9-2018

DSH: Deniable secret handshake framework

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

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

Yinghui ZHANG
Singapore Management University, yinghuizhang@smu.edu.sg

Nan LI
University of Newcastle

Guomin YANG
University of Wollongong

See next page for additional authors

DOI: https://doi.org/10.1007/978-3-319-99807-7_21

Citation

TIAN, Yangguang; LI, Yingjiu; ZHANG, Yinghui; LI, Nan; YANG, Guomin; and YU, Yong. DSH: Deniable secret handshake framework. (2018). *Information Security Practice and Experience: 14th International Conference, ISPEC 2018, Tokyo, September 25-27: Proceedings*. 11125, 341-353. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/4158

DSH: Deniable Secret Handshake Framework

Yangguang Tian[1(✉)], Yingjiu Li[1], Yinghui Zhang[1], Nan Li[2], Guomin Yang[3], and Yong Yu[4]

[1] School of Information System, Singapore Management University, Singapore, Singapore
{ygtian,yjli,yinghuizhang}@smu.edu.sg

[2] School of Electrical Engineering and Computing, University of Newcastle, Callaghan, NSW, Australia
nan.li@newcastle.edu.au

[3] School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia
gyang@uow.edu.au

[4] School of Computer Science, Shaanxi Normal University, Xi'an, China
yuyong@snnu.edu.cn

Abstract. Secret handshake is a useful primitive that allows a group of authorized users to establish a shared secret key and authenticate each other anonymously. It naturally provides a certain degree of user privacy and deniability which are also desirable for some private conversations that require secure key establishment. The inherent user privacy enables a private conversation between authorized users without revealing their real identities. While deniability allows authorized users to later deny their participating in conversations. However, deniability of secret handshakes lacks a comprehensive treatment in the literature. In this paper, we investigate the deniability of existing secret handshakes. We propose the first generic framework that converts any secret handshake protocols into fully deniable ones. In particular, we define two formal security models, including session key security and deniability for our proposed framework.

Keywords: Secret handshake · Deniability · Generic framework

1 Introduction

The notion of secret handshake (SH) was firstly introduced by Balfanz et al. [1], and has been extensively studied afterwards. Specifically, it allows authorized users of the same organization to establish a shared secret key in an anonymous manner. To ensure a successful handshake, authorized users need to prove his/her membership to its peer. The only information they need to know is the peer belongs to the same organization.

While anonymity implicitly holds in the secret handshakes setting, other properties are also desirable, such as affiliation-hiding [9], unlinkability [11] and user untraceability [16]. The affiliation-hiding secret handshake is a stronger privacy guarantee than conventional SH, which means non-authorized users cannot identify the membership of authorized users from their handshake sessions or computed session keys. Unlinkable secret handshakes are intuitively more desirable than linkable ones, which means multiple sessions with the same user cannot be linked together. However, it is a challenging task to construct an unlinkable secret handshakes with efficient and practical revocation mechanism (e.g., linear computation complexity in the number of revoked users at [11]). We stress that linkable constructions can easily provide efficient revocation mechanism using pseudonym/certificate revocation list. As for user untraceability, it allows authorized users to remain untraceable during private conversations with respect to the untrusted issuing authority.

Motivation. In addition to aforementioned privacy properties, the secret handshakes can further be explored in the context of "off the record" (OTR). We allow session participants to later deny participating in a private conversation, while anonymous authentication between handshake users is still held. We stress that such deniable communications are particularly important to private conversations in practice. For example, private conversations among handshake users may under mass surveillance by intelligence services, and personal or business communications may be revealed or leaked.

The existing deniable key exchanges allow session participants to plausibly deny their participating in conversations, even if the security of communications is later compromised [5, 21]. In particular, deniability is the most important property in secure messaging protocols such as Off-the-Record Messaging protocol [2]. In other words, deniability is particularly useful to some applications that require secure and private channels without producing cryptographic evidence of communication. Therefore, the main *goal* of this work is to design secret handshakes with maximal range of deniability while the existing privacy guarantees are preserved.

Deniable secret handshake cannot be simply constructed by implementing the existing *fully* deniable authenticated key exchange (AKE) protocols. Full deniability means that anyone can produce protocol transcripts and session keys that look valid to a trusted party (judge). In particular, the fully deniable AKE requires the resulting shared key is merely generated from Diffie-Hellman (DH) exponents [23, 24]. With regard to secret handshakes, authorized users may use their given secret certificate to derive a shared secret key. The fully deniable secret handshakes allow any authorized users (even issuing authority) of the same organization to produce those protocol transcripts and session keys. Therefore, the maximal range of deniable secret handshakes limits to a group of authorized users in the same organization.

In this work, we introduce the notion of deniable secret handshake (DSH in short), and we briefly summarize our main *contributions* as follows: (1) The proposed generic framework can convert any secret handshakes into fully deniable ones; (2) The handshake users can plausibly deny their participating in secure and private conversations, even if the security of their conversations are completely compromised later; (3) We provide two security models for our proposed generic framework, which is used to capture the security and privacy requirements of deniable secret handshakes.

1.1 Related Work

Secret Handshakes. The secret handshake protocol proposed by Balfanz et al. [1] allows any users in the same group to generate a shared value secretly using the reusable (i.e., long-term) certificate approach. Afterwards, Castelluccia et al. [4] constructed a more efficient scheme than [1] under the standard assumptions. However, their schemes [1,4] did not provide the "unlinkability" property. In [22], Xu and Yung provided an unlinkable scheme with a new notion, namely, k-unlinkability. That is, an adversary can infer that a session participant is one out of certain k users in the worst case.

As for achieving full unlinkability, Jarecki et al. [9] proposed two group secret handshake protocols using the Burmester and Desmedt (BD) group key exchange protocol [3]. In particular, their second construction can achieve full unlinkability using unlimited one-time certificate. Meanwhile, another research line was formed in the literature [7,8]. Their solutions can also achieve full unlinkability using long-time certificate. However, their proposed solutions [7,8] lack revocation mechanism, which is necessary and imperative in the secret handshake setting. Recently, Tian et al. [20] proposed a k-time unlinkable secret handshake protocol. Specifically, the proposed solution achieves full unlinkability based on a k-size one-time certificates set, but the central authority (CA, also known as group authority GA) is fully trusted.

To relax the strong assumption on CA, Kawai et al. [14] split the CA into (non-colluding) authorities: one is responsible for registration and issuing certificates, while the other is responsible for tracing users based on protocol transcript. Meanwhile, Manulis et al. [16] considered user's traceability against untrusted CA. Their solution is based on the construction in [10], and uses blinded RSA signature schemes at Add stage for tackling untrusted CA. Similarly, Manulis et al. [17] also proposed a Discrete-Logarithm based secret handshake, and used blinded Schnorr signature to tackle untrusted CA.

Deniable Key Exchange. Deniable authentication was formally introduced by Dwork et al. [6] using the simulation-based paradigm. It requires that the transmitted messages are authenticated, and the simulator's view can be simulated using adversary's knowledge only. Later, Di Raimondo et al. [5] considered deniability of key exchange protocols, and formally presented two definitions: *strong* deniability and *partial* deniability. In particular, they used novel techniques to analyze strong deniability of SKEME and partial deniability of SIGMA respectively. More precisely, they prove strong deniability of SKEME based on the plaintext awareness of the underlying encryption scheme, and prove *partial* deniability of SIGMA based on a special "oracle" since non-repudiable digital signature schemes are *explicitly* used for authentication.

Jiang and Safavi-Naini [12] proposed an efficient key exchange protocol with *full* deniability, such that anyone can prove to a judge that the communication between two participants happened. Their deniable key exchange protocol is formally proven secure in the public random oracle (pRO)[1]. A similar deniable work was proposed by Yao and Zhao [24]. They proposed the first provably secure internet key exchange protocol that provides strong deniability for protocol participants simultaneously. In particular, their deniability analysis relies on the restricted random oracle model[2] and (concurrent) knowledge of exponent assumption (KEA).

Also, *implicitly* AKE protocols [15,19,23] formed another important research direction in the literature. They not only enjoy high performance, but also ensure strong deniability. The *strong* sense of deniability (e.g., [19,21]) means that adversary acts as one of protocol participants (see Definition 2 for detailed comparison between *strong* and *full* deniability).

2 Security Model

In this section, we present the security models for secret handshakes. Note that the secret handshake protocol in this work should (at least) achieve session key security and deniability. The linkable affiliation-hiding (LAH) and untraceability models are directly from [10,16] respectively.

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 pseudonyms 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 $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j$.

[1] pRO is introduced by Pass [18], and it is a weaker assumption compared to random oracle model.

[2] It analogous to Pass's non-programmable random oracle methodology pRO, see detailed comparison in [25].

2.1 System Model

A deniable secret handshake (DSH) protocol consists of the following algorithms:

- Setup: The algorithm takes security parameter λ as input, outputs public parameters params.
- KeyGen: CA takes public parameter param as input, outputs public/secret key pair $(\mathtt{mpk}, \mathtt{msk})$ of group $\mathbb{G}$, and an empty pseudonym revocation list $\mathbb{L}$.
- Add: This is an interactive algorithm between a user and CA. It takes group secret key $\mathtt{msk}$ as input, outputs a pseudonym/certificate pair $(\mathtt{pk}, \mathtt{cert})$, where $\mathtt{pk}$ denotes public pseudonym and $\mathtt{cert}$ denotes secret certificate. The user will become a registered user after interaction with CA. Note that the interaction between CA and users is assumed to be authentic, and CA maintains a group pseudonym list $\mathbb{L}$ by adding public pseudonym $\mathtt{pk}$.
- Revoke: This algorithm is executed by CA of group $\mathbb{G}$ and results in the update of the pseudonym revocation list $\mathbb{L}$.
- $\mathsf{Handshake}$: This is an interactive algorithm among registered users. Each user takes his/her certificate $(\mathtt{pk}, \mathtt{cert})$, $\mathtt{mpk}$ and $\mathbb{L}$ as input, outputs a shared secret key SK if and only if his/her counterparts are non-revoked and registered users.

2.2 Session Key Security

We define the session key security model for DSH protocols, in which each user obtains secret certificate from group CA, and establishes a session key using the given secret certificate. 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 communications channel among all the users.

- Setup: $\mathcal{S}$ first generates group public/secret key pair $(\mathtt{mpk}_j, \mathtt{msk}_j)$ $(j \in [1, \cdots m])$ for m groups in the system. In addition, $\mathcal{S}$ generates the public pseudonym and secret certificate $(\mathtt{pk}_i/\mathtt{cert}_i^j)$ $(i \in [1, \cdots n])$ for n users in a group $\mathbb{G}_j$ by running the corresponding Add algorithm, and returns $\{\mathtt{mpk}_j, \mathtt{pk}_i\}$ to $\mathcal{A}$. $\mathcal{S}$ also tosses a random coin b which will be used later in the game. Let $\mathcal{U}$ denote all the registered and non-revoked users in group $\mathbb{G}_j$.
- $\mathsf{Training}$: $\mathcal{A}$ can make the following queries in arbitrary sequence to $\mathcal{S}$.
  - $\mathsf{establish}$: $\mathcal{A}$ is allowed to register a user U' with public pseudonym $\mathtt{pk} \in \mathbb{G}_j$. 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 $(U_i, \mathbb{G}_j, s, m)$ to simulate a network message for the s-th session of user U_i in group $\mathbb{G}_j$, then $\mathcal{S}$ would simulate the reaction of instance oracle $\Pi_{U_i}^s$ upon receiving message m, and returns to $\mathcal{A}$ the response that $\Pi_{U_i}^s$ would generate; If $\mathcal{A}$ issues send query in the form of $(U_i, \mathbb{G}_j, s, \text{‘}start\text{’})$, then $\mathcal{S}$ creates a new instance oracle $\Pi_{U_i}^s$ and returns to $\mathcal{A}$ the first protocol message.

- ephemeral secret key reveal: If $\mathcal{A}$ issues an ephemeral secret key reveal query to (possibly unaccepted) instance oracle $\Pi_{U_i}^s$, then $\mathcal{S}$ will return all ephemeral secret values contained in $\Pi_{U_i}^s$ 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 the secret certificate $\mathtt{cert}_i^j$ to $\mathcal{A}$.
- group secret key reveal: If $\mathcal{A}$ issues a group secret key reveal query w.r.t. $\mathbb{G}_j$, then $\mathcal{S}$ will return the group secret key $\mathtt{msk}_j$ to $\mathcal{A}$.
- session key reveal: $\mathcal{A}$ can issue reveal query to an accepted instance oracle $\Pi_{U_i}^s$. If the session is accepted, then $\mathcal{S}$ will return the session key to $\mathcal{A}$; Otherwise, a special symbol '$\perp$' is returned to $\mathcal{A}$.
- test: This query can only be made to an accepted and *fresh* (as defined below) session i of a user U ($U \in \mathcal{U}$) in group $\mathbb{G}_j$. Then $\mathcal{S}$ does the following:
  * If the coin $b = 1$, $\mathcal{S}$ returns the real session key to $\mathcal{A}$;
  * Otherwise, a random session key is drawn from the session key space and returns to $\mathcal{A}$.

  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 $\mathcal{S}$ outputs 1; Otherwise, $\mathcal{S}$ outputs 0.

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

- $\mathcal{A}$ issues establish query, where the new user $U' \in \mathsf{pid}_U^i$;
- $\mathcal{A}$ 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 group secret key reveal query to CA is equivalent to the long-term secret key reveal to all registered users in group $\mathbb{G}_j$. We define the advantage of $\mathcal{A}$ in the above game as

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

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

2.3 Deniability

Informally, an adversary aims to present a "proof" to a third party judge, claim that any non-revoked and authorized users of the same organization were participated in a conversation. We formally define the deniability model for secret handshake protocols as follows.

Definition 2. *Let Σ be a secret handshake protocol defined by a key generation algorithm* KeyGen *and interactive machines Σ_I, Σ_R specifying the role of the initiator and responder respectively. We say that* $(\mathsf{KeyGen}, \Sigma_I, \Sigma_R)$ *is a concurrently deniable SH protocol w.r.t the class* Aux[3] *of auxiliary inputs if for any PPT $\mathcal{A}$, for any input of public pseudonym* $\mathsf{pk} = (\mathtt{pk}_1, \cdots, \mathtt{pk}_n)$*, group public keys* $\mathsf{mpk} = (\mathtt{mpk}_1, \cdots, \mathtt{mpk}_n)$ *and any auxiliary input $aux \in Aux$, there exists a simulator $\mathcal{S}$ who runs the same inputs as $\mathcal{A}$*[4]*, aims to produce a simulated view which is indistinguishable from the real view of $\mathcal{A}$. That is, considering the following two probability distributions where* $\mathsf{pk} = (\mathtt{pk}_1, \cdots, \mathtt{pk}_n)$ *is the set of public pseudonym of* honest *users and* $\mathsf{mpk} = (\mathtt{mpk}_1, \cdots, \mathtt{mpk}_n)$ *is the set of* honest *group public keys:*

$$\begin{aligned}
\mathsf{Real}(\lambda, aux) &= [(\mathtt{cert}_i, \mathtt{pk}_i, \mathtt{msk}_j, \mathtt{mpk}_j) \\
&\quad \leftarrow \mathsf{KeyGen}(\lambda, n, m); (aux, \mathsf{pk}, \mathtt{mpk}, \mathsf{View}(aux, \mathsf{pk}, \mathsf{mpk}))] \\
\mathsf{Sim}(\lambda, aux) &= [(\mathtt{cert}_i, \mathtt{pk}_i, \mathtt{msk}_j, \mathtt{mpk}_j) \\
&\quad \leftarrow \mathsf{KeyGen}(\lambda, n, m); (aux, \mathsf{pk}, \mathtt{mpk}, \ \mathcal{S} \ (aux, \mathsf{pk}, \mathsf{mpk}))]
\end{aligned}$$

then for all PPT distinguishers Dist *and all $aux \in$* Aux*, we have*

$$|\Pr_{x \in \mathsf{Real}(\lambda, aux)}[\mathsf{Dist}(x)] = 1 - \Pr_{x \in \mathsf{Sim}(\lambda, aux)}[\mathsf{Dist}(x)] = 1| \leq \epsilon(\lambda) \qquad (2)$$

where ϵ is a negligible function of the security parameter λ. In particular, the actions of distinguisher Dist *after protocol executions are described as follows:*

- *The* Dist *is given the full protocol transcripts and accepted session keys in which $\mathcal{A}$ participated.*
- *The* Dist *is allowed to obtain the secret certificates of all participants and the corresponding master secret key in specific group $\mathbb{G}$.*

Remark. In the sense of *full* deniability, the $\mathsf{View}(aux, \mathsf{pk}, \mathsf{mpk})$ means that $\mathcal{A}$'s view when honest users are faithfully performing secret handshake protocols in group $\mathbb{G}$, while $\mathcal{S}(aux, \mathsf{pk}, \mathsf{mpk})$ means that the simulator $\mathcal{S}$ produces an indistinguishable view to $\mathcal{A}$ *without* honest user's certificates and CA's master secret key. As for the *strong* sense of deniability, where $\mathcal{A}$ acts as one of protocol participants in group $\mathbb{G}$. The $\mathsf{View}(aux, \mathsf{pk}, \mathsf{mpk})$ means that $\mathcal{A}$ maliciously performs

[3] Aux may consist of legal transcripts of protocol runs.

[4] $\mathcal{A}$ is not allowed to reveal honest user's secret certificates $\{\mathsf{cert}_i^j\}$ and group secret keys $\{\mathtt{msk}_j\}$.

secret handshakes with other honest users in a real view. While $\mathcal{S}(aux, \mathsf{pk}, \mathsf{mpk})$ means that the simulator $\mathcal{S}$, who is running on the same inputs as $\mathcal{A}$ (including $\mathcal{A}$'s secret certificates and randomness), simulates an indistinguishable view from a real view to a judge.

3 Our Construction

In this section, we firstly review some complexity assumptions and the building blocks that will be used in our proposed generic framework. We then present our construction afterwards.

3.1 Preliminaries

Secret Handshake (SH). A *forward-secure* secret handshake protocol consists of the following algorithms: $\mathsf{SH} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Add}, \mathsf{Revoke}, \mathsf{Handshake})$.

- Setup: The algorithm takes security parameter λ as input, outputs public parameters params.
- KeyGen: CA takes security parameter param as input, outputs public/secret key pair $(\mathtt{mpk}_l, \mathtt{msk}_l)$ of group $\mathbb{G}_l$, and an empty certificate revocation list Ł_l.
- Add: CA takes secret key $\mathtt{msk}_l$ and user $U_i \in \mathcal{U}$ as input, outputs a certificate cert_i for user U_i.
- Revoke: CA takes user U_i as input, retrieves the corresponding cert_i, and updates the group revocation list Ł_l by adding this certificate cert_i.
- $\mathsf{Handshake}$: This is an interactive algorithm between two users, e.g. U_i and U_j. The input of user U_i (resp., U_j) is a tuple $(\mathsf{cert}_i, \mathtt{mpk}_l, \text{Ł}_l, \mathsf{role}_i)$, where cert_i is U_i's certificate in that group, $\mathtt{mpk}_l$ is the public key of the group with which U_i wants to establish an authenticated connection, Ł_l is U_i's current Ł_l for this group and $\mathsf{role}_i \in \{\mathsf{init}, \mathsf{resp}\}$ (resp., $(\mathsf{cert}_j, \mathtt{mpk}_j, \text{Ł}_j, \mathsf{role}_j)$). Each party either outputs a shared key K or reject otherwise. Note that the interactive $\mathsf{Handshake}$ algorithm generally consists of the following sub-algorithms.
  - $\mathsf{Handshake.Ephemeral}$: User i outputs an ephemeral secret/public key pair $(\mathtt{esk}_i, \mathtt{epk}_i)$;
  - $\mathsf{Handshake.KE}$: Users exchange their ephemeral public keys, i.e., $\mathtt{epk}_i$ and $\mathtt{epk}_j$;
  - $\mathsf{Handshake.KDF}$: User i executes a key derivation function and obtains $K = \mathsf{KDF} : (\mathtt{esk}_i, \mathtt{epk}_j, \mathsf{cert}_i, \mathtt{mpk}_l, \text{Ł}_i, \mathsf{role}_i)$ $(j \neq i)$.

We discover that many existing secret handshake protocols (either RSA based or Discrete-Logarithm based constructions [7–10,16,17]) can support strong deniability, then we have the following Lemma.

Lemma 1. *Forward-secure SH protocols have inherent strong deniability.*

Proof. For simplicity, we assume a SH protocol executed between Alice and Bob where Alice sends ephemeral public key $\mathtt{epk}_A$ to Bob, and vice versa for Bob. We assume that Alice and Bob aim to establish a shared key $K \leftarrow (\mathtt{esk}_A, \mathtt{epk}_B, \mathsf{cert}_A, \mathtt{mpk}, Ł, \mathsf{role})$ in a real view. Note that two simulated values are presented to a distinguisher: the exchanged ephemeral public keys and the resulting shared keys.

The exchanged ephemeral public key pair $(\mathtt{epk}_A, \mathtt{epk}_B)$ derives from either the ephemeral secret key pair $(\mathtt{esk}_A, \mathtt{esk}_B)$ or the function $f(\mathtt{esk}_A, \mathsf{cert}_A)$, $f(\mathtt{esk}_B, \mathsf{cert}_B)$ (f denotes a randomized function, such as hash function or pseudorandom function). Simulator $\mathcal{S}$ (i.e., adversary Bob) is simply choosing a random value $\mathtt{epk}_A$ from ephemeral public space to simulate transmitted ephemeral public key on behalf of Alice, while the judge cannot *statistically* distinguish it since the real/simulated value is uniformly distributed in either ephemeral public key space or the output of function f (except collisions with a negligible probability);

The resulting shared key K, can be easily simulated by simulator $\mathcal{S}$. Specifically, $\mathcal{S}$ (i.e., adversary Bob) randomly chooses ephemeral public key $\mathtt{epk}_B$ and computes $K \leftarrow (\mathtt{epk}_A, \mathtt{esk}_B, \mathsf{cert}_B, \mathtt{mpk}, Ł, \mathsf{role})$, where ephemeral secret key $\mathtt{esk}_B$ and secret certificate cert_B are known to $\mathcal{S}$. Note that the correctness of resulting shared keys require Alice and Bob also exchange their public pseudonyms $(\mathtt{pk}_A, \mathtt{pk}_B)$ which is deriving from respective secret certificates $(\mathsf{cert}_A, \mathsf{cert}_B)$. □

Definition 3 (Generic Concurrent Knowledge Extraction Assumption (GCKEA)). *We define a domain* $\{\mathsf{Dom}_\lambda\}_{\lambda \in \mathbb{N}}$*, where* $\mathbb{N}$ *is the set of natural numbers, and define a set* $\mathcal{D} \overset{R}{\leftarrow} \mathsf{Dom}_\lambda$*. We denote* $p(\lambda), q(\lambda)$ *are two polynomials in the security parameter* λ*, and define a predicate algorithm* $\mathcal{O}_\mathcal{C}$ *w.r.t the random challenge set* $\mathcal{C} = \{C_1, \cdots, C_{p(\lambda)}\}$*. On a query of the form* (X, Y, Z)*, for arbitrary* $(X, Y) \overset{R}{\leftarrow} \mathcal{D}$ *outputs 1 if* $X \overset{R}{\leftarrow} \mathcal{C}$ *and* $Z = \mathsf{PKDF}(X, Y)$[5]*. We define an algorithm* $\mathcal{A}$ *with predicate oracle* $\mathcal{O}_\mathcal{C}$*, denote* $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$*, which takes* $\mathcal{C}$ *as input, outputs a set of triples* $\{(X_1, Y_1, Z_1), \cdots, (X_{q(\lambda)}, Y_{q(\lambda)}, Z_{q(\lambda)})\}$*. We say* $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ *is a GCKEA extractor if, with overwhelming probability,* $\mathcal{A}^{\mathcal{O}_\mathcal{C}}(\mathcal{C})$ *outputs* $\{(X_1, Y_1, Z_1), \cdots, (X_{q(\lambda)}, Y_{q(\lambda)}, Z_{q(\lambda)})\}$ *satisfying* $X_i \in \mathcal{C}$ *and* $Z_i = \mathsf{PKDF}(X_i, Y_i)$ *for all* $i, 1 \leq i \leq q(\lambda)$*.*

We say that the GCKEA holds if for every PPT algorithm $\mathcal{A}$*, there exists another PPT algorithm* $\mathcal{A}'$ *that given the same inputs, random coins, oracle answers, and additionally outputs* y_i *such that* $Y_i \leftarrow G(y_i)$ *for all* $i, 1 \leq i \leq q(\lambda)$*, where* G *denotes an efficient computable function which takes* y_i *as input and outputs* Y_i*.*

Remark. GCKEA is a *generalized* version of Concurrent KEA (CKEA) [24] and Knowledge of Pairing Pre-Image Assumption (KPA) [19]. Specifically, the extracted value y_i by extractor $\mathcal{A}'$ is either exponent w.r.t. CKEA assumption [24] or group element w.r.t. KPA assumption [19]. For example, the concrete

[5] The PKDF algorithm means that a key derivation function in the public-key setting.

CKEA assumption [24] is used to extract the DH exponent for their proposed *interactive* protocol, while running against the *concurrent* man-in-the-middle adversaries.

3.2 Proposed Framework

Our proposed generic framework (GF) consists of the following building blocks.

– A forward-secure secret handshake protocol $\mathsf{SH} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Add}, \mathsf{Revoke}, \mathsf{Handshake})$; Note that SH protocol may have LAH property.
– A blind digital signature scheme $\mathsf{BS} = (\mathsf{KeyGen}, \mathsf{Signer\ and\ User}, \mathsf{Verify})$; Note that if SH has untracebility property, then this building block is removed.
– A public key based key derivation function PKDF;
– A proof of knowledge PoK;
– A collision-resistant hash function $\mathtt{H}$.

Now we present our proposed generic framework below (for simplicity, we use user $\widehat{A}$ and user $\widehat{B}$ in the two-party setting here, and we can extend it to a multi-party setting using BD protocol [3]):

– Setup: This algorithm takes security parameter λ as input, outputs public parameters $\mathsf{params} \leftarrow \mathsf{SH.Setup}$ which are published to all users and groups.
– KeyGen: The group CA runs the $\mathsf{SH.KeyGen}$ algorithm to obtain the group public/secret key pair $(\mathtt{mpk}, \mathtt{msk})$ and an empty pseudonym revocation list Ł.
– Add: The group CA and user $\widehat{A}$ run the $\mathsf{BS.Signer\ and\ User}(\mathtt{msk})$ interactive algorithm[6] to obtain a pseudonym/certificate pair $(\mathtt{pk}_a, \mathtt{cert}_a)$ of user $\widehat{A}$. Note that user $\widehat{A}$ takes $\mathtt{pk}_a$ as public pseudonym.
– Revoke: The group CA runs the $\mathsf{SH.Revoke}(\mathtt{pk}_a)$ algorithm to update the group pseudonym revocation list Ł. Note that public pseudonym $\mathtt{pk}_a$ is added to revocation list Ł.
– $\mathsf{Handshake}$:
  • User $\widehat{A}$ runs the $\mathsf{SH.Handshake.Ephemeral}$ algorithm to obtain ephemeral secret/public key pair $(\mathtt{esk}_a, \mathtt{epk}_a)$ and sends $(\mathtt{epk}_a, \mathtt{pk}_a)$ to user $\widehat{B}$;
  • Upon receiving $(\mathtt{epk}_a, \mathtt{pk}_a)$ from user $\widehat{A}$, user $\widehat{B}$ performs the following steps.
    1. Run the $\mathsf{SH.Handshake.Ephemeral}$ algorithm to obtain ephemeral secret and public key pair $(\mathtt{esk}_b, \mathtt{epk}_b)$;
    2. Compute the proof of knowledge $\mathsf{PoK}\{(\mathtt{esk}_b) : \mathtt{H}(\mathsf{PKDF}(\mathtt{epk}_b, \mathtt{epk}_a))\}$;
    3. Send $(\mathtt{epk}_b, \mathtt{pk}_b, \mathsf{PoK}(\mathtt{esk}_b))$ to user $\widehat{A}$.
  • Upon receiving $(\mathtt{epk}_b, \mathtt{pk}_b, \mathsf{PoK}(\mathtt{esk}_b))$ from user $\widehat{B}$, user $\widehat{A}$ computes the proof of knowledge (i.e., non-malleable zero-knowledge) $\mathsf{PoK}\{(\mathtt{esk}_a, \mathtt{cert}_a) : \mathtt{H}(\mathsf{PKDF}(\mathtt{epk}_a, \mathtt{epk}_b)||\mathsf{PKDF}(\mathtt{pk}_a, \mathtt{epk}_b))\}$ and sends it to user $\widehat{B}$. Meanwhile, $\widehat{A}$ computes the final session key $SK_a = \mathtt{H}(K_a||\mathsf{sid})$, where $K_a = \mathsf{SH.Handshake.KDF}(\mathtt{esk}_a, \mathtt{epk}_b, \mathtt{cert}_a, \mathtt{mpk}, Ł, \mathsf{init})$ and the session identifier is $\mathsf{sid} = (\mathtt{epk}_a||\mathtt{epk}_b)$.

[6] Refer to [13] for detailed $\mathsf{Signer\ and\ User}$ algorithm of BS scheme.

- Upon receiving $\mathsf{PoK}(\mathtt{esk}_a, \mathtt{cert}_a)$ from user $\widehat{A}$, user $\widehat{B}$ computes the proof of knowledge $\mathsf{PoK}\{(\mathtt{esk}_b, \mathtt{cert}_b) : \mathtt{H}(\mathsf{PKDF}(\mathtt{epk}_b, \mathtt{epk}_a)||\mathsf{PKDF}(\mathtt{pk}_b, \mathtt{epk}_a))\}$ and sends it to user $\widehat{A}$. Meanwhile, $\widehat{B}$ computes the final session key $SK_b = \mathtt{H}(K_b||\mathsf{sid})$, where $K_b = \mathsf{SH.Handshake.KDF}(\mathtt{esk}_b, \mathtt{epk}_a, \mathtt{cert}_b, \mathtt{mpk}, \mathrm{Ł}, \mathsf{resp})$. Note that the equation $K_a = K_b$ holds due to the correctness of $\mathsf{SH.Handshake}$ algorithm.

3.3 Security Analysis

Theorem 1. *The proposed generic framework achieves* session key security (Definition 1) *in the random oracle model if the underlying* SH *is session key secure.*

Proof Sketch. Due to page limitation, the detailed security proof and the subsequent proof are deferred to the full version of this work. We here only present the proof sketch. We define a sequence of games G_i, $i = 0, \cdots, 4$ for session key security and analyze the advantage of the adversary in game G_i.

The first game G_0 is original game for session key security. The second game G_1 is used to capture replay attacks, such that no PPT adversary can find the collision of hash function $\mathtt{H}$ if users follow the framework execution honestly. In game G_2, we assume that the adversary must choose the specific session for test query, which is specified by the simulator. In game G_3, we assume that if adversary can distinguish game G_2 and G_3 (the real SH session key is replaced by a random value), then we can built an attacker to break the session key security of underlying SH protocol. In the last game G_4, the final session key in the test session is replaced by a random value. No PPT adversary can distinguish this change since we model $\mathtt{H}$ as a random oracle. Therefore, the advantage of adversary in this game is zero.

Theorem 2. *The proposed generic framework achieves* full deniability *in the sense of* Definition *2.*

The full deniability proof is similar to the proof of deniability described in [24].

4 Conclusion

In this paper, we proposed a generic framework for deniable secret handshake protocols. We defined the formal security models for session key security and deniability of secret handshake protocols, and proved the security of the proposed generic framework is secure under standard assumptions. We leave the construction of an efficient and fully deniable instantiation as our future work.

Acknowledgements. This work is supported by the Singapore National Research Foundation under NCR Award Number NRF2014NCR-NCR001-012, the NSFC Research Fund for International Young Scientists (61750110528), National Cryptography Development Fund during the 13th Five-year Plan Period (MMJJ20170216), the Fundamental Research Funds for the Central Universities (GK201702004).

References

1. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.: Secret handshakes from pairing-based key agreements. In: IEEE (S&P 2003), pp. 180–196 (2003)
2. Borisov, N., Goldberg, I., Brewer, E.: Off-the-record communication, or, why not to use PGP. In: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, pp. 77–84 (2004)
3. 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). https://doi.org/10.1007/3-540-62494-5_12
4. 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). https://doi.org/10.1007/978-3-540-30539-2_21
5. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Deniable authentication and key exchange. In: CCS, pp. 400–409. ACM (2006)
6. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. J. ACM (JACM) **51**(6), 851–898 (2004)
7. Gu, J., Xue, Z.: An improved efficient secret handshakes scheme with unlinkability. IEEE Commun. Lett. **15**(2), 259–261 (2011)
8. Huang, H., Cao, Z.: A novel and efficient unlinkable secret handshakes scheme. IEEE Commun. Lett. **13**(5), 363–365 (2009)
9. 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). https://doi.org/10.1007/11967668_19
10. Jarecki, S., Kim, J., Tsudik, G.: Beyond secret handshakes: affiliation-hiding authenticated key exchange. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 352–369. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79263-5_23
11. Jarecki, S., Liu, X.: Private mutual authentication and conditional oblivious transfer. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 90–107. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_6
12. Jiang, S., Safavi-Naini, R.: An efficient deniable key exchange protocol (extended abstract). In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 47–52. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85230-8_4
13. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052233
14. Kawai, Y., Yoneyama, K., Ohta, K.: Secret handshake: strong anonymity definition and construction. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 219–229. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00843-6_20
15. 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). https://doi.org/10.1007/11535218_33
16. Manulis, M., Poettering, B., Tsudik, G.: Affiliation-hiding key exchange with untrusted group authorities. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 402–419. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13708-2_24

17. Manulis, M., Poettering, B., Tsudik, G.: Taming big brother ambitions: more privacy for secret handshakes. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 149–165. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14527-8_9
18. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_19
19. Schäge, S.: TOPAS: 2-pass key exchange with full perfect forward secrecy and optimal communication complexity. In: CCS, pp. 1224–1235. ACM (2015)
20. Tian, Y., Zhang, S., Yang, G., Mu, Y., Yu, Y.: Privacy-preserving k-time authenticated secret handshakes. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10343, pp. 281–300. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59870-3_16
21. Unger, N., Goldberg, I.: Deniable key exchanges for secure messaging. In: CCS, pp. 1211–1223. ACM (2015)
22. Xu, S., Yung, M.: K-anonymous secret handshakes with reusable credentials. In: CCS 2004, pp. 158–167. ACM (2004)
23. Yao, A.C.-C., Zhao, Y.: OAKE: a new family of implicitly authenticated Diffie-Hellman protocols. In: CCS, pp. 1113–1128. ACM (2013)
24. Yao, A.C.-C., Zhao, Y.: Privacy-preserving authenticated key-exchange over internet. IEEE TIFS **9**(1), 125–140 (2014)
25. Yung, M., Zhao, Y.: Interactive zero-knowledge with restricted random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 21–40. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_2