# Secret Handshake: Strong Anonymity Definition and Construction

Yutaka Kawai[1], Kazuki Yoneyama[1⋆], and Kazuo Ohta[1]

The University of Electro-Communications 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan
{ kawai,yoneyama,ota } @ice.uec.ac.jp

**Abstract.** Secret handshake allows two members in the same group to authenticate each other secretly. In previous works of secret handshake schemes, two types of anonymities against the group authority (GA) of a group $G$ are discussed: 1)*Even* GA cannot identify members, namely nobody can identify them (No-Traceability), 2)*Only* GA can identify members (Traceability). In this paper, first the necessity of tracing of the identification is shown. Second, we classify abilities of GA into the ability of identifying players and that of issuing the certificate to members. We introduce two anonymities *Co-Traceability* and *Strong Detector Resistance*. When a more strict anonymity is required ever for GA, the case 2) is unfavorable for members. Then, we introduce *Co-Traceability* where even if $\mathcal{A}$ has GA's ability of identifying members or issuing the certificate, $\mathcal{A}$ cannot trace members identification. However, if a scheme satisfies Co-Traceability, GA may be able to judge whether handshake players belong to the own group. Then, we introduce *Strong Detector Resistance* where even if an adversary $\mathcal{A}$ has GA's ability of identifying members, $\mathcal{A}$ cannot make judgments whether a handshaking player belongs to $G$. Additionally, we propose a secret handshake scheme which satisfies previous security requirements and our proposed anonymity requirements by using group signature scheme with message recovery.

**keywords: secret handshake, anonymity, traceability, privacy**

## 1 Introduction

### 1.1 Background

A secret handshake scheme (SHS), introduced by Balfanz et al. [3], allows two members of the same group to authenticate each other secretly, in the sense that each party reveals his affiliation information to the other only if the other party belongs to the same group. For example scenario: a CIA agent Alice wants to authenticate to Bob, but only if Bob is also

⋆ Supported by JSPS Research Fellowships for Young Scientists.

a CIA agent. In addition, if Bob is not a CIA agent, Alice does not want to reveal her affiliation information, whether Alice is a CIA agent or not, for Bob.

Balfanz, et al. [3] constructed a 2-party SHS by adapting the key agreement protocol of Sakai, et al. [10]. Its security rests on the hardness of the Bilinear Diffie Hellman (BDH) problem. Subsequently, Castelluccia, et al. [6] developed a more efficient 2-party SHS through the use of so-called CA-oblivious encryption. It is secure under Computational Diffie Hellman (CDH) assumption.

Both previous works satisfy the basic security properties of secret handshake system; *correctness, impersonator resistance,* and *detector resistance*. Recently, the *unlinkability* is added to the basic security requirement. Unlinkability means that two occurrences of handshaking by the same party cannot be linked with each other by anyone. In [3, 6], however, the member sends one's ID information in a handshake protocol. Thus, the construction of [3, 6] does not satisfy *unlinkability* unless members use one-time certificate (i.e. member change IDs whenever members execute handshake protocol.).

Xu and Yung [11] constructed a secret handshake scheme which achieves weaker unlinkability "$k$-anonymity" with reusable certificate, instead of one-time certificates. Members can reuse their certificate because they always authenticate as someone among $k$ users.

The work of [1] presented the first construction of SHS with unlinkability using reusable certificate in the standard model. The scheme in [1] allows each participant to specify the role and group of the other party and thus add flexibility to the authentication rules. Moreover, they achieve attribute-based secret handshakes using fuzzy identity-based-encryption.

### 1.2 Anonymity against Group Authority

*Motivation:* Let us consider the case in which whistle-blowing the system of company is troublesome. In this case the GA is a manager of company and the members are employees of the company and the authorized persons of the company (e.g. lawyer). When the employee uses this system, he executes a secret handshake with the lawyer in order to pass on the whistle-blowing. The employee wants to tell to only his company's lawyer about exposure. The lawyer should only know the fact that the employee belongs to his company. In above scenario, a secret handshake scheme is very useful.

However, if previous secret handshake schemes are used as above whistle-blowing system, there is a problem. In previous secret handshake

schemes, GA can identify handshake players (player's names, IDs, etc). In this scenario, the employees, who want to exposure, want to execute a secret handshake to remain anonymous. If an employee is not guaranteed anonymity, he will not blow the whistle in the fear of the dismissal etc. However, for example in [6], a handshake player must send own ID to a handshake partner. IDs are registered to GA, when a member joins to the group. So, GA can identify handshake players by referring ID list and watching the transcript of handshake.

From the above reason, the previous schemes are not applicable to this case. So in this paper, we will define new security requirements about anonymities against GA. We introduce two new anonymities *Co-Traceability* and *Strong Detector Resistance*. *Strong Detector Resistance* means that even if an adversary $\mathcal{A}$, who does not belong to $G$, has GA's ability of identifying members, $\mathcal{A}$ cannot make judgments whether a handshaking player belongs to $G$. *Co-Traceability* means that GA alone cannot reveal members identification.

Co-Traceability can never be satisfied against GA with all of abilities, since if $U$ belongs to $G$, $U$ might be revealed identification executing a handshake protocol with a dummy member $D$ whom GA creates using an ability to issue a user. Then, GA can identify $U$ by using the information of $D$ and the ability of identifying members. Also, Strong Detector Resistance can never be satisfied against GA with all of the abilities. The reason is that if $U$ belongs to $G$, $U$ outputs accept certainly executing a handshake protocol with a dummy member $D$ whom GA creates using an ability to issue a user.

Therefore, in order to discuss the anonymity against GA, we split the role of GA into the two; issue authority and trace authority. The issue authority (IA) issues the certificate to users. The trace authority (TA) has an ability to identify a member. Co-Traceability is defined against an adversary who has an ability of either TA or IA. Strong Detector Resistance is defined against an adversary who has an ability of TA.
*Co-Traceability:* If a secret handshake scheme adopt whistle-blowing, handshake players do not want to reveal own identification against even GA. However, tracing of handshake players is useful for the handshake player in case the evidence of handshaking is required.

So, we introduce new security requirement Co-Traceability. Intuitively, Co-Traceability means that TA alone cannot identify handshake players, but TA can identify players by cooperating with another. In our proposed secret handshake scheme, another is a handshake player. If Alice and Bob

execute a secret handshake, TA alone cannot identify Alice and Bob, but if TA cooperates with Alice, TA could identify Bob.

At the same time, in this paper, we will define this algorithm "SHS.Co-Trace". This new algorithm is useful in the situation which a handshake player wants to know the other handshake partner when the output of the protocol is *accept*. In previous secret handshake schemes, identifications of handshake players always reveal against everyone. By using SHS.Co-Trace, a handshake player can execute a handshake protocol hiding own identification and can know the identification of the other handshake player by revealing own identification to TA.

*Strong Detector Resistance:* In previous works, in Secret Handshake system, information about the group of the player must not be leaked, if players does not belong to the same group. There are the cases that a player wants to hide the information of his own group from TA as well as members and users. If TA can know whether a handshake player belongs to his own group $G$ or not, GA might examine the frequency that a member of group $G$ executes a handshake protocol. This consideration leads to the new security property, *Strong Detector Resistance*.

In this paper, we discuss the anonymity and groups which members belong to in the case TA or IA is corrupted. Here, "TA (IA) is corrupted" means that an adversary $\mathcal{A}$ can get a secret key of TA (IA). The levels of trust for each authority for each requirement are summarized in Table1.2.

**Table 1.** Levels of trust in authorities for each of security requirements

| Previous Security | Issuer Authority | Trace Authority |
|---|---|---|
| Impersonator Resistance | Uncorrupted | Uncorrupted |
| Detector Resistance | Uncorrupted | Uncorrupted |
| Unlinkability | Uncorrupted | Uncorrupted |
| **Proposed Security** | **Issuer Authority** | **Trace Authority** |
| Co-Traceability (against TA) | Uncorrupted | Corrupted |
| Co-Traceability (against IA) | Corrupted | Uncorrupted |
| Strong Detector Resistance | Uncorrupted | Corrupted |

In order to realize *Strong Detector Resistance* and *Co-Traceability*, we introduce a new algorithm SHS.Co-Trace. SHS.Co-Trace, given a group public key, a group authority secret key, and "an internal information of a player", outputs ID of the other player. Concretely, we realize SHS.Co-

Trace and *Co-Traceability* with a secret key to execute SHS.Co-Trace separately by GA and a handshake player.

## 2 Definition of Secret Handshake

### 2.1 Entity

In SHS, there exist three entities in the group $G$ as follows.

**User:** the entity which does not belong to the group. $A \notin G$ means that the user $A$ does not belong to the group $G$.

**Member:** the entity which is made belong to the group, by the Group Authority. $A \in G$ means that the member $A$ belongs to the group $G$.

**GA (Group Authority):** the manager of a group. GA is responsible for adding users into the group he manages. GA maintains a list $\mathcal{L}$, which includes certificates and registration data of all members.

### 2.2 The Model of Secret Handshake

A secret handshake scheme SHS consists of the following five algorithms:

SHS.Setup**:** generates the public parameters $param$ which is common to all groups.

SHS.CreateGroup**:** generates a key pair $gpk$ (group public key) and $gsk$ (private key for GA), using $param$. SHS.CreateGroup is run by GA.

SHS.AddMember**:** is executed between a player $U$ and a GA of some group. Inputs of SHS.AddMember are $gsk$, $param$, and $gpk$, and outputs are a membership certificate ($cert_U$) and a secret key ($sk_U$)

SHS.Handshake**:** is the authentication protocol executed between $U$ and $V$, based on the public input $param$. The group public keys ($gpk_U$, $gpk_V$) and certificates (($cert_U, sk_U$), ($cert_V, sk_V$)) of $U$ and $V$ are input to handshake protocol. The result of the algorithm is either reject or accept. $U \overset{Handshake}{\longleftrightarrow} V$ means the above situation.

SHS.Trace**:** , given $gpk$, $gsk$ and a transcript $\mathcal{T}$ of the handshaking of $U$ and $V$, outputs the member $U$ (or $V$).

### 2.3 Security Properties of SHS scheme

A secret handshake protocol must satisfy the following *basic* security properties: **Correctness, Impersonator Resistance, Detector Resistance, Unlinkability**[6].

– $\mathcal{O}_{\mathcal{CG}}(\cdot)$ :This oracle, given an identity of GA, outputs $gpk, gsk$, namely SHS.CreateGroup.
– $\mathcal{O}_{\mathcal{AM}}(\cdot,\cdot)$ : This oracle, given an identity of group $G$ and a user $U$, executes SHS.AddMember.
– $\mathcal{O}_{\mathcal{HS}}(\cdot)$ :This oracle, given a member $U$, executes SHS.Handshake as honest members $U$ between $\mathcal{A}$. The honest members executes according to the protocol.
– $\mathcal{O}_{\mathcal{CO}}(\cdot)$ :This oracle, given a member $U$, outputs $U$'s current internal information including secrets.

**Correctness:** If honest members $U$,$V$ of the same group run handshake protocol, then both players always output "*accept*".

**Impersonator Resistance** (IR)**:** IR demands that the adversary $\mathcal{A}$, who does not belong to a group $G$, is not able to authenticate an honest who belongs to $G$.

Formally, we say that SHS guarantees IR if the following function $\mathsf{Adv}^{\mathsf{IR}}_{\mathcal{A}}(k)$ is negligible for any polynomially-bounded adversary $\mathcal{A}$.

$$\mathsf{Adv}^{\mathsf{IR}}_{\mathcal{A}}(k) = \Pr[\mathbf{Exp}^{\mathsf{IR}}_{\mathcal{A}}(k) = 1]$$

Experiment $\mathbf{Exp}^{\mathsf{IR}}_{\mathcal{A}}(k)$
$param \leftarrow \mathsf{SHS.Setup}(k)$
$(gpk, gsk) \leftarrow \mathsf{SHS.CreateGroup}(param)$;
$(cert_U, sk_U) \leftarrow \mathsf{SHS.AddMember}(param, gpk, gsk, U)$
$(cert_V, sk_V) \leftarrow \mathsf{SHS.AddMember}(param, gpk, gsk, V)$
$\mathsf{honest}(param, gpk, sk_U, cert_U)$
$\overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk, cert_V)$
If honest outputs *accept*, outputs 1.
Otherwise outputs 0.

**Detector Resistance** (DR)**:** DR demands that the adversary $\mathcal{A}$, who does not belong to a group $G$, is not able to distinguish whether some honest is a member of some group $G$.

Formally, we say that SHS guarantees DR if the following function $\mathsf{Adv}^{\mathsf{DR}}_{\mathcal{A}}(k)$ is negligible for any polynomially-bounded adversary $\mathcal{A}$.

$$\mathsf{Adv}^{\mathsf{DR}}_{\mathcal{A}}(k) = \left|\Pr[\mathbf{Exp}^{\mathsf{DR}}_{\mathcal{A}}(1,k) = 1] - \Pr[\mathbf{Exp}^{\mathsf{DR}}_{\mathcal{A}}(0,k) = 1]\right|$$

Experiment $\mathbf{Exp}^{\mathsf{DR}}_{\mathcal{A}}(b,k)$
$param \leftarrow \mathsf{SHS.Setup}(k)$
$(gpk, gsk) \leftarrow \mathsf{SHS.CreateGroup}(param)$;
$(sk_U, cert_U) \leftarrow \mathsf{SHS.AddMember}(param, gpk, gsk, U)$
$\mathbf{player}_b := \mathsf{honest}(param, gpk, sk_U, cert_U)$;
$\mathbf{player}_{1-b} := \mathsf{SIM}(param)$
$\mathbf{player}_0 \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk)$
$\mathbf{player}_1 \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(param, gpk)$
Return $b'$

**Unlinkability** (Unlink)**:** Unlink demands that the adversary $\mathcal{A}$, who does not belong to a group $G$, is not able to decried whether two executions of the handshake protocol were performed by the same party or not, even if both of them were successful.

Formally, we say that SHS guarantees Unlink if the following function $\mathsf{Adv}^{\mathsf{Unlink}}_{\mathcal{A}}(k)$ is negligible for any polynomially-bounded adversary $\mathcal{A}$.

$$\mathsf{Adv}^{\mathsf{Unlink}}_{\mathcal{A}}(k) = \left|\Pr[\mathbf{Exp}^{\mathsf{Unlink}}_{\mathcal{A}}(1,k)=1] - \Pr[\mathbf{Exp}^{\mathsf{Unlink}}_{\mathcal{A}}(0,k)=1]\right|$$

Experiment $\mathbf{Exp}^{\mathsf{Unlink}}_{\mathcal{A}}(b,k)$
$param \leftarrow \mathsf{SHS.Setup}(k)$
$(gpk, gsk) \leftarrow \mathsf{SHS.CreateGruop}(param)$;
If $b = 0$ then $\mathbf{player}_U = \mathbf{player}_V$;
Otherwise $\mathbf{player}_U \neq \mathbf{player}_V$;
$\mathbf{player}_U \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk)$
$\mathbf{player}_V \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(param, gpk)$
Return $b'$

## 3 Anonymity against Group Authority

In previous works of secret handshake schemes, anonymities against GA have not been discussed. In this section we will define two new anonymity requirements against GA, *Co-Traceability* and *Strong Detector Resistance* of secret handshake schemes.

### 3.1 Issue authority and Trace authority

In this subsection, let us classify the GA from the view point of abilities. GA has the two abilities. First ability is to issue the certificate to users. Second ability is to trace handshake players. We call the authority with the former ability *Issue authority* (IA). and the authority with the latter ability *Trace authority* (TA). IA has a public and secret key $(ipk, isk)$ and TA has a public and secret key $(tpk, tsk)$ [1].

### 3.2 Co-Traceability

In previous works of secret handshake schemes, two types of anonymities against the GA are discussed from the view point of revealing the identification of handshake players: 1)*Even* GA cannot identify players, namely nobody can identify them, 2)*Only* GA can identify players.

Tracing of handshake players is useful for the handshake player in case the evidence of handshaking is required. However when the anonymity is preferred as the case of prosecution from inside, a more strict anonymity is required ever for GA.

In the way of $\mathsf{SHS.Trace}$, members except TA cannot join $\mathsf{SHS.Trace}$ process. From the view point of power of TA, this situation could be troublesome for members. Then, we introduce a new security requirement *Co-Traceability*. Intuitively, *Co-traceability* means that TA alone cannot identify handshake players.

Co-Traceability can never be satisfied against GA with all abilities, since if $U$ belongs to $G$, $U$ might be revealed identification executing a handshake protocol with a dummy member $D$ who $GA$ creates using an ability to issue a user. So, two different types of adversarys, $\mathcal{A}_T$ and $\mathcal{A}_I$, should be concerned. The difference between $\mathcal{A}_T$ and $\mathcal{A}_I$ is the input. The input of $\mathcal{A}_T$ is $param$, $gpk$ and $tsk$. The input of $\mathcal{A}_I$ is $param$, $gpk$ and $isk$. The formal definition of Co-traceability is as follow:

**Definition 1 (Co-Traceability).** Co-Traceability ($\mathsf{Co\text{-}Trace}$) means that an adversary $\mathcal{A}$ who has GA's secret key ($isk$ or $tsk$) cannot identify the player which execute $\mathsf{SHS.Handshake}$ even though $\mathcal{A}$ is not a member of $G$. Formally, we say that $\mathsf{SHS}$ guarantees $\mathsf{Co\text{-}Trace}$ if the following function $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k)$is negligible. Let a member which executes according to the protocol be $\mathsf{honest}$. We consider a PPT adversary $\mathcal{A}$ that can access to the oracles $\mathcal{O}$.

[1] In previous definition, $gpk = (ipk, tpk)$, $gsk = (isk, tsk)$.

Formally, we say that SHS guarantees Co-Trace if the following function $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k)$ is negligible for any PPT adversary $\mathcal{A} = (\mathcal{A}_T, \mathcal{A}_I)$.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k) = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k) = 1]$$

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k)$
$param \leftarrow \mathsf{SHS.Setup}(k)$
$(ipk, tpk, isk, tsk) \leftarrow \mathsf{SHS.CreateGroup}(param)$;
$(cert_U, sk_U) \leftarrow \mathsf{SHS.AddMember}(param, ipk, tpk, isk, U)$
$\mathsf{honest}(param, gpk, sk_U, cert_U) \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}$
$U' \leftarrow \mathcal{A}^{\mathcal{O}}$
If $U = U'$, outputs 1. Otherwise outputs 0.

### 3.3 Strong Detector Resistance

If a secret handshake scheme satisfies Co-Traceability, TA alone cannot identify handshake players. However, GA may be able to identify whether handshake players belong to the own group. Then, GA can know the frequency that members of own group execute handshake. In order to achieve high anonymity against GA, even the frequency of handshake should be hidden for GA.

We then introduce the concept of *Strong Detector Resistance* in order to cope with this requirement. Intuitively, Strong Detector Resistance means that even if an adversary $\mathcal{A} \notin G$ has all the ability of revealing the identity of members, $\mathcal{A}$ cannot make judgment whether a handshaking player belongs to $G$ or not.

Strong Detector Resistance never satisfies against IA, since if $U$ belongs to $G$, $U$ outputs *accept* executing a handshake protocol with a dummy member who GA creates using an ability to issue a user. The formal definition of Strong Detector Resistance is as follow:

**Definition 2 (Strong Detector Resistance).** Strong Detector Resistance (SDR) demand that an adversary $\mathcal{A}$ who have "the all ability of revealing the identity of members" (e.g. trace key, member-list $\mathcal{L}$) cannot distinguish whether some honest player, who is a member of group $G$, even though $\mathcal{A}$ is not a member of $G$. SDR is easier security goal than SDR for an adversary $\mathcal{A}$.

Formally, we say that SHS guarantees SDR if the following function $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SDR}}(k)$ is negligible for any polynomially-bounded adversary $\mathcal{A}$.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SDR}}(k) = \left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{SDR}}(1,k) = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{SDR}}(0,k) = 1]\right|$$

SDR has a close relationship to the SHS.Trace algorithm. If GA can reveal an identify ID of a member [6, 3], an adversary can break SDR by executing SHS.Trace. On the other hand, if SHS does not have algorithm revealing an identify [1], SDR is equivalent to DR.

Experiment $\mathbf{Exp}^{\mathsf{SDR}}_{\mathcal{A}}(b,k)$
$param \leftarrow \mathsf{SHS.Setup}(k)$
$(ipk, tpk, isk, tsk) \leftarrow \mathsf{SHS.CreateGroup}(param)$;
$(cert_U, sk_U) \leftarrow \mathsf{SHS.AddMember}(param, ipk, tpk, isk, U)$
$\mathbf{player}_b := \mathsf{honest}(param, gpk, sk_U, cert_U)$;
$\mathbf{player}_{1-b} := \mathsf{SIM}(param)$
$\mathbf{player}_0 \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk, tsk)$
$\mathbf{player}_1 \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk, tsk)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(param, gpk, tsk)$ Return $b'$

## 4 Proposed Scheme

### 4.1 Preliminaries

**Definition 3 (Bilinear map).** This paper follows the notation regarding bilinear groups Let $(\mathbb{G}_1, \mathbb{G}_2)$ be bilinear groups as follows:

- $\mathbb{G}_1$,$\mathbb{G}_2$ are two cyclic groups of prime order $p$.
- $G_1$ is a generator of $\mathbb{G}_1$ and $G_2$ is a generator of $\mathbb{G}_2$.
- $\psi$ is a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(G_2) = G_1$.
- $e$ is a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.
- $e, \psi$ and the group action in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ can be computed efficiently.

In this paper, we adopt a setup algorithm Set on input a security parameter, outputs $\mathbb{G}_1 = \langle G_1 \rangle, \mathbb{G}_2 = \langle G_2 \rangle$ of prime order $p$ that have a bilinear map $e$ and $e(G_1, G_2)$ generates $\mathbb{G}_T$.

**Definition 4 (Descrete Logarithm (DL) Assumption).**
For all PPT algorithm $\mathcal{A}$, the following function $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A}}(l)$ is negligible.

$$\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A}}(l) = \Pr\left[\mathcal{A}(param, xP, P) = x : x \in \mathbb{Z}_p, P \in \mathbb{G}_1\right]$$

where $param$=$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, G, \psi, e) \leftarrow \mathsf{Set}(k)$.

**Definition 5 (Decisional Linear Diffie-Hellman (DLDH) assumption [5]).**
For all PPT algorithm $\mathcal{A}$, the following function $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DLDH}}(l)$ is negligible. Here, $U, V, H, Q \leftarrow_R \mathbb{G}_1, a, b \leftarrow \mathbb{Z}_p$

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DLDH}}(l) =& |\Pr\left[(\mathcal{A}(param, U, V, H, aU, bV, (a+b)H) = \mathsf{yes}\right] \\ & - \Pr\left[(\mathcal{A}(param, U, V, H, aU, bV, Q)) = \mathsf{yes}\right]|\end{aligned}$$

where $param$=$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, G, \psi, e) \leftarrow \mathsf{Set}(k)$.

### 4.2 Group Signature with Message Recovery (GSMR) [12]

Our proposed SHS, including SHS.Co-Trace algorithm and satisfied Co-Traceability and Strong Detector Resistance, is based on the SHS using GSMR (Group Signature with Message Recovery) [12].

A group signature, first introduce by Chaum and van Heyst [7] and followed by [2, 8], allows a member which belong to a group to sign messages on behalf of the group without a member reveal own identity. In group signature system, as the same secret handshake systems, there exists a manager (authority) of group. A manager, in the case of a dispute, can reveal an identity of any group signature make valid group signatures A standard group signature scheme involves consists of five algorithms GS.KeyGen, GS.Join, GS.Sign, GS.Verify, GS.Open. GS.KeyGen ,is a key generation algorithm, is given security parameter and outputs a group public key $gpk$ and a group manager secret key $gmsk$. GS.Join, given $gmsk$ and a member secret key $sk$, outputs membership certificate. GS.Sign, given $gpk$, a member secret key $sk$ and a message $m$, outputs a group signature $\sigma$. GS.Verify, given $gpk$, $m$, and $\sigma$, outputs accept if $\sigma$ is valid for $m$ with respect to $gpk$. GS.Open, given $m, \sigma$, and $gmsk$, outputs signer's identity. The security requirements of a group signature scheme are traceability, anonymity, and non-frameability[4].

Intuitively, [12] achieve Unlink by using group signature (more precisely, anonymity of group signatures) and realized SHS.Trace by SHS.Open. In [12], a group signature with message recovery (GSMR) is constructed from the standard group signature[2]. In order to apply GSMR to SHS, GSMR can be forged a signature corresponding to an arbitrary message using a valid signature and $gpk$. If GSMR does not this property, anyone can be convinced of handshake players' groups by carrying out GS.Verify.

### 4.3 Construction of Proposed Scheme

We show the construction of proposed SHS scheme. Our GSMR scheme is based on [8]. We assume that $\mathbb{G}_1 = \langle G_1 \rangle, \mathbb{G}_2 = \langle G_2 \rangle$ of prime order $p$ that have a bilinear map $e$ and $e(G_1, G_2)$ generates $\mathbb{G}_T$. Here, $(\mathbb{G}_1, \mathbb{G}_2)$ is bilinear groups.

SHS.Setup: Given a security parameter $k$, generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e)$ and chooses hash functions $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$ and $\mathcal{G} : \{0,1\}^* \to \mathbb{Z}_p$.

SHS.CreateGroup: First, IA chooses $w \leftarrow_R \mathbb{Z}_p$ and $H \leftarrow_R \mathbb{G}_1$ and generates $W = wG_2$. IA outputs $isk = w, ipk = (H, W)$. Next, TA chooses $(t, s) \leftarrow_R (\mathbb{Z}_p)^2$ and generates $T, S$ such that $H = sS = tT$. TA outputs $tsk = (t, s), tpk = (T, S)$.

SHS.AddMember: First, user $U$, who wants to join to the group, chooses $(x', z) \leftarrow_R (\mathbb{Z}_p)^2$ and generates $H' = x'H + zG_1$. $U$ sends $H'$ to IA. Next, IA chooses $(x'', z') \leftarrow_R (\mathbb{Z}_p)^2$ and sends $(x'', z')$ to $U$. $U$ generates $x_U = x'x'' + z'$ and $H_U = x_U H$. $U$ sends $H_U$ to IA. $U$ proves in zero-knowlege to IA the knowledge of $x_U$ and $x''z$ satisfying $H_U = x_U H$ and $x''H' + z'H - H_U = x''zG_1$. Finally, IA chooses $y_U \leftarrow_R \mathbb{Z}_p$ and generates $A_U = \frac{1}{w+y_U}(G_1 - H_U)$. IA sends $(A_U, y_U)$ to $U$. IA adds $(U, A_U, y_U)$ to the group member-list $\mathcal{L}$.

GSMR$(param, gpk, sk, cert, m) \to \sigma$:

1. $U$ chooses $s', t', S', T', H'$ s.t. $s'S' = t'T' = H'$ $((s', t') \in (\mathbb{Z}_p)^2$, $(S', T', H') \in (\mathbb{G}_1)^3)$. If $U$ will want to execute SHS.Co-Trace, $U$ has to memorize their parameters.
2. $U$ chooses $(\alpha, \beta, \alpha', \beta') \leftarrow_R \mathbb{Z}_p^*$ and generates $R_1 = \alpha T, R_2 = \beta S, R_3 = \alpha' T', R_4 = \beta' S'$ and $R_5 = (\alpha + \beta)H + (\alpha' + \beta')H' + A_U$
3. $U$ chooses $(r_x, r_y, r_\alpha, r_\beta, r_{y\alpha}, r_{y\beta}, r_{\alpha'}, r_{\beta'}, r_{y\alpha'}, r_{y\beta'}) \leftarrow_R (\mathbb{Z}_p^*)^{10}$ and generate $R'_1 = r_\alpha T, R'_2 = r_\beta S, R'_3 = r_{\alpha'} T', R'_4 = r_{\beta'} S'$, $R'_5 = e(R_5, G_2)^{r_y} e(H, W)^{-(r_\alpha + r_\beta)} e(H', W)^{-(r_{\alpha'} + r_{\beta'})} e(H, G_2)^{-(r_{y\alpha} + r_{y\beta}) + r_x}$ $e(H', G_2)^{-(r_{y\alpha'} + r_{y\beta'})}, R'_6 = r_y R_1 - r_{y\alpha} T, R'_7 = r_y R_2 - r_{y\beta} S, R'_8 = r_y R_3 - r_{y\alpha'} T'$ and $R'_9 = r_y R_4 - r_{y\beta'} S'$
4. $U$ generates $c' = \mathcal{H}(param, gpk, R_1, R_2, R_3, R_4, R_5, R'_1, R'_2, R'_3, R'_4, R'_5, R'_6, R'_7, R'_8, R'_9)$ and $c = c' \oplus m$.
5. $U$ generates $s_x := r_x + cx_U, s_y := r_y + cy_U, s_\alpha := r_\alpha + c\alpha, s_\beta := r_\beta + c\beta s_{y\alpha} := r_{y\alpha} + c(y\alpha), s_{y\beta} := r_{y\beta} + c(y\beta), s_{\alpha'} := r_{\alpha'} + c\alpha', s_{\beta'} := r_{\beta'} + c\beta', s_{y\alpha'} := r_{y\alpha'} + c(y\alpha')$ and $s_{y\beta'} := r_{y\beta'} + c(y\beta')$
6. $U$ outputs $\sigma = (R_1, R_2, R_3, R_4, R_5, s_x, s_y, s_\alpha, s_\beta, s_{y\alpha}, s_{y\beta}, s_{\alpha'}, s_{\beta'}, s_{y\alpha'}, s_{y\beta'}, c)$

MR:$(param, gpk, \sigma) \to m$

1. $V$ is given $param, gpk$, and $\sigma$.

2. $V$ generates $R'_1 = s_\alpha T - cR_1, R'_2 = s_\beta S - cR_2, R'_3 = s_\alpha T' - cR_3, R'_4 = s_\beta S' - cR_4, R'_5 = e(R_5, G_2)^{s_y} e(H, W)^{-(s_\alpha + s_\beta)} e(H, G_2)^{-(s_{y\alpha} + s_{y\beta}) + s_x}$ $e(H', W)^{-(s_{\alpha'} + s_{\beta'})} e(H, G_2)^{-(s_{y\alpha'} + s_{y\beta'})} \left(\frac{e(R_5, G_2)}{e(G_1, G_2)}\right)^c, R'_6 = s_y R_1 - s_{y\alpha} T$, $R'_7 = s_y R_2 - s_{y\beta} S, R'_8 = s_y R_3 - s_{y\alpha'} T'$, and $R'_9 = s_y R_4 - s_{y\beta'} S'$
3. $V$ generates $c' = \mathcal{H}(param, gpk, R_1, R_2, R_3, R_4, R_5, R'_1, R'_2, R'_3, R'_4, R'_5, R'_6, R'_7, R'_8, R'_9)$ and $m = c \oplus c'$. $V$ outputs $m$.

SHS.Handshake: Suppose the member $U$ with certificate $cert_U = (A_U, y_U)$ and secret key $sk_U = x_U$ , and the member $V$ with certificate $cert_V = (A_V, y_V)$ and secret key $sk_V = x_V$, engage in handshake protocol.

1. $U$ and $V$ generates $(s'_U, t'_U, S'_U, T'_U, H'_U)$, $(s'_V, t'_V, S'_V, T'_V, H'_V)$ s.t. $s'_U S_U = t'_U T = H'_U, s'_V S_V = t'_V T = H'_V$ $(s'_U, t'_U, s'_V, t'_V \in (\mathbb{Z}_p)^4, S'_U, T'_U, H'_U, S'_V, T'_V, H'_V \in \mathbb{G}_1)$.
2. $U$ chooses $r_U \leftarrow_R \mathbb{Z}_p^*$ and generates $m_U := r_U G_1$ and $\sigma_U \leftarrow \mathsf{GSMR}(param, gpk_U, sk_U, cert_U, m_U)$. $U$ send $\sigma_U$ to $V$.
3. $V$ chooses $r_V \leftarrow_R \mathbb{Z}_p^*$ and generates $m_V := r_V G_1$, $\sigma_V \leftarrow \mathsf{GSMR}(param, gpk_V, sk_V cert_V, m_V)$ and $m'_U \leftarrow \mathrm{MR}(param, gpk_V, \sigma_U)$. $V$ send $\sigma_V$ to $U$.
4. $U$ generates $m'_V \leftarrow \mathrm{MR}(param, gpk_U, \sigma_V)$ and $resp_U := \mathcal{G}(r_U m'_V, m_U)$ and send $resp_U$ to $V$.
5. $V$ generates $resp_V := \mathcal{G}(r_V m'_U, m_V)$. If $resp_U = \mathcal{G}(r_V m'_U, m'_U)$, $V$ outputs *accept* and send $resp_V$ to $U$. Otherwise $V$ outputs *reject*.
6. If $resp_V = \mathcal{G}(r_U m'_V, m'_V)$, $U$ outputs *accept*. Otherwise $U$ outputs *reject*.

SHS.Co-Trace: Suppose the member $U$ with $cert_U = (A_U, y_U)$, secret key $sk_U = x_U$, and the parameters $(s'_U, t'_U, S'_U, T'_U, H'_U)$ used in SHS.Handshake.

1. TA is given $cert_U = (A_U, y_U), sk_U = x_U$ from $U$. If $e(A, W)e(H, G_2)^x e(A, G_2)^y = e(G_1, G_2)$, TA executes the following. Otherwise TA outputs $\perp$.
2. TA is given $\sigma_V (= R_{V1}, R_{V2}, R_{V3}, R_{V4}, R_{V5}, s_{Vx}, s_{Vy}, s_{V\alpha}, s_{V\beta}, s_{Vy\alpha}, s_{Vy\beta}, s_{V\alpha'}, s_{V\beta'}, s_{Vy\alpha'}, s_{Vy\beta'}, c_V)$ (in the transcript of SHS.Handshake) and computes $R_{V5} - (tR_{V1} s R_{V2} + t' R_{V3} s' R_{V4}) = A_V$ and outputs $V$ such that $(V, A_V, y_V) \in \mathcal{L}$.

Among SHS, we show the comparisons with respect to important factors in Table4.3. [3], [1] and proposed scheme use bilinear maps. We assume that $\mathbb{G}_1 \neq \mathbb{G}_2$ such that the representation of $G_1$ can be a 172-bit prime when $|p| = 171$. Also, we assume that [6] are instantiated on a 160-bits prime order subgroup of a prime finite field of 1024 bits. In [3], [6] and

proposed scheme, a member has own ID. On the other hand, in [1], a member does not have ID, so [1] can not include both SHS.Trace and SHS.Co-Trace.

**Table 2.** Comparison among SHS

| Scheme | [3] | [6] | [1] | Proposed Scheme |
|---|---|---|---|---|
| Underlying Assumption | CBDH | CDH | SXDH and BDH | DL and DLDH |
| Number of rounds | 3 | 4 | 2 | 4 |
| Communication complexity (bits) | 640 | 8512 | 684 | 5804 |
| IR security | Yes | Yes | Yes | Yes |
| DR security | Yes | Yes | Yes | Yes |
| Unlink security | No | No | Yes | Yes |
| SDR security | No | No | Yes | Yes |
| Co-Trace security | No | No | No | Yes |
| SHS.Trace algorithm | Yes | Yes | No | No |
| SHS.Co-Trace algorithm | No | No | No | Yes |

Our proposed scheme is secure under discrete logarithm assumption and Decisional Linear Diffie-Hellman (DLDH) assumption[5].

**Theorem 1.** *The proposed scheme has Impersonator Resistance property, if the discrete logarithm problem is hard to solve.*

**Theorem 2.** *The proposed scheme has Strong Detector Resistance, Unlinkability and Co-Traceability property, if the Decisional Linear Diffie-Hellman (DLDH) problem is hard to solve.*

## References


1. G. Ateniese, M. Blanton, and J. Kirsch. Secret handshakes with dynamic and fuzzy matching. In *Network and Distributed System Security Symposium*, 2007.
2. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology CRYPTO2000*, volume 1880 of LNCS, pages 255–270. Springer, 2000.
3. D. Balfanz, G. Durfee, N. Shankar, D.K. Smetters, J. Staddon, and H. C. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy, 2003*, pages 180–196. IEEE Computer Society, 2003.
4. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptology CT-RSA 2005*, volume 3376 of LNCS, pages 136–153. Springer, 2005.

5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology Crypto 2004*, volume 3152 of LNCS, pages 41–55. Springer, 2004.
6. C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from ca-oblivious encryption. In *Advances in Cryptology ASIACRYPT2004*, volume 3329 of LNCS, pages 293–307. Springer, 2004.
7. D. Chaum and E. Van Heyst. Group signatures. In *Advances in Cryptology EUROCRYPT 1991*, volume 265 of LNCS, pages 257–265. Springer, 1991.
8. J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. In *Information Security And Privacy, ACISP 2005*, volume 3574 of LNCS, pages 455–467. Springer, 2005.
9. D. Pointcheval. Security arguments for digital signatures and blind signatures. In *Journal of Cryptology*, volume 13, pages 361–396. Springer, 2000.
10. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *the Symposium on Cryptography and Information Security , SCIS 2000*, 2000.
11. S. Xu and M. Yung. k-anonymous secret handshakes with reusable credentials. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 158–167. ACM, 2004.
12. Takeshi Yamashita and Misuru Tada. How to construct secret handshake schemes (in japanese). In *Computer Security Symposium 2007, CSS2007*, pages 321–326, 2006.

## A Security Proof

**Theorem 1.** *The proposed scheme has Impersonator Resistance property, if the discrete logarithm problem is hard to solve.*

*Proof.* We show that if there exists an adversary $\mathcal{A}$ breaks Impersonator Resistance, then an algorithm $\mathcal{B}$ which solves the discrete logarithm problem can be constructed using $\mathcal{A}$ as a black-box. We show the description of $\mathcal{B}$ as follow:

Algorithm $\mathcal{B}$ is given $(param, H, Y = x_U H)$. First, $\mathcal{B}$ generates $(isk, ipk) = (w, (H, W = wG_2))$ s.t. $w \in \mathbb{Z}_p, H \in \mathbb{G}_1$ and $(tsk, tpk) = ((s,t),(S,T))$ s.t. $s, t \in \mathbb{Z}_p, sS = tT = H \in \mathbb{G}_1$ according to SHS.CreateGroup. Next, $\mathcal{B}$ generates a member-list $\mathcal{L}$. $\mathcal{B}$ chooses $(x_U, y_U, y_V) \leftarrow_R (\mathbb{Z}_p)^3$ and $(x_i, y_i) \leftarrow_R (\mathbb{Z}_p)^2 (i = 1, ..., n)$ and generates $A_U = \frac{1}{w+y_U}(G_1 - x_U H)$, $A_V = \frac{1}{w+y_V}(G_1 - Y)$, and $A_i = \frac{1}{w+y_i}(G_1 - x_i H)$. $\mathcal{B}$ adds $(i, A_i, y_i)(i = 1, ..., n)$, $(U, A_U, y_U)$, and $(V, A_V, y_V)$to $\mathcal{L}$. Also $\mathcal{B}$ generates a list $\mathcal{L}'$ including a secret key $sk_i = x_i$ and an internal information $I_i$ (e.g. random numbers which were used when a member $i$ executed SHS.Handshake). Elements of list $\mathcal{L}'$ is $(i, x_i, A_i, y_i, I_i)(i = 1, ...n, U)$ and $(V, \perp, A_V, y_V, \perp)$.

$\mathcal{A}$ is given $(param, H, W, S, T, cert_V (= y_V, A_V))$ by $\mathcal{B}$. When $\mathcal{A}$ access $\mathcal{O}_{\mathcal{AM}}$, $\mathcal{O}_{\mathcal{CG}}$, $\mathcal{O}_{\mathcal{HS}}$, $\mathcal{O}_{CO}$, random oracle $\mathcal{O}_{\mathcal{H}}$, and random oracle $\mathcal{O}_{\mathcal{G}}$, $\mathcal{B}$ behaves as follow:

– $\mathcal{O}_{\mathcal{AM}}$: When $\mathcal{A}$ querys a group identity $G$ and a user name $A$, $\mathcal{B}$ executes SHS.AddMember using $isk = w$.

– $\mathcal{O}_{\mathcal{CG}}$: When $\mathcal{A}$ querys an identity of $GA$, $\mathcal{B}$ executes SHS.CreateGroup.
– $\mathcal{O}_{\mathcal{HS}}$: When $\mathcal{A}$ querys a member ID $i$, $\mathcal{B}$ executes SHS.Handshake as a member $i$ according to the scheme. If $(i, *, *) \notin \mathcal{L}$, $\mathcal{B}$ outputs $\perp$. If $\mathcal{A}$ query $V$, $\mathcal{B}$ abort. The probability that this case occurs is $\frac{1}{2^k}$.
– $\mathcal{O}_{\mathcal{CO}}$ When $\mathcal{A}$ querys a member ID $i$, $\mathcal{B}$ return a secret key $sk_i$ and an internal information $I_i$ s.t. $(i, x_i, A_i, y_i, I_i) \in \mathcal{L}'$. If $(i, x_i, A_i, y_i, I_i) \notin \mathcal{L}'$, $\mathcal{B}$ return $\perp$. If $\mathcal{A}$ query $V$, $\mathcal{B}$ abort. The probability that this case occurs is $\frac{1}{2^k}$.
– $\mathcal{O}_{\mathcal{H}}$: When $\mathcal{A}$ querys $(param, gpk_i, R_{i1}, ..., R_{i5}, R'_{i1}, ..., R'_{i9})$, if its query is first, $\mathcal{B}$ chooses $c'_i \leftarrow_R \mathbb{Z}_p$ and return it to $\mathcal{A}$ and add $(i, gpk_i, R_{i1}, ..., R_{i5}, R'_{i1}, ..., R'_{i9}, c'_i)$ in H-List. if $(i, gpk_i, R_{i1}, ..., R_{i5}, R'_{i1}, ..., R'_{i9}, c'_i) \in$H-List, $\mathcal{B}$ return $c'_i$.
– $\mathcal{O}_{\mathcal{G}}$: When $\mathcal{A}$ querys $(M1_i, M2_i)$, if its query is first, $\mathcal{B}$ chooses $resp_i \leftarrow_R \mathbb{G}_1$ and return it to $\mathcal{A}$ and add $(i, resp_i, M1_i, M2_i)$ in G-List. If $(i, resp_i, M1_i, M2_i) \in$G-List, $\mathcal{B}$ return $resp_i$.

When $\mathcal{B}$, as honest, executes SHS.Handshake with $\mathcal{A}$, $\mathcal{B}$ behaves as follow:

– $\mathcal{B}$ generates $(s'_B, t'_B, S'_B, T'_B, H'_B)$ s.t. $s'_B S_B = t'_B T = H'_B$.
– $\mathcal{B}$ chooses $r_U \leftarrow_R \mathbb{Z}_p$ and generates $m_U = r_U G_1$ and generate
– $\mathcal{B}$ receives $\sigma_A = (R_{A1}, ..., R_{A5}, s_{Ax}, s_{Ay}, s_{A\alpha}, s_{A\beta}, s_{Ay\alpha}, s_{Ay\beta}, s_{A\alpha'}, s_{A\beta'}, s_{Ay\alpha'}, s_{Ay\beta'}, c_A)$ and executes $m'_A \leftarrow \mathsf{MR}(param, gpk, \sigma_A)$. $\mathcal{B}$ generate $resp_B = \mathcal{G}(r_B m'_A, m_B)$ and sends it to $\mathcal{A}$.
– $\mathcal{B}$ receives $resp_A = \mathcal{G}(r_A m_V, m_A) = \mathcal{G}(r_V m'_A, m'_A)$. (Since $\mathcal{A}$ can break IR, $resp_A = \mathcal{G}(r_A m_V, m_A) = \mathcal{G}(r_V m'_A, m'_A)$.)

If $(r_A m_V, m_A, resp_A) \notin$ G-List or , $(i, gpk, R_{A1}, ..., R_{A5}, R'_{A1}, ..., R'_{A9}, c'_A) \notin$ H-List $\mathcal{B}$ abort.

Then, from the Forking Lemma [9] by rewinding $\mathcal{A}$ and choosing other random oracle $\mathcal{O}_{\mathcal{H}}$, $\mathcal{B}$ is given two set as follows:

$$\{R_1, R_2, R_3, R_4, R_5, R'_1, R'_2, R'_3, R'_4, R'_5, R'_6, R'_7, R'_8, R'_9, s_x^{(1)}, s_y^{(1)}, s_\alpha^{(1)}, s_\beta^{(1)}, s_{y\alpha}^{(1)}, s_{y\beta}^{(1)}, s_{\alpha'}^{(1)}, s_{\beta'}^{(1)}, s_{y\alpha'}^{(1)}, s_{y\beta'}^{(1)}, c^{(1)}\}$$

$$\{R_1, R_2, R_3, R_4, R_5, R'_1, R'_2, R'_3, R'_4, R'_5, R'_6, R'_7, R'_8, R'_9, s_x^{(2)}, s_y^{(2)}, s_\alpha^{(2)}, s_\beta^{(2)}, s_{y\alpha}^{(2)}, s_{y\beta}^{(2)}, s_{\alpha'}^{(2)}, s_{\beta'}^{(2)}, s_{y\alpha'}^{(2)}, s_{y\beta'}^{(2)}, c^{(2)}\}$$

such that
$(s_x^{(1)}, s_y^{(1)}, s_\alpha^{(1)}, s_\beta^{(1)}, s_{y\alpha}^{(1)}, s_{y\beta}^{(1)}, s_{\alpha'}^{(1)}, s_{\beta'}^{(1)}, s_{y\alpha'}^{(1)}, s_{y\beta'}^{(1)}, c^{(1)})$

$$\neq (s_x^{(2)}, s_y^{(2)}, s_\alpha^{(2)}, s_\beta^{(2)}, s_{y\alpha}^{(2)}, s_{y\beta}^{(2)}, s_{\alpha'}^{(2)}, s_{\beta'}^{(2)}, s_{y\alpha'}^{(2)}, s_{y\beta'}^{(2)}, c^{(2)})$$

and

$$\begin{aligned}
R'_1 &= s_\alpha^{(1)} T - c^{(1)} R_1 = s_\alpha^{(2)} T - c^{(2)} R_1 \\
R'_2 &= s_\beta^{(1)} S - c^{(1)} R_2 = s_\beta^{(2)} S - c^{(2)} R_2 \\
R'_3 &= s_\alpha^{(1)} T - c^{(1)} R_3 = s_\alpha^{(2)} T - c^{(2)} R_3 \\
R'_4 &= s_\beta^{(1)} S - c^{(1)} R_4 = s_\beta^{(2)} S - c^{(2)} R_4 \\
R'_5 &= e(R_5, G_2)^{s_y^{(1)}} e(H, W)^{-(s_\alpha^{(1)} + s_\beta^{(1)})} e(H, G_2)^{-(s_{y\alpha}^{(1)} + s_{y\beta}^{(1)}) + s_x^{(1)}} \\
&\quad e(H', W)^{-(s_{\alpha'}^{(1)} + s_{\beta'}^{(1)})} e(H, G_2)^{-(s_{y\alpha'}^{(1)} + s_{y\beta'}^{(1)})} \left( \frac{e(R_5, G_2)}{e(G_1, G_2)} \right)^{c^{(1)}} \\
&= e(R_5, G_2)^{s_y^{(2)}} e(H, W)^{-(s_\alpha^{(2)} + s_\beta^{(2)})} e(H, G_2)^{-(s_{y\alpha}^{(2)} + s_{y\beta}^{(2)}) + s_x^{(2)}} \\
&\quad e(H', W)^{-(s_{\alpha'}^{(2)} + s_{\beta'}^{(2)})} e(H, G_2)^{-(s_{y\alpha'}^{(2)} + s_{y\beta'}^{(2)})} \left( \frac{e(R_5, G_2)}{e(G_1, G_2)} \right)^{c^{(2)}} \\
R'_6 &= s_y^{(1)} R_1 - s_{y\alpha}^{(1)} T = s_y^{(2)} R_1 - s_{y\alpha}^{(2)} T \\
R'_7 &= s_y^{(1)} R_2 - s_{y\alpha}^{(1)} S = s_y^{(2)} R_2 - s_{y\alpha}^{(2)} S \\
R'_8 &= s_y^{(1)} R_1 - s_{y\alpha}^{(1)} T' = s_y^{(2)} R_1 - s_{y\alpha}^{(2)} T' \\
R'_9 &= s_y^{(1)} R_2 - s_{y\alpha}^{(1)} S' = s_y^{(2)} R_2 - s_{y\alpha}^{(2)} S'
\end{aligned}$$

then $\mathcal{B}$ can compute $x_V = \frac{s_x^{(1)} - s_x^{(2)}}{c^{(1)} - c^{(2)}}$. This solves the discrete logarithm problem.

The success probability of $\mathcal{B}$ is as followes; Here, $\varepsilon_{\mathcal{A}}$ and $\varepsilon_{\mathcal{B}}$ are success probabilities of $\mathcal{A}$ and $\mathcal{B}$, and $q_h$ is the number of query to $\mathcal{O}_{\mathcal{H}}$. From similar discussion of [9], $\varepsilon_{\mathcal{B}} = \frac{1}{4}\varepsilon_{\mathcal{A}}^2 \frac{1}{q_h}(1 - \frac{1}{2^k}) - \frac{1}{2^{k-1}}$. □

**Theorem 2.** *The proposed scheme has Strong Detector Resistance, Unlinkability and Co-Traceability property, if the Decisional Linear Diffie-Hellman (DLDH) problem is hard to solve.*

**Lemma 1.** *The proposed scheme has Strong Detector Resistance property, if the Decisional Linear Diffie-Hellman (DLDH) problem is hard to solve.*

*Proof.* We show that if there exists an adversary $\mathcal{A}$ breaks Strong Detector Resistance, then an algorithm $\mathcal{B}$ which solves the DLDH problem can be constructed using $\mathcal{A}$ as a black-box. We show the description of $\mathcal{B}$ as follow:

An algorithm $\mathcal{B}$ is given $(param, T', S', H', aT', bS', Q)$. Here, $(T', S', H', aT', bS', Q)$ is DLDH instance. First, $\mathcal{B}$ generates $(isk, ipk) = (w, (H, W = wG_2))$ s.t. $w \in \mathbb{Z}_p, H \in \mathbb{G}_1$ and $(tsk, tpk) = ((s,t),(S,T))$ s.t. $s,t \in \mathbb{Z}_p, sS = tT = H \in \mathbb{G}_1$ according to $\mathsf{SHS.CreateGroup}$. $\mathcal{B}$ chooses $(x_U, y_U) \leftarrow_R (\mathbb{Z}_p)^2$ and $(x_i, y_i) \leftarrow_R (\mathbb{Z}_p)^2 (i = 1, ..., n)$ and generates $A_U = \frac{1}{w+y_U}(G_1 - x_U H)$ and $A_i = \frac{1}{w+y_i}(G_1 - x_i H)$. $\mathcal{B}$ adds $(i, A_i, y_i)(i = 1, ..., n)$ and $(U, A_U, y_U)$ to a member-list $\mathcal{L}$. Also $\mathcal{B}$ generates a list $\mathcal{L}'$ including a secret key $sk_i = x_i$ and an internal information $I_i$ . Elements of list $\mathcal{L}'$ is $(i, x_i, A_i, y_i, I_i)(i = 1, ...n, U)$.

$\mathcal{A}$ is given $(param, H, W, S, T, s, t)$ and $\mathcal{L}$ by $\mathcal{B}$. When $\mathcal{A}$ access $\mathcal{O}_{\mathcal{AM}}$, $\mathcal{O}_{\mathcal{CG}}$, $\mathcal{O}_{\mathcal{HS}}$, random oracle $\mathcal{O}_{\mathcal{H}}$, and random oracle $\mathcal{O}_{\mathcal{G}}$, $\mathcal{B}$ simulates them by the way same as the proof of $\mathsf{IR}$. The method of simulating $\mathcal{O}_{\mathcal{CO}}$ is as follows ($\mathcal{A}$ querys $i$). if $i = U$, $\mathcal{B}$ abort. If the length of $ID$ is $k$ bits, the probability that this case occurs is $\frac{1}{2^k}$. If $(i, x_i, A_i, y_i, I_i) \in \mathcal{L}'$, $\mathcal{B}$ returns $sk_i, I_i$. If $(i, x_i, A_i, y_i, I_i) \notin \mathcal{L}'$, $\mathcal{B}$ returns $\perp$.

$\mathcal{B}$ picks up $b \leftarrow_R \{0, 1\}$. $\mathcal{B}$ executes $\mathsf{handshake}$ with $\mathcal{A}$, 1) when $b = 0$ as $U$ for the first time and as $\mathsf{SIM}$ for the second time, 2) when $b = 1$ as $\mathsf{SIM}$ for the first time and as $U$ for the second time.

When $\mathcal{B}$, as $\mathsf{honest}$ $U$, executes $\mathsf{SHS.Handshake}$ with $\mathcal{A}$, $\mathcal{B}$ behaves as follow:

- $\mathcal{B}$ sends $(T', S', H')$ to $\mathcal{A}$ and receives $(T'_A, S'_A, H'_A)$ from $\mathcal{A}$.
- $\mathcal{B}$ chooses chooses $r_U \leftarrow_R \mathbb{Z}_p$ and generates $m_U = r_U G_1$. When $\mathcal{B}$ generates $\sigma_B$ using $\mathsf{GSMR}$, $\mathcal{B}$ generates $R_3 := aT', R_4 := bS', R_5 = (\alpha + \beta)H + Q + A_U$. So, $\mathcal{B}$ generates $\sigma_B \leftarrow \mathsf{GSMR}(param, gpk, cert_U, sk_U, m_U)$. $\mathcal{B}$ sends it to $\mathcal{A}$.
- $\mathcal{B}$ receives $\sigma_A$ and executes $m'_A \leftarrow \mathsf{MR}(param, gpk, \sigma_A)$.$\mathcal{B}$ generate $resp_B = \mathcal{G}(r_B m'_A, m_B)$ and sends it to $\mathcal{A}$.
- $\mathcal{B}$ receives $resp_A$.

When $\mathcal{B}$, as $\mathsf{SIM}$, executes $\mathsf{SHS.Handshake}$ with $\mathcal{A}$, $\mathcal{B}$ behaves as follow:

- $\mathcal{B}$ chooses $(T'_{SIM}, S'_{SIM}, H'_{SIM}) \leftarrow_R (\mathbb{G}_1)^3$ and sends it to $\mathcal{A}$ and receives $(T'_A, S'_A, H'_A)$ from $\mathcal{A}$.
- $\mathcal{B}$ chooses $(R_{SIM1}, R_{SIM2}, R_{SIM3}, R_{SIM4}, R_{SIM5}, c_{SIM}) \leftarrow_R (\mathbb{G}_1)^6$ and $(s_{SIMx}, s_{SIMy}, s_{SIM\alpha}, s_{SIM\beta}, s_{SIMy\alpha}, s_{SIMy\beta}, s_{SIM\alpha'}, s_{SIM\beta'}, s_{SIMy\alpha'}, s_{SIMy\beta'}) \leftarrow_R (\mathbb{Z}_p)^{10}$.
  $\mathcal{B}$ sends it to $\mathcal{A}$.
- $\mathcal{B}$ receives $\sigma_A$ and executes $m'_A \leftarrow \mathsf{MR}(param, gpk, \sigma_A)$. $\mathcal{B}$ chooses $resp_{SIM} \leftarrow_R \mathbb{Z}_p$ and sends it to $\mathcal{A}$.
- $\mathcal{B}$ receives $resp_A$.

$\mathcal{A}$ outputs $b'$. If $(T', S', H', aT', bS', Q)$ is satisfied $Q = (a+b)H'$, $\mathcal{A}$ can distinguish $U$ and $\mathsf{SIM}$, since $\mathcal{B}$ can completely simulate honest $U$. Then, if $b = b'$, $\mathcal{B}$ outputs $\mathsf{yes}$, otherwise outputs $\mathsf{no}$. This solves the DLDH problem.

Here, $\varepsilon_{\mathcal{A}}$ and $\varepsilon_{\mathcal{B}}$ are success probabilities of $\mathcal{A}$ and $\mathcal{B}$. If $(T', S', H', aT', bS', Q)$ is the random tuple, the probability that $b = b'$ is $\frac{1}{2}$. Also, if $(T', S', H', aT', bS', Q)$ is the DLDH tuple, the probability that $b = b'$ is $\varepsilon_{\mathcal{A}} + \frac{1}{2}$. So, the advantage of $\mathcal{B}$ is $\varepsilon_{\mathcal{B}} = \varepsilon_{\mathcal{A}} + \frac{1}{2} - \frac{1}{2} - \frac{1}{2^k} = \varepsilon_{\mathcal{A}} - \frac{1}{2^k}$. So, if $\varepsilon_{\mathcal{A}}$ is non-negligible, $\varepsilon_{\mathcal{B}}$ is non-negligible. □

**Lemma 2.** *The proposed scheme has Unlinkability property, if the Decisional Linear Diffie-Hellman (DLDH) problem is hard to solve.*

*Proof.* We show that if there exists an adversary $\mathcal{A}$ breaks Unlinkability, then an algorithm $\mathcal{B}$ which solves the DLDH problem can be constructed using $\mathcal{A}$ as a black-box. We show the description of $\mathcal{B}$ as follow:

Algorithm $\mathcal{B}$ is given $(param, T, S, H, aT, bS, Q)$. First, $\mathcal{B}$ generates $(isk, ipk) = (w, (H, W = wG_2))$ s.t. $w \in \mathbb{Z}_p, H \in \mathbb{G}_1$ according to $\mathsf{SHS.CreateGroup}$ and $tpk = (S, T)$ ($itk$ is unknown by $\mathcal{B}$).

Next, $\mathcal{B}$ generates a member-list $\mathcal{L}$. $\mathcal{B}$ chooses $(x_U, y_U, x_V, y_V) \leftarrow_R (\mathbb{Z}_p)^4$ and $(x_i, y_i) \leftarrow_R (\mathbb{Z}_p)^2 (i = 1, ..., n)$ and generates $A_i = \frac{1}{w+y_i}(G_1 - x_i H)(i = 1, \ldots, n, U, V)$. $\mathcal{B}$ adds $(i, A_i, y_i)(i = 1, ..., n)$, $(U, A_U, y_U)$ and $(V, A_V, y_V)$ to a member-list $\mathcal{L}$. Also $\mathcal{B}$ generates a list $\mathcal{L}'$ including a secret key $sk_i = x_i$ and an internal information $I_i$ Elements of list $\mathcal{L}'$ is $(i, x_i, A_i, y_i, I_i)(i = 1, ...n, U, V)$.

$\mathcal{A}$ is given $(param, H, W, S, T)$ by $\mathcal{B}$. When $\mathcal{A}$ access $\mathcal{O}_{\mathcal{AM}}$, $\mathcal{O}_{\mathcal{CG}}$, $\mathcal{O}_{\mathcal{HS}}$, $\mathcal{O}_{\mathcal{CO}}$ random oracle $\mathcal{O}_{\mathcal{H}}$, and random oracle $\mathcal{O}_{\mathcal{G}}$, $\mathcal{B}$ simulates them by the way same as the proof of $\mathsf{SDR}$.

$\mathcal{B}$ picks up $b \leftarrow_R \{0, 1\}$. $\mathcal{B}$ executes $\mathsf{handshake}$ with $\mathcal{A}$, 1) when $b = 0$ twice as $U$, 2) when $b = 1$ as $U$ for the first time and as $V$ for the second time.

When $\mathcal{B}$, as $\mathsf{honest}$, executes $\mathsf{SHS.Handshake}$ with $\mathcal{A}$, $\mathcal{B}$ behaves as follow: (In the following behavior, $\mathcal{B}$ carries out $\mathsf{SHS.Handshake}$ as U. When $\mathcal{B}$ carry it out as V, it is similar.)

- $\mathcal{B}$ generates $(s'_B, t'_B, S'_B, T'_B, H'_B)$ s.t. $s'_B S_B = t'_B T = H'_B$.
- $\mathcal{B}$ chooses chooses $r_U \leftarrow_R \mathbb{Z}_p$ and generates $m_U = r_U G_1$. When $\mathcal{B}$ generates $\sigma_B$ using $\mathsf{GSMR}$, $\mathcal{B}$ generates $R_3 := aT', R_4 := bS', R_5 = (\alpha + \beta)H + Q + A_U$. So, $\mathcal{B}$ generates $\sigma_B \leftarrow \mathsf{GSMR}(param, gpk, cert_U, sk_U, m_U)$. $\mathcal{B}$ sends it to $\mathcal{A}$.
- $\mathcal{B}$ receives $\sigma_A$ and executes $m'_A \leftarrow \mathsf{MR}(param, gpk, \sigma_A)$. $\mathcal{B}$ generate $resp_B = \mathcal{G}(r_B m'_A, m_B)$ and ]sends it to $\mathcal{A}$.

– $\mathcal{B}$ receives $resp_A$.

Finally, $\mathcal{A}$ output $b'$. If $b = b'$, $\mathcal{B}$ outputs yes, otherwise outputs no. This solves the DLDH problem.

Here, $\varepsilon_{\mathcal{A}}$ and $\varepsilon_{\mathcal{B}}$ are success probabilities of $\mathcal{A}$ and $\mathcal{B}$. By the same discussion of Lemma 1, The advantage of $\mathcal{B}$ is $\varepsilon_{\mathcal{B}} = \varepsilon_{\mathcal{A}} + \frac{1}{2} - \frac{1}{2} - \frac{1}{2^k} = \varepsilon_{\mathcal{A}} - \frac{1}{2^k}$. So, if $\varepsilon_{\mathcal{A}}$ is non-negligible, $\varepsilon_{\mathcal{B}}$ is non-negligible.
□

**Lemma 3.** *The proposed scheme has Co-Traceability property, if the Decisional Linear Diffie-Hellman (DLDH) problem is hard to solve.*

*Proof.* We show that if there exists an adversary $\mathcal{A}$ breaks Co-Traceability, then an algorithm $\mathcal{B}$ which solves the DLDH problem can be constructed using $\mathcal{A}$ as a black-box. We show the description of $\mathcal{B}$ as follow:

An algorithm $\mathcal{B}$ is given $(param, T', S', H', aT', bS', Q)$. First, $\mathcal{B}$ generates $(isk, ipk) = (w, (H, W = wG_2))$ s.t. $w \in \mathbb{Z}_p, H \in \mathbb{G}_1$ and $(tsk, tpk) = ((s,t),(S,T))$ s.t. $s, t \in \mathbb{Z}_p, sS = tT = H \in \mathbb{G}_1$ according to SHS.CreateGroup.

Next, $\mathcal{B}$ generates a member-list $\mathcal{L}$. $\mathcal{B}$ chooses $(x_U, y_U) \leftarrow_R (\mathbb{Z}_p)^2$ and $(x_i, y_i) \leftarrow_R (\mathbb{Z}_p)^2 (i = 1, ..., n)$ and generates $A_U = \frac{1}{w+y_U}(G_1 - x_U H)$ and $A_i = \frac{1}{w+y_i}(G_1 - x_i H)$. $\mathcal{B}$ adds $(i, A_i, y_i)(i = 1, ..., n)$ and $(U, A_U, y_U)$ to $\mathcal{L}$. Also, $\mathcal{B}$ generates a list $\mathcal{L}'$ including a secret key $sk_i = x_i$ and an internal information $I_i$. Elements of list $\mathcal{L}'$ is $(i, x_i, A_i, y_i, I_i)$(i=1,...n,U).

$\mathcal{A}$ is given $(H, W, S, T, w, s, t, \mathcal{L})$ by $\mathcal{B}$. When $\mathcal{A}$ access $\mathcal{O}_{\mathcal{AM}}$, $\mathcal{O}_{\mathcal{CG}}$, $\mathcal{O}_{\mathcal{HS}}$, $\mathcal{O}_{\mathcal{CO}}$ random oracle $\mathcal{O}_{\mathcal{H}}$, and random oracle $\mathcal{O}_{\mathcal{G}}$, $\mathcal{B}$ simulates them by the way same as the proof of SDR.

When $\mathcal{B}$, as honest, executes SHS.Handshake with $\mathcal{A}$, $\mathcal{B}$ behaves as follow:

– $\mathcal{B}$ sends $(T', S', H')$ to $\mathcal{A}$ and receives $(T'_A, S'_A, H'_A)$ from $\mathcal{A}$.
– $\mathcal{B}$ chooses chooses $r_U \leftarrow_R \mathbb{Z}_p$ and generates $m_U = r_U G_1$. When $\mathcal{B}$ generates $\sigma_B$ using GSMR, $\mathcal{B}$ generates $R_3 := aT', R_4 := bS', R_5 = (\alpha + \beta)H + Q + A_U$. So, $\mathcal{B}$ generates $\sigma_B \leftarrow \mathsf{GSMR}(param, gpk, cert_U, sk_U, m_U)$. $\mathcal{B}$ sends it to $\mathcal{A}$.
– $\mathcal{B}$ receives $\sigma_A$ and executes $m'_A \leftarrow \mathsf{MR}(param, gpk, \sigma_A)$. $\mathcal{B}$ generate $resp_B = \mathcal{G}(r_B m'_A, m_B)$ and sends it to $\mathcal{A}$.
– $\mathcal{B}$ receives $resp_A = \mathcal{G}(r_A m_V, m_A) = \mathcal{G}(r_V m'_A, m'_A)$.

$\mathcal{A}$ output $(U', A_{U'}, y_{U'}) \in \mathcal{L}$. If $U' = U$, $\mathcal{B}$ outputs yes, otherwise outputs no. This solves the DLDH problem.
□