3$. Therefore, if the advantage of $\mathcal{A}$ is significantly different in $\mathbb{G}_2$ and $\mathbb{G}_3$, $\mathcal{S}$ can break the eCK security of HMQV protocol. Hence, we have

$$\left|\mathtt{Adv}_2^{kLSH} - \mathtt{Adv}_3^{kLSH}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{HMQV}(\lambda) \quad (3)$$

Combining the above results together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{kLSH}(\lambda) = n \cdot n(\lambda)^2/2^\lambda + n(\lambda) \cdot \mathtt{Adv}_{\mathcal{S}}^{HMQV}(\lambda).$$

Theorem 5.2. *The proposed kLSH protocol achieves* anonymity *if the SXDH assumption is held in the underlying group* $\mathbb{G}_1$.

Proof. Let $\mathcal{S}$ denote a SXDH problem distinguisher, who is given $(g_1, g_1^a, g_1^b, g_1^z)$ and a bilinear maps $\hat{\mathsf{e}}: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathbb{T}$, aims to distinguish g_1^{ab} and $Z \xleftarrow{R} \mathbb{G}_1$. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

- Setup: $\mathcal{S}$ generates master public/secret key pair $(y, f, h \in \mathbb{Z}_q)$ by running the Setup algorithm honestly. $\mathcal{S}$ also generates the long-term public key of challenge user $\mathtt{pk}_i$ as $g_1^{\mathbf{x_i}}$ ($\mathbf{x_i} \in \mathbb{Z}_q$ is chosen by $\mathcal{S}$), and sets the long-term public key of challenge user $\mathtt{pk}_j$ as $g_1^{b\cdot e}$ ($e \in \mathbb{Z}_q$ is chosen by $\mathcal{S}$).
- Register: $\mathcal{S}$ simulates the Register algorithm on challenge users $(\mathtt{pk}_i, \mathtt{pk}_j)$ as follows.
  - For user $\mathtt{pk}_i$, $\mathcal{S}$ simulates a k-size pseudonym and credential set $\{id_i, id_i.cred\}$ honestly using user's secret key $\mathbf{x_i}$.
  - For user $\mathtt{pk}_j$, $\mathcal{S}$ simulates it as follows.
    * Upon receiving a pair $(\delta_1, \delta_2) = (g^k, (g_1^{b\cdot e} \cdot g_2)^k)$, $\mathcal{S}$ first executes the extractor defined in the KEA assumption to extract the value k. Then, $\mathcal{S}$ computes $\alpha_j = g_1^{z\cdot e\cdot r\cdot y} \cdot g_1^{a\cdot r\cdot y\cdot h/f}, \beta_j = g_1^{z\cdot e\cdot r} \cdot g_1^{a\cdot r\cdot h/f}, m_j = \mathtt{H}_1(\alpha_j, \beta_j)$. Note that the randomness implicitly sets as $x_{(j,1)} = a \cdot r$ and $g = g_1^{f^{-1}}, g_2 = g_1^{h/f}$, where $r \in \mathbb{Z}_q$ is chosen by $\mathcal{S}$.
    * $\mathcal{S}$ chooses blind factors $a_j, b_j \in \mathbb{Z}_q$, computes $r_j = m_j \cdot (g_1^{z\cdot e\cdot r} \cdot g_1^{a\cdot r\cdot h/f})^{a_j} \cdot (g_1^{z\cdot e\cdot r} \cdot g_1^{a\cdot r\cdot h/f})^{b_j\cdot k}, m_j' = \mathtt{H}_2(m_j, r_j)/b_j$ and sends m_j' to $\mathcal{A}$.
    * On receiving $id_j.cred'$ from $\mathcal{A}$, $\mathcal{S}$ computes $id_j.cred = id_j.cred' \cdot b_j + a_j$ and stores $(\alpha_j, \beta_j, r_j, id_j.cred)$ as a valid pseudonym and credential pair.

  In the end, $\mathcal{S}$ will generate two pseudonym and credential set $\{id_i, id_i.cred\}$ and $\{id_j, id_j.cred\}$ for user $\mathtt{pk}_i$ and user $\mathtt{pk}_j$, respectively.
- Training: It is easy to see that all queries to users can be simulated perfectly using the long-term secret credentials and master secret keys.
- Challenge: Upon receiving (R^*, id_j) $(R^* = g_2^{t^*}, t^* \in \mathbb{Z}_q)$ from $\mathcal{S}$ as an authentication request, $\mathcal{A}$ then issues a send query in the form of (R, id, Y, F) to user j, where $R = g_2^t$, then $\mathcal{S}$ performs the simulation as follows
  - computes $f(x_j) \xleftarrow{k} id_j.cred + a_1 \cdot x_i + a_2 \cdot x_i^2 + \cdots + a_k \cdot x_i^k$ and $x_j = \mathtt{H}_2(R^{t^*})$.
  - computes $X^* \leftarrow (g_1^{b\cdot e} \cdot g_2)^{f(x_j)}, Y^* \leftarrow R^{f(x_j)}$.
  - generates a proof $F_j \xleftarrow{\text{PoK}} \{(f(x_j)) : X^* \wedge Y^*\}$.
  - sends (Y^*, F_j) to $\mathcal{A}$ as the response.

  Similarly, $\mathcal{S}$ can simulate the interaction with user i using the same method as described above. In particular, $\mathcal{S}$ can perfectly simulate the shared session keys between challenge users and adversary $\mathcal{A}$ using the simulated secret credentials.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ could break the SXDH problem.

Theorem 5.3. *The proposed kLSH protocol achieves* untraceability *if the BS scheme has the blindness.*

Proof. We let $\mathcal{S}$ denote an attacker against the blindness of BS scheme, who is given the public/secret key pair of Signer $(\mathtt{pk}^*, \mathtt{sk}^*)$ and two interactive Signer and User oracles, aims to break the blindness property. Since Handshake sessions are completely independent of the challenge users when running them, and depends solely on the given public pseudonym id and secret credential $id.cred$. Even the long-term secret key reveal query to user's long-term public key $\mathtt{pk}$ does not reveal the owning user of given pseudonym id. Therefore, we mainly analyze the Register algorithm below.

- Setup: $\mathcal{S}$ generates long-term public/secret key pair $(\mathtt{pk}_i, \mathtt{sk}_i)$ for n users by running the corresponding KeyGen algorithms. $\mathcal{S}$ sets master public/secret key pair of GA as $(\mathtt{mpk}, \mathtt{msk}) = (\mathtt{pk}^*, \mathtt{sk}^*)$. $\mathcal{S}$ generates public pseudonym and secret credential $(id_i, id_i.cred)$ for n users by running the corresponding Register algorithms. Eventually, $\mathcal{S}$ publishes all user's public keys and pseudonyms.
- $\mathcal{S}$ can simulate all queries to the existing users made by $\mathcal{A}$ using master secret key and user's long-term secret keys. When $\mathcal{A}$ issues Register algorithm on two distinct users $(\mathtt{pk}_0, \mathtt{pk}_1)$, then $\mathcal{S}$ randomly chooses two messages (m_0, m_1) with respect to two new users, and invokes his challenger (i.e., two

parallel interactive Signer and User oracles) to obtain two signatures (or credentials) $(s(m_b), s(m_{1-b}))$. In particular, $\mathcal{S}$ computes the corresponding public pseudonyms $(id_b, id_{1-b}) \leftarrow F(s(m_b), s(m_{1-b}))$ where F denotes an efficient key generation function (e.g., $id_b \leftarrow g^{s(m_b)}$). $\mathcal{S}$ picks one of public pseudonyms as challenge id_b and forwards it to $\mathcal{A}$. We also allow $\mathcal{A}$ to access both $s(m_b)$ and $s(m_{1-b})$ during the challenge stage.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ could break the blindness property of BS scheme.

Theorem 5.4. *The proposed LSH protocol achieves the* linkable affiliation hiding *in the random oracle model if the underlying HMQV protocol is session key secure and the BS scheme is unforgeable.*

Proof. The security proof is similar to the proof of linkable affiliation hiding described in [6, 7]. Recall that when $b = 0$, we let $\mathcal{SIM}$ simulate the protocol executions *without knowing honest participants' affiliation (or credentials).* Specifically, $\mathcal{SIM}$ will answer the Handshake, Send and Session key reveal queries made by $\mathcal{A}$.

- Handshake. $\mathcal{SIM}$ chooses $id_i.cred \leftarrow s_i \in_R \mathbb{Z}_q$ as user i's secret credential. $\mathcal{SIM}$ outputs the state $(id_i, \mathtt{sk}_i, id_i.cred)$. Note that $\mathcal{SIM}$ can simulate the pseudonym/credential pair $(id_i, id_i.cred)$ successfully using the same method as described in [17] (i.e., simulate the signing oracle), and user i's secret key $\mathtt{sk}_i$ is known to $\mathcal{SIM}$.

- Send. $\mathcal{SIM}$ chooses $\mu_i \in_R \mathbb{Z}_q$ and outputs $(R_i, id_i, Y_i, F_i, \mu_i)$, where F_i is honestly generated by $\mathcal{SIM}$ since user i's secret key $\mathtt{sk}_i$ is known to $\mathcal{SIM}$. Meanwhile, $\mathcal{SIM}$ can successfully simulate the shared session keys using the simulated pseudonym/credential pair according to the protocol specification.

We stress that $\mathcal{SIM}$ simulates the handshake sessions that are run on behalf of honest pseudonyms in the groups that are *intact* (i.e., not trivially intrudable). The difference between real protocol run and simulated run by $\mathcal{SIM}$ is presented as follows:

1. The simulated secret credential $id_i.cred$ might differ. The difference between two methods to generate secret credential is bounded by the probability of adversary to forge a BS scheme, which is negligible in λ. Therefore, the advantage of adversary in this case is reduce to $\mathtt{Adv}_{\mathcal{A}}^{BS}(\lambda)$.

2. The simulated session keys SK_i might be inconsistent with real ones. This difference is bounded by the session key security of underlying HMQV protocol such that no outsider adversary can distinguish a real session key from random, unless adversary actively takes part in protocol sessions as a participant of the group (such attack is disallowed in the LAH model since $\mathcal{SIM}$ only simulates the *intact* groups). Therefore, the advantage of adversary in this case is reduce to $\mathtt{Adv}_{\mathcal{A}}^{HMQV}(\lambda)$.

3. The simulated key confirmation message μ_i might differ. Since the key confirmation message μ_i derived from the underlying HQMV protocol which is secure, the difference is only bounded by the probability of adversary to find a collision in a collision-resistant hash function. Therefore, the advantage of adversary is reduced to $\mathtt{Adv}_{\mathcal{A}}^{H}(\lambda)$.

   By combing the above cases together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{kLSH}(\lambda) = \mathtt{Adv}_{\mathcal{A}}^{BS}(\lambda) + \mathtt{Adv}_{\mathcal{A}}^{HMQV}(\lambda) + \mathtt{Adv}_{\mathcal{A}}^{H}(\lambda).$$

Therefore, the adversary cannot distinguish the real protocol executions from the simulated ones.

Efficiency Analysis. We evaluate the total number of transferred bits when running the Handshake algorithm. In practice security parameters $\lambda = 80$, and $\lambda' = 1024$ (standard group setting) or $\lambda' = 160$ (ECC group setting) would be chosen. The length of user i's pseudonym id_i is $3 \cdot \lambda'$ bits, proof of knowledge F_i is $3 \cdot \lambda'$ bits and Diffie-Hellman instance R_i is λ' bits. So the total number of transferred bits between user i and user j is $2 \times (8 \cdot \lambda' + \lambda)$ (including a transmitted Y_i whose length is λ' bit, and a key confirmation message μ_i whose length is λ). If $q < 2^{160}$, then the overall communication cost (without key confirmation messages) is 320 bytes. The communication cost is not only a constant value, but also the smallest (or optimal) one to achieve our complete design goal. We then compare our kLSH with a closely related LSH protocol [7], who was also claimed the smallest (and constant) bandwidth complexity. Specifically, the total number of transferred bits in [7] is $2 \times (2 \cdot \lambda' + \lambda)$, and our kLSH requires more λ' bits because: 1) it has pseudonym size three times the pseudonym in [7]; and 2) the proof of knowledge is required due to the Shamir's secret sharing scheme for public tracing.

For the computational cost, if user i releases secret share $f(x_i) \overset{\mathrm{k}}{\leftarrow} x_{(i,1)}$, then her computational cost would increase because two secrets need to be shared for the Tracing algorithm. Specifically, user i runs Shamir's secret sharing scheme on her secrets $(x_{(i,1)}, \mathbf{x}_i \cdot x_{(i,1)})$, then user j verifies them based on two sets of commitments and pseudonym id_i. In contrast, we let user i run Shamir's secret sharing scheme once by taking the secret $x_{(i,1)}$ as input, and transmit a secret share in the form of $g_2^{f(x_i)}$ which can be replaced by a group based on ECC. In particular, user i publishes (or transmits) one set of commitments only, which in turns prove the optimal communication cost as we claimed in this work. Then user j verifies it based on one set of commitments and the proof of equality of discrete logarithms. Furthermore, the expensive operation such as pairing is performed in the Tracing algorithm, so the computational cost when running the Handshake algorithm is not affected by the pairing operations, which makes it efficient and suitable for devices having limited amount of computational resources.

Performance Analysis. The experiment was running on a desktop computer (Inter Core i7 at 2.9Ghz, 6GB RAM). We use a hash function SHA-256 (with 256-bit output size). We also use the Java PBC library [34], and the security level is 160-bit (i.e., key size in the ECC setting). The cryptographic primitives involved in the kLSH protocol include blind digital signature [10] (BS), HMQV protocol [24], PoK and secret sharing scheme [35] (SSS). During the register stage, the signing procedure of BS takes 42ms, while the verification takes 10ms to complete. In the handshake stage, each user takes 10ms to complete HMQV, 52ms to complete PoK (the prover takes 25ms for generating the proof while the verifier takes 27ms to verify the proof) and 1.6ms to complete SSS (the dealer takes 1ms to distribute the secret while the reconstruct algorithm takes 0.6 ms to complete, and we use a $(k, n) = (7, 10)$ SSS as an example). Therefore, we estimate that the total running time of Handshake algorithm by each user takes approximately 63ms.

6. CONCLUSION

In this work, we have proposed a secure and private (yet efficient) construction of linkable secret handshake based on a blind signature scheme [10] and proof of equality of discrete logarithms [18]. The proposed construction achieves anonymity and untraceability against insiders (including the group authority) within a k-time handshakes, and linkable affiliation-hiding against outsiders. In addition, the proposed construction supports an optimal communication cost between handshake users. As for the future work, we would like to design: 1) a new construction for linkable secret handshake in a multi-group setting which is supporting an efficient group discovery [28, 8]; and 2) a generic framework for linkable secret handshakes that can trace any dishonest users.

ACKNOWLEDGEMENTS

The work is supported by the Singapore National Research Foundation under NCR Award Number NRF2014NCR-NCR001-012. It is also supported by AXA Research Fund and National Science Foundation of China under Grant No. 61872264.

REFERENCES

[1] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *2003 Symposium on Security and Privacy, Berkeley, CA, USA, 11-14 May*, pages 180–196. IEEE, 2003.

[2] Shouhuai Xu and Moti Yung. K-anonymous secret handshakes with reusable credentials. In *Proceedings of the 11th ACM conference on Computer and communications security, Washington DC, USA, 25-29 October*, pages 158–167. ACM, New York, NY, USA, 2004.

[3] Claude Castelluccia, Stanisław Jarecki, and Gene Tsudik. Secret handshakes from ca-oblivious encryption. In *International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, 5-9 December*, pages 293–307. Springer, Berlin, Heidelberg, 2004.

[4] Stanisław Jarecki, Jihye Kim, and Gene Tsudik. Group secret handshakes or affiliation-hiding authenticated group key agreement. In *Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 5-9 February*, pages 287–308. Springer, Berlin, Heidelberg, 2007.

[5] Stanisław Jarecki, Jihye Kim, and Gene Tsudik. Beyond secret handshakes: Affiliation-hiding authenticated key exchange. In *Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 8-11 April*, pages 352–369. Springer, Berlin, Heidelberg, 2008.

[6] Mark Manulis, Bertram Poettering, and Gene Tsudik. Affiliation-hiding key exchange with untrusted group authorities. In *International Conference on Applied Cryptography and Network Security, Beijing, China, 22-25 June*, pages 402–419. Springer, Berlin, Heidelberg, 2010.

[7] Mark Manulis, Bertram Poettering, and Gene Tsudik. Taming big brother ambitions: More privacy for secret handshakes. In *International Symposium on Privacy Enhancing Technologies Symposium, Berlin, Germany, 21-23 July*, pages 149–165. Springer, Berlin, Heidelberg, 2010.

[8] Mark Manulis and Bertram Poettering. Affiliation-hiding authentication with minimal bandwidth consumption. In *IFIP International Workshop on Information Security Theory and Practices, Heraklion, Crete, Greece, 1-3 June*, pages 85–99. Springer, Berlin, Heidelberg, 2011.

[9] Isamu Teranishi, Jun Furukawa, and Kazue Sako. K-times anonymous authentication. In *International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, 5-9 December*, pages 308–322. Springer, Berlin, Heidelberg, 2004.

[10] Stefan Brands. Untraceable off-line cash in wallet with observers. In *Annual international cryptology conference, Santa Barbara, California, USA, 22?6 August*, pages 302–318. Springer, Berlin, Heidelberg, 1993.

[11] Yannan Li, Guomin Yang, Willy Susilo, Yong Yu, Man Ho Au, and Dongxi Liu. Traceable monero: Anonymous cryptocurrency with enhanced accountability. *IEEE Transactions on Dependable and Secure Computing*, doi: 10.1109/TDSC.2019.2910058, 2019.

[12] Stanislaw Jarecki and Vitaly Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow

scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2-6 May*, pages 590–608. Springer, Berlin, Heidelberg, 2004.

[13] Xiapu Luo, Edmond WW Chan, Peng Zhou, and Rocky KC Chang. Robust network covert communications based on tcp and enumerative combinatorics. *IEEE Transactions on Dependable and Secure Computing*, 9(6):890–902, 2012.

[14] Xiapu Luo, Junjie Zhang, Roberto Perdisci, and Wenke Lee. On the secrecy of spread-spectrum flow watermarks. In *European Symposium on Research in Computer Security, Athens, Greece, 20-22 September*, pages 232–248. Springer, Berlin, Heidelberg, 2010.

[15] David Chaum and Torben Pryds Pedersen. Transferred cash grows in size. In *Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfred, Hungary, 24-28 May*, pages 390–407. Springer, Berlin, Heidelberg, 1992.

[16] Yutaka Kawai, Kazuki Yoneyama, and Kazuo Ohta. Secret handshake: Strong anonymity definition and construction. In *International Conference on Information Security Practice and Experience, Xi'an, China, 13-15 April*, pages 219–229. Springer, Berlin, Heidelberg, 2009.

[17] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology, Santa Barbara, CA, USA, 20-24 August*, pages 239–252. Springer, New York, NY, 1989.

[18] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Annual International Cryptology Conference, Santa Barbara, California, USA, 15-19 August*, pages 148–164. Springer, Berlin, Heidelberg, 1999.

[19] Mike Burmester and Yvo G Desmedt. Efficient and secure conference-key distribution. In *International Workshop on Security Protocols, United Kingdom, 10?2 April*, pages 119–129. Springer, Berlin, Heidelberg, 1996.

[20] Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In *Annual International Cryptology Conference, Santa Barbara, California, USA, 17-21 August*, pages 110–125. Springer, Berlin, Heidelberg, 2003.

[21] Jonathan Katz and Ji Sun Shin. Modeling insider attacks on group key-exchange protocols. In *Proceedings of the 12th ACM conference on Computer and communications security, Alexandria, VA, USA, 07-11 November*, pages 180–189. ACM, New York, NY, USA, 2005.

[22] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Annual international cryptology conference, Santa Barbara, California, USA, 22?6 August*, pages 232–249. Springer, Berlin, Heidelberg, 1993.

[23] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques, Innsbruck, Austria, 6?0 May*, pages 453–474. Springer, Berlin, Heidelberg, 2001.

[24] Hugo Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. In *Annual International Cryptology Conference, Santa Barbara, California, USA, 14-18 August*, pages 546–566. Springer, Berlin, Heidelberg, 2005.

[25] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *International conference on provable security, Wollongong, Australia, 1-2 November*, pages 1–16. Springer, Berlin, Heidelberg, 2007.

[26] Zheng Yang, Yu Chen, and Song Luo. Two-message key exchange with strong security from ideal lattices. In *Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 16-20 April*, pages 98–115. Springer, Cham, 2018.

[27] Yangguang Tian, Shiwei Zhang, Guomin Yang, Yi Mu, and Yong Yu. Privacy-preserving k-time authenticated secret handshakes. In *Australasian Conference on Information Security and Privacy, Auckland, New Zealand, 3 July*, pages 281–300. Springer, Cham, 2017.

[28] Stanisław Jarecki and Xiaomin Liu. Affiliation-hiding envelope and authentication schemes with efficient support for multiple credentials. In *International Colloquium on Automata, Languages, and Programming, Reykjavik, Iceland, 7-11 July*, pages 715–726. Springer, Berlin, Heidelberg, 2008.

[29] Mark Manulis, Benny Pinkas, and Bertram Poettering. Privacy-preserving group discovery with linear complexity. In *International Conference on Applied Cryptography and Network Security, Beijing, China, 22-25 June*, pages 420–437. Springer, Berlin, Heidelberg, 2010.

[30] Ron D Rothblum. On the circular security of bit-encryption. In *Theory of Cryptography Conference, Tokyo, Japan, 3-6 March*, pages 579–598. Springer, Berlin, Heidelberg, 2013.

[31] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *Annual International Cryptology Conference, Santa Barbara, California, USA, 15-19 August*, pages 273–289, 2004.

[32] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques, Santa Barbara, California, USA, 11-15 August*, pages 186–194. Springer, Berlin, Heidelberg, 1986.

[33] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures. In *Annual International Cryptology Conference, Santa Barbara, California, USA, 17-21 August*, pages 150–164. Springer, Berlin, Heidelberg, 1997.

[34] Angelo De Caro and Vincenzo Iovino. jpbc: Java pairing based cryptography. In *IEEE symposium on computers and communications, Kerkyra, Greece, 28 June-1 July*, pages 850–855. IEEE, 2011.

[35] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.