Information Sciences 222 (2013) 761–778

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Group signatures with controllable linkability for dynamic membership

Jung Yeon Hwang [a], Sokjoon Lee [a], Byung-Ho Chung [a], Hyun Sook Cho [a], DaeHun Nyang [b,*]

[a] Electronics and Telecommunications Research Institute (ETRI), Daejeon 305-700, Republic of Korea
[b] School of Information and Computer Engineering, InHa University, Incheon 402-751, Republic of Korea

ARTICLE INFO

Article history:
Received 9 November 2010
Received in revised form 2 July 2012
Accepted 30 July 2012
Available online 14 August 2012

Keywords:
Group signature
Anonymity
Traceability
Non-frameability
Linkability
Join

ABSTRACT

In this paper we present a novel group signature scheme for dynamic membership which enables fine-grained control over the release of user information. This scheme could be widely used for various anonymity-based applications such as privacy-preserving data mining and customized anonymous authentication owing to a useful property called *controllable linkability*. A valid signer is able to create signatures that hide his or her identity as normal group signatures but can be anonymously linked regardless of changes to the membership status of the signer and without exposure of the history of the joining and revocation. From signatures, only linkage information can be disclosed, with a special linking key. Using this controllable linkability and the controllable anonymity of a group signature, anonymity may be flexibly or elaborately controlled according to a desired level. To begin construction of our scheme, we first introduce the Decision Linear Combination (DLC) assumption in a so-called gap Diffie–Hellman group where the DDH problem is tractable but the CDH problem is hard, and we prove that this assumption can be guaranteed in generic bilinear groups. To identify security requirements more precisely, we formally present definitions of anonymity, traceability, non-frameabilty, and linkability. We then prove that our scheme achieves all these security properties in the random oracle model. Our scheme supporting controllable linkability yields a short signature that is only 33.3% longer than the best-known normal group signature. Furthermore, we show that our scheme is comparable to the group signature scheme in terms of the amount of computation for basic operations such as signing, verification, and the key update caused by revocation. Finally, using the linkability for dynamic membership, computation overhead in opening signer's identity can be significantly reduced or minimized.

1. Introduction

As various information communication technologies have been developed and deployed for real applications, it has become easier to collect and distribute a large amount of user information. Obviously, there are potential dangers associated with the high volume collection and distribution of user information, such as sensitive user information being unexpectedly collected and intentionally abused. To avoid such hazards in the face of the proliferation of user information, proper cryptographic mechanisms for protecting user privacy should be constructed.

Among cryptographic primitives for privacy, a group signature (GS) scheme has recently attracted increasing attention as a promising tool [6,30]. In contrast to conventional digital signature schemes where a signer can be publicly identified, a GS scheme allows a member to anonymously sign a message on behalf of a group without revealing his identity. A GS scheme

* Corresponding author.
E-mail addresses: videmot@etri.re.kr (J.Y. Hwang), junny@etri.re.kr (S. Lee), cbh@etri.re.kr (B.-H. Chung), hscho@etri.re.kr (H.S. Cho), nyang@inha.ac.kr (D. Nyang).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.07.065

provides a kind of *controllable anonymity* such that a signer can be disclosed from his signature under a master opening key that is secretly managed by a trusted authority. On the one hand, this feature imposes responsibility on the signer and thus prevents malicious behaviors, and on the other hand, it provides the appropriate confirmation for a signer when benefits are acquired, or as otherwise necessary.

However, in the model of a normal group signature, anonymity has been coarsely treated by hiding and revealing only identities. This treatment of anonymity seems excessively simplified and may not be suitable for diverse privacy applications. For example, consider that anonymity is applied to real world applications such as data mining, mileage services, and personalized services. A verifier or a service provider needs individual statistics on transactions, such as a consumer's buying pattern that are based on the "linkability" of the transactions, while preserving anonymity. Meanwhile, a signer may want to provide a specific verifier or a service provider with linking capability, while his signatures remain unlinked to other verifiers. As the controllable anonymity, the linkability also must be able to be controlled. In the conventional group signature schemes, verifiers are able to have this linkability by querying two signatures to an Opener who responds only to privileged verifiers. This imposes severe burden on the Opener.

Although linkability can be easily realized by using a pseudonym instead of a real identity in a public key certificate and by establishing an Opener to manage the association between a pseudonym and a real identity [29], this simple approach has some drawbacks. First of all, in the pseudonym system that has a single pseudonym and a single secret key, a user cannot avoid being linked by anyone who obtains his signatures. The disclosure of linkage information cannot be controlled either by the user or by an authority. Sometimes, a user or an authority might want to designate verifiers who have the right to link signatures. On the other hand, in the pseudonym system that has onetime pseudonym, where fresh pseudonym is used whenever a signer makes a signature, the problem of being linked by anyone is resolved, but it gives too much burden to pseudonym issuer for checking linkability. Furthermore, a signer must be issued a fresh pseudonym whenever he is about to generate a signature. Even if the signature is encrypted with some specific verifier's public key, the verifier can transfer the decrypted signatures to others who can see the linkability of signatures in an obvious way. Another drawback is that it is necessary to trust the Opener, and there is no way to validate whether the identity revealed by the Opener is really the owner of the pseudonym. In addition, when the linkability is required even after repeated joining and revocation, a verifier can find the history of membership status by keeping a list of revoked public key certificates of a pseudonym. This situation is undesirable for user privacy.

1.1. Our contributions

In order to resolve these problems, we consider a new useful property called *controllable linkability* (CL). Under this notion, a signer can generate signatures that look random and so hide his or her identity properly as normal group signatures while the signatures can be anonymously linked under a linking key secretly managed by a Linker. Since only linkage information can be revealed from the signatures with the linking key, we can establish a fine-grained control on anonymity by adding this CL to the controllable anonymity of a GS. A group signature supporting the CL is referred to here as a CL–GS. From a signature, the corresponding signer's identity can be revealed to an Opener who can access an "opening key." Our scheme provides an algorithm Judge for public validation of opening results. Thus, a validating entity of the opened identity does not need to trust the Opener.

In this paper, we construct a CL–GS scheme for dynamic membership which achieves the "efficient" CL. Our scheme achieves CL regardless of changes to the membership status of the signer and without exposure of the history of joining and revocation. By giving linking keys to designated Linkers, Opener's linking tasks can be distributed into the Linkers.

Owing to our trapdoor based approach to anonymity and also linkability, we can overcome all the drawbacks of the pseudonym system mentioned above. In our scheme the linkage information for signatures cannot be transferred to others without the Linker revealing the linking key, which the Linker is most unlikely to do. The exposure problem of membership status history can be solved in our scheme, because a user's secret key to be revoked and published is independent of the signer's linkage information.

To Linker, our scheme provides the same level of anonymity of the pseudonym-based schemes, because it can see the linkage information. Thus, we should assume that the verifier is not able to obtain any signature of which signer is known. Practically, this assumption is plausible considering that a user registers to a service provider anonymously from the beginning and she/he will use another link index (a kind of a pseudonym) in other service providers. The assumption is required also in pseudonym-based anonymous schemes. We stress, however, that to other than Linkers, our scheme provides stronger notion of anonymity than pseudonym-based schemes. Thus, our scheme can be regarded as a group signature scheme that efficiently and effectively implements an encrypted pseudonym for Linker, while not losing properties of the normal group signature scheme to others.

To capture the notion of CL more precisely, we present a formal definition of this security notion. Essentially, it should be guaranteed that a linking key cannot be used to open identities. This requirement is necessary to distinguish between the Opener and the Linker roles. In addition, it is required that an adversary should be unable to generate two signatures such that identities recovered from the signatures are the same but are proven to be "unlinked," or such that identities recovered from the signatures are different but are proven to be "linked." This requirement is critical to preventing an adversarial signer or colluding (conspiring) signers from generating false information about linkage. To reflect the linking capability that is possible through link queries, we slightly modified the security model of [9] to measure anonymity, traceability, non-frameability, and CL.

As a main building block for our construction, we introduce the Linear Combination Encryption (LCE) scheme that extends the well-known Linear Encryption scheme [6]. To show that the LCE scheme is semantically secure, we introduce a new computational problem in bilinear groups, called the Decision Linear Combination (DLC), and show that the difficulty of the problem can be guaranteed in generic bilinear groups. We also introduce a modified q Strong Diffie–Hellman (q-SDH$^+$) problem which is not easier than the original q Strong Diffie–Hellman problem [4,6]. In principle, our CL–GS scheme is based on a zero-knowledge proof of knowledge of a valid q-SDH$^+$ instance and the equality of two discrete logarithms. We prove that our scheme achieves all the security properties, i.e., anonymity, traceability, non-frameability, and CL, under the above assumptions.

Our CL–GS scheme works with any pairing map regardless of the type of the pairing map. Our scheme supporting CL yields a short signature which is only 33.3% longer than the best-known GS of [6] in the literature. More concretely, if a pairing map based on the MNT curve is used, where the order of a bilinear group associated with the pairing map is a 170-bit prime and the size of the relative base field is 171 bits, then our signature length is about 2044 bits or 256 bytes. Signing requires no pairing computation with pre-computation of parameters, and verifying requires a single pairing computation, as is also the case for the BBS scheme. Interestingly, due to the linkability for dynamic membership, computation overhead for opening can be significantly reduced or minimized, because the Opener can directly find a static signer's index that is set up for linkability. As shown in the comparison results that will be provided later, our scheme is comparable to [6] in terms of the amount of computation for basic operations such as signing, verification, and key updates caused by revocation.

A CL–GS can be used for various kinds of anonymity applications including not only known systems such as Vehicle Safety Communications (VSCs), Trusted Platform Modules (TPMs), and anonymous packet authentication for future internet systems, but also new kinds of services such as anonymously customized authentication (with a premium mileage service) and privacy-preserving data mining. Also, if we apply our scheme so that a signer should have a linking key, then a group signature scheme with a new functionality is established, where a signer is able to know whether a signature is belonged to him/her or not later. More specific applications with this functionality will be further studied in the future work.

1.2. Related works

The concept of a GS was introduced in [18]. Since that revolutionary work, group signatures have attracted attention due to their applicability in the various areas of privacy protection, and a number of GS schemes have been proposed [37,2,16,14,13,17,6,31,8,9,19,11,7,8].

The GS scheme of [2] is the first practical and coalition-resistant solution. To treat a dynamic group, some research has addressed efficient revocation problems [3,16]. The scheme in [10] provides a verifier-local revocation method where a verifier can check if a given signature has been generated by a revoked user. Formal definitions of security properties were well presented in [8] for static groups and in [31,9] for dynamic groups. Ref. [31] considered concurrent joining, which enables users to register simultaneously.

In [6], Boneh et al. proposed an efficient GS scheme that yields a short signature with bilinear pairings. The scheme was proven secure under the strong Diffie–Hellman and Decisional Linear assumptions. They also informally presented methods to achieve strong exculpability (or non-frameability) and revoke users by updating keys, realizing the revocation idea of [17]. Handling dynamic groups, [19] extended the BBS scheme to support concurrent joining and proved that the proposed scheme achieves stronger anonymity against chosen ciphertext attacks under the XDH assumption, which seems too strong in bilinear groups.

Recently, construction methods for a GS scheme without random oracles have been suggested [1,11,24,12,25]. As the GS schemes in [11,12] employ a symmetric bilinear map on groups of composite order, the signature size is large. The scheme of [1] does not achieve forward anonymity. The signature size in [24] is large for a practical deployment. In [25], a GS scheme is proposed to address some drawbacks of [24]. However, the above GS schemes are not sufficient to handle dynamic groups, particularly with revocation.

There have been other research efforts to design a more general framework of group signatures [15] and to utilize group signatures for wireless environments such as vehicular ad hoc networks [26,38].

1.3. Organization

The remainder of this paper is organized as follows. We present a security model for a CL–GS scheme in Section 2. We present cryptographic assumptions for the security of our scheme in Section 3 and an honest-verifier ZKPK protocol in Section 4. Using this protocol we propose a dynamic CL–GS scheme and formally prove that it is secure in Section 5, and we present efficient join and revocation methods in Section 6. We analyze the performance of our scheme in Section 7. Finally, we conclude in Section 8.

2. Model

We present a security model for a CL–GS scheme. In this model we consider three main authorities, the Issuer, the Opener, and the Linker, who have their independent privilege and a certain level of trust. The Issuer is an authority who manages and updates a group public key gpk when revocation occurs, and also takes the responsibility of issuing a secret signing key

$\mathtt{Link}(M_0, \sigma_0, M_1, \sigma_1)$
If $(M_0, \sigma_0) \in \mathbf{GSet}$ or $(M_1, \sigma_1) \in \mathbf{GSet}$ then return $\perp$;
Else return a bit $b \leftarrow \mathsf{Link}(gpk, mlk, M_0, \sigma_0, M_1, \sigma_1)$

Fig. 1. Link oracle.

Experiment $\mathsf{Exp}^{corr}_{\mathcal{GS},\mathcal{A}}(k)$ for Correctness
$(gpk,\ mik,\ mok,\ mlk) \leftarrow \mathsf{SetUp}(1^k)$; $\mathbf{CU} \leftarrow \emptyset$; $\mathbf{HU} \leftarrow \emptyset$;
$(i, M) \leftarrow \mathcal{A}(gpk : \mathtt{AddU}(\cdot), \mathtt{RReg}(\cdot))$;
If $i \notin \mathbf{HU}$ then return 0; If $\mathbf{usk}[i] = \varepsilon$ then return 0;
$\sigma \leftarrow \mathsf{GSig}(gpk, \mathbf{usk}[i], M)$;
If $\mathsf{GVf}(gpk, M, \sigma) = 0$, return 1;
$(j, \tau) \leftarrow \mathsf{Open}(gpk, mok, \mathbf{REG}, M, \sigma)$;
If $i \neq j$ then return 1;
If $\mathsf{Judge}(gpk,\ \mathbf{upk[i]},\ M,\ \sigma,\ \tau) = 0$ then return 1;
$(i, M_0, j, M_1) \leftarrow \mathcal{A}(gpk : \mathtt{AddU}(\cdot), \mathtt{RReg}(\cdot))$;
if $i = j$, then
  $\sigma_0 \leftarrow \mathsf{GSig}(gpk, \mathbf{usk}[i], M_0)$; $\sigma_1 \leftarrow \mathsf{GSig}(gpk, \mathbf{usk}[i], M_1)$;
  $b \leftarrow \mathsf{Link}(gpk, mlk, M_0, \sigma_0, M_1, \sigma_1)$;
  If $b = 0$ then return 1 else return 0;
Otherwise, i.e., $i \neq j$,
  $\sigma_0 \leftarrow \mathsf{GSig}(gpk, \mathbf{usk}[i], M_0)$; $\sigma_1 \leftarrow \mathsf{GSig}(gpk, \mathbf{usk}[j], M_1)$;
  $b \leftarrow \mathsf{Link}(gpk, mlk, M_0, \sigma_0, M_1, \sigma_1)$;
  If $b = 1$ then return 1 else return 0

Experiment $\mathsf{Exp}^{anon-b}_{\mathcal{GS},\mathcal{A}}(k)$ for Anonymity
$(gpk, mik, mok, mlk) \leftarrow \mathsf{SetUp}(1^k)$; $\mathbf{CU} \leftarrow \emptyset$; $\mathbf{HU} \leftarrow \emptyset$; $\mathbf{GSet} \leftarrow \emptyset$;
$(i_0, i_1, M) \leftarrow \mathcal{A}(gpk, mik : \mathtt{SndToU}, \mathtt{Link}, \mathtt{WReg}, \mathtt{USK}, \mathtt{CrptU})$;
$\sigma_{i_b} \leftarrow \mathtt{Ch}_b(i_0, i_1, M)$;
$d \leftarrow \mathcal{A}(gpk, mik, \sigma_{i_b} : \mathtt{SndToU}, \mathtt{Link}, \mathtt{WReg}, \mathtt{USK}, \mathtt{CrptU})$,
where $\mathcal{A}$ queried to neither $\mathtt{Link}(\sigma_{i_b}, M, \cdot, \cdot)$ nor $\mathtt{Link}(\cdot, \cdot, \sigma_{i_b}, M)$;
Return d

Experiment $\mathsf{Exp}^{trace}_{\mathcal{GS},\mathcal{A}}(k)$ for Traceability
$(gpk, mik, mok, mlk) \leftarrow \mathsf{SetUp}(1^k)$; $\mathbf{CU} \leftarrow \emptyset$; $\mathbf{HU} \leftarrow \emptyset$;
$(M, \sigma) \leftarrow \mathcal{A}(gpk, mok, mlk : \mathtt{SndToI}, \mathtt{AddU}, \mathtt{RReg}, \mathtt{USK}, \mathtt{CrptU})$;
If $\mathsf{GVfy}(gpk, M, \sigma) = 0$ then return 0;
$(i, \tau) \leftarrow \mathsf{Open}(gpk, mok, \mathbf{REG}, M, \sigma)$;
If $i = 0$ or $\mathsf{Judge}(gpk, i, \mathbf{upk}[i], M, \sigma, \tau) = 0$ then return 1, else return 0

Experiment $\mathsf{Exp}^{nf}_{\mathcal{GS},\mathcal{A}}(k)$ for Non-Frameability
$(gpk, mik, mok, mlk) \leftarrow \mathsf{SetUp}(1^k)$; $\mathbf{CU} \leftarrow \emptyset$; $\mathbf{HU} \leftarrow \emptyset$;
$(M, \sigma, i) \leftarrow \mathcal{A}(gpk, mik, mok, mlk : \mathtt{SndToU}, \mathtt{WReg}, \mathtt{GSig}, \mathtt{USK}, \mathtt{CrptU})$;
If $\mathsf{GVfy}(gpk, M, \sigma) = 0$ then return 0;
If the following conditions are all true then return 1, else return 0:
- $i \in \mathbf{HU}$ and $\mathbf{gsk}[i] \neq \varepsilon$;
- $(i, \tau) \leftarrow \mathsf{Open}(gpk, mok, \mathbf{REG}, M, \sigma)$;
- $\mathsf{Judge}(gpk, i, \mathbf{upk}[i], M, \sigma, \tau) = 1$;
- $\mathcal{A}$ did not query $\mathsf{USK}(i)$ or $\mathsf{GSig}(i, M)$

Fig. 2. Experiments for correctness, anonymity, traceability, and non-frameability.

usk[i] to a joining user by using the master issuing key *mik*. The Opener is an authority who can use the master opening key *mok* to open a signature and generate a proof to reveal the corresponding signer. The Linker is an authority who can use the master linking key *mlk* to "link" signatures, i.e., to discern whether those signatures are from the same user. Note that the Opener can do all of the Linker's tasks by recovering and comparing identities from signatures. In contrast, the Linker is only allowed to confirm linkage information about signatures, not signers' identities. A legal user can anonymously sign a message using its signing key **usk[i]**. A verifier can confirm only the validity of a signature without knowledge of its corresponding signer.

A CL–GS scheme consists of the following algorithms.

- SetUp: On input a security parameter 1^k, it outputs a group public key *gpk* and private master keys *mik*, *mok*, and *mlk*.
- (UserJoin, Issue): UserJoin with index *i* and a secret key corresponding to a user's public key **upk**[*i*], and Issue with *mik* run an interactive protocol. After completion of this protocol, UserJoin generates a full user signing key **usk**[*i*] corresponding to **upk[i]**. Issue updates or adds the corresponding user's record in **REG**.
- GSig: On input *gpk*, **usk**[*i*], and a message $M \in \{0,1\}^*$, it outputs a signature σ.
- GVfy: On input *gpk* and (σ,*M*), it outputs 1 (valid) or 0 (invalid).

– Open: On input *gpk*, *mok*, **REG**, and (σ, M), it outputs a signer's public key **upk**[*i*] and a proof τ.
– Judge: On input *gpk*, (σ, M), **upk**[*i*], and τ, it outputs 1 (meaning user *i* generated σ) or 0.
– Link: On input *gpk*, *mlk*, (M_0, σ_0), (M_1, σ_1), it outputs 1, meaning the signatures are generated by a user, or 0.l.

2.1. Security notions

We define security notions for a CL–GS scheme by modifying those of a GS scheme in [9] to reflect CL. Denote by **HU** and **CU** the lists of honest and corrupted users, respectively. **GSet** is the set of message–signature pairs. **REG** is the list of transcripts generated in the join process of users. It is managed by Issuer, and shared only with Opener. $\mathbf{REG}_i$ is the user *i*'s field on the list and **usk**[**i**] is user *i*'s signing key.

The Oracles: The following oracles are used by an adversary in security experiments (see Figs. 2 and 3).

– AddU(i): *add user* oracle. By running UserJoin and Issue algorithms, user *i* is added to the group as a member of the **HU** list, given **usk**[**i**] on this oracle.
– CrptU(i, upk): *corrupt user* oracle. A new user *i* with his identifiable information, **upk**[**i**] is added to **CU**.
– SndToI(i, M): *send to issuer* oracle. A corrupted user *i* can run UserJoin with an honest Issuer to send *M*.
– SndToU(i, M): *send to user* oracle. A corrupted Issuer can run Issue with an honest user to send a message *M*.
– USK(i): *user secret key* oracle. **usk**[**i**] is revealed and an honest user *i* is turned into a corrupted user.
– RReg(i): *read registration table* oracle. This oracle gives a read access to $\mathbf{REG}_i$.
– WReg(i, ρ): *write registration table* oracle. This permits a write/modify access to $\mathbf{REG}_i$ to write ρ into $\mathbf{REG}_i$.
– GSig(i, M): *group signing* oracle. This oracle gives the group signature of an honest user *i* for a message *m*.
– $\mathrm{Ch}_b(i_0, i_1, M)$: *challenge* oracle. For given values of i_0, i_1, *M*, the oracle outputs i_b's signature σ_{i_b} for a random bit *b* and records (σ_{i_b}, M) in the **GSet**.
– Open(M, σ): *opening* oracle. This oracle runs an Open algorithm with (σ, m), not generated by the $\mathrm{Ch}_b(\cdot,\cdot,\cdot)$ oracle, and returns the corresponding signer's identity.
– Link($M_0, \sigma_0, M_1, \sigma_1$): *linking* oracle. This oracle runs the Link algorithm to check whether the two signatures are generated by a user. Return the output bit of Link. Refer to Fig. 1.

Security Notions: We present correctness and anonymity, traceability, non-frameability, and linkability.

Correctness: In a CL–GS scheme $\mathcal{GS}$, signatures generated by an honest user should be verified correctly. With inputs of a message and a signature, Open should correctly identify the signer. Link should link the signatures from a signer. To any adversary $\mathcal{A}$ and $k \in \mathbb{N}$, we define the experiment $\mathrm{Exp}^{corr}_{\mathcal{GS},\mathcal{A}}(k)$ in Fig. 2 and $\mathrm{Adv}^{corr}_{\mathcal{GS},\mathcal{A}}(k) = \Pr\left[\mathrm{Exp}^{corr}_{\mathcal{GS},\mathcal{A}}(k) = 1\right]$, and say that $\mathcal{GS}$ is *correct* if $\mathrm{Adv}^{corr}_{\mathcal{GS},\mathcal{A}}(k) = 0$.

Anonymity: We consider a relaxed version of CCA (Chosen Ciphertext Attack)-anonymity [9], i.e., CPA (Chosen Plaintext Attack)-anonymity [6]. Given a signature σ generated by one of two users chosen by the adversary, it should be hard to guess the real generator of σ. To any adversary $\mathcal{A}$, a bit $b \in \{0,1\}$ and any $k \in \mathbb{N}$, we define $\mathrm{Exp}^{anon-b}_{\mathcal{GS},\mathcal{A}}(k)$ in Fig. 2 and $\mathrm{Adv}^{anon}_{\mathcal{GS},\mathcal{A}}(k) = \left|\Pr\left[\mathrm{Exp}^{anon-1}_{\mathcal{GS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathrm{Exp}^{anon-0}_{\mathcal{GS},\mathcal{A}}(k) = 1\right]\right|$, and say that $\mathcal{GS}$ is *CPA-anonymous* if $\mathrm{Adv}^{anon}_{\mathcal{GS},\mathcal{A}}(\cdot)$ is negligible.

Traceability: Given a valid signature σ, it should be impossible that the origin of the signature cannot be identified or that a (public) proof to check the origination cannot be generated. To any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we define the experiment $\mathrm{Exp}^{trace}_{\mathcal{GS},\mathcal{A}}(k)$ in Fig. 2 and $\mathrm{Adv}^{trace}_{\mathcal{GS},\mathcal{A}}(k) = \Pr[\mathrm{Exp}^{trace}_{\mathcal{GS},\mathcal{A}}(k) = 1]$ and say that $\mathcal{GS}$ is *traceable* if $\mathrm{Adv}^{trace}_{\mathcal{GS},\mathcal{A}}(\cdot)$ is negligible.

Non-frameability: It should be impossible that colluding users and even trusted authorities forge a signature of a user without the secret key of the user. To any adversary $\mathcal{A}$ and $k \in \mathbb{N}$, we define the experiment $\mathrm{Exp}^{nf}_{\mathcal{GS},\mathcal{A}}(k)$ in Fig. 2, and $\mathrm{Adv}^{nf}_{\mathcal{GS},\mathcal{A}}(k) = \Pr\left[\mathrm{Exp}^{nf}_{\mathcal{GS},\mathcal{A}}(k) = 1\right]$. We say that $\mathcal{GS}$ is *non-frameable* if $\mathrm{Adv}^{nf}_{\mathcal{GS},\mathcal{A}}(\cdot)$ is negligible for any PPT $\mathcal{A}$.

Linkability: We consider three security notions for linkability, called *LO-linkability* (Link-Only linkability), *JP-Unforgeability* (Judge-Proof Unforgeability), and *E-linkability* (Enforced linkability). LO-linkability and JP-Unforgeability strictly limit the Linker's role by not allowing any opening functionality such as identifying the signer and generating a proof for Judge.

- LO-linkability captures that a linking key should be used only for linking signatures, not for gaining useful information for opening. Formally, to any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we define the experiment $\mathrm{Exp}^{LO-link}_{\mathcal{GS},\mathcal{A}}(k)$ in Fig. 3 and $\mathrm{Adv}^{LO-link}_{\mathcal{GS},\mathcal{A}}(k) = \Pr\left[\mathrm{Exp}^{LO-link}_{\mathcal{GS},\mathcal{A}}(k) = 1\right]$, and say that $\mathcal{GS}$ is *LO-linkable* if $\mathrm{Adv}^{LO-link}_{\mathcal{GS},\mathcal{A}}(k)$ is negligible. Here, the adversary's capability is restricted such that it should query to the signing oracle without designating the two target identities that he chose. In the game, we may allow the adversary to query with any identities except the target identities i_0, i_1 to the signing oracle GSig, though this allowance is not explicitly described. This restriction excludes the trivial opening by linking trials with signatures of which owner he knows. In practice, this restricted signing oracle access means that a signer gives linker more trust than a normal verifier but less than an opener. A linker is assumed to behave honestly but curiously, so it will try to find passively user's identity only with group signatures collected.

Experiment $\mathsf{Exp}^{E-link}_{\mathcal{GS},\mathcal{A}}(k)$ for E-Linkability

$(gpk, mik, mok, mlk) \leftarrow \mathsf{SetUp}(1^k)$;
$\mathbf{CU} \leftarrow \emptyset$; $\mathbf{HU} \leftarrow \emptyset$; $\mathbf{GSet} \leftarrow \emptyset$;
$(M_0, \sigma_0, M_1, \sigma_1) \leftarrow \mathcal{A}(gpk, mok, mlk : \mathtt{SndToU}, \mathtt{AddU}, \mathtt{RReg}, \mathtt{GSig}, \mathtt{USK}, \mathtt{CrptU})$;
If $\mathsf{GVfy}(gpk, M_b, \sigma_b) = 0$ for $b = 0$ or 1 then return 0;
$(i_0, \tau_{i_0}) \leftarrow \mathsf{Open}(gpk, mok, \mathbf{REG}, M_0, \sigma_0)$;
$(i_1, \tau_{i_1}) \leftarrow \mathsf{Open}(gpk, mok, \mathbf{REG}, M_1, \sigma_1)$;
If $\mathsf{Judge}(gpk, \mathbf{upk[i]}, M_0, \sigma_0, \tau_{i_0}) = 0$ or $\mathsf{Judge}(gpk, \mathbf{upk[j]}, M_1, \sigma_1, \tau_{i_1}) = 0$ then return 0;
If $i_0 \neq i_1$ and $1 = \mathsf{Link}(gpk, mlk, M_0, \sigma_0, M_1, \sigma_1)$ then return 1;
else if $i_0 = i_1$ and $0 = \mathsf{Link}(gpk, mlk, M_0, \sigma_0, M_1, \sigma_1)$ then return 1;
else return 0

Experiment $\mathsf{Exp}^{LO-link}_{\mathcal{GS},\mathcal{A}}(k)$ for LO-Linkability

$(gpk, mik, mok, mlk) \leftarrow \mathsf{SetUp}(1^k)$;
$\mathbf{CU} \leftarrow \emptyset$; $\mathbf{HU} \leftarrow \emptyset$; $\mathbf{GSet} \leftarrow \emptyset$;
$(i_0, i_1, M) \leftarrow \mathcal{A}(gpk, mlk : \mathtt{SndToU}, \mathtt{AddU}, \mathtt{GSig}, \mathtt{Open}, \mathtt{USK}, \mathtt{CrptU})$;
$b \leftarrow_R \{0,1\}$; $\sigma \leftarrow \mathsf{GSig}(i_b, M)$;
$b' \leftarrow \mathcal{A}(gpk, mlk, (i_0, i_1, M, \sigma) : \mathtt{SndToU}, \mathtt{AddU}, \mathtt{GSig}, \mathtt{Open}, \mathtt{USK}, \mathtt{CrptU})$;
If the following conditions are all true then return 1, else return 0;
- $i_0, i_1 \in \mathbf{HU}$;
- $\mathtt{GSig}(i_b, \cdot)$ and $\mathtt{Open}(M, \sigma)$ have not been queried;
- $b' = b$

Experiment $\mathsf{Exp}^{JP-UF}_{\mathcal{GS},\mathcal{A}}(k)$ for JudgeProof-Unforgeability

$(gpk, mik, mok, mlk) \leftarrow \mathsf{SetUp}(1^k)$;
$\mathbf{CU} \leftarrow \emptyset$; $\mathbf{HU} \leftarrow \emptyset$; $\mathbf{GSet} \leftarrow \emptyset$;
$(i, M) \leftarrow \mathcal{A}(gpk, mik, mlk : \mathtt{SndToU}, \mathtt{WReg}, \mathtt{GSig}, \mathtt{Open}, \mathtt{USK}, \mathtt{CrptU})$;
$\sigma \leftarrow \mathsf{GSig}(i, M)$;
$\tau \leftarrow \mathcal{A}(gpk, mik, mlk, (i, M, \sigma) : \mathtt{SndToU}, \mathtt{WReg}, \mathtt{GSig}, \mathtt{Open}, \mathtt{USK}, \mathtt{CrptU})$;
If the following conditions are all true then return 1, else return 0;
- $i \in \mathbf{HU}$ and $\mathbf{gsk}[i] \neq \varepsilon$;
- $\mathtt{Open}(M, \sigma)$ has not been queried;
- $\mathsf{Judge}(gpk, \mathbf{upk}[i], M, \sigma, \tau) = 1$

Fig. 3. Security experiments for linkability.

- JP-Unforgeability captures that a linking key cannot be used for generating a Judge proof, which completes the representation of Linker's restricted capability. Formally, to any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we define the experiment $\mathsf{Exp}^{JP-UF}_{\mathcal{GS},\mathcal{A}}(k)$ in Fig. 3 and $\mathsf{Adv}^{JP-UF}_{\mathcal{GS},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{JP-UF}_{\mathcal{GS},\mathcal{A}}(k) = 1]$, and say that $\mathcal{GS}$ is *JP-Unforgeable* if $\mathsf{Adv}^{JP-UF}_{\mathcal{GS},\mathcal{A}}(k)$ is negligible.
- E-linkability captures that colluding users should not be able to generate two pairs of a message and a signature satisfying any of the following conditions,even with help of authorities such as the Linker or Opener: (1) Identities recovered from the two valid signatures by Open are the same, and also they are successfully judged, while Link outputs 0 (“unlinked”). (2) Identities recovered from the two valid signatures by Open are different and they are successfully judged, while Link outputs 1 (“linked”). Formally, to any adversary $\mathcal{A}$, $k \in \mathbb{N}$, we define the experiment $\mathsf{Exp}^{E-link}_{\mathcal{GS},\mathcal{A}}(k)$ in Fig. 3 and $\mathsf{Adv}^{E-link}_{\mathcal{GS},\mathcal{A}}(k) = \Pr\left[\mathsf{Exp}^{E-link}_{\mathcal{GS},\mathcal{A}}(k) = 1\right]$, and say that $\mathcal{GS}$ is *E-linkable* if $\mathsf{Adv}^{E-link}_{\mathcal{GS},\mathcal{A}}(\cdot)$ is negligible.

3. Preliminaries

We now review bilinear maps and computational assumptions on which our scheme relies. Let $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$ for a prime number p. Denote by $s \xleftarrow{R} S$ the operation that picks an element s of a set S uniformly at random.

3.1. Bilinear maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be multiplicative groups of prime order p and $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map which satisfies the following properties: (1) Bilinear: $\mathbf{e}(g^a, h^b) = \mathbf{e}(g,h)^{ab}$ for all $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$. (2) Non-degenerate: There exists $h \in \mathbb{G}_1$ such that $\mathbf{e}(g,h) \neq 1$. (3) Computable: There exists an efficient algorithm to compute $e(g', h')$ for all $g' \in \mathbb{G}_1$ and $h' \in \mathbb{G}_2$.

As reported in the literature [21,34], the bilinear map can be realized on Abelian varieties using variants of the Weil or Tate pairing. These pairings can be efficiently computed on algebraic curves using the Miller algorithm. For a concrete instantiation, $\mathbb{G}_1$ and $\mathbb{G}_2$ are defined to be subgroups of groups, which are defined by a elliptic curve over a finite field $\mathbb{F}$ or in an extension of $\mathbb{F}$. Recently, more efficient variants of the Tate pairing such as “Eta” and “Ate” have been proposed [33].

3.2. Computational assumptions

The Decision Diffie–Hellman (DDH) Problem: The DDH problem is to decide $c \stackrel{?}{=} ab$ when a multiplicative group $\mathbb{G}$ of prime order p, a generator g of $\mathbb{G}$, and (g^a, g^b) for random numbers $a, b \in \mathbb{Z}_p^*$, and g^c are given.

The q-Strong Diffie–Hellman (q-SDH) Problem [4,6]: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative groups of prime order p and g_1 and h_1 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The q-SDH problem is to compute a pair $\left(g_1^{1/(\gamma+x)}, x\right)$ for a given $(q+2)$-tuple $\left(g_1, g_1^{\gamma}, \ldots, g_1^{\gamma^q}, h_1, h_1^{\gamma}\right)$ where $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. We say that the *q-SDH assumption* holds if any PPT adversary $\mathcal{A}$ has negligible advantage in solving the q-SDH problem.

The Modified q-Strong Diffie–Hellman (q-SDH$^+$) Problem: We introduce a modified q-SDH problem, called q-SDH$^+$, which is not easier than the q-SDH problem. Let g_1, g_2, and g_3 be random generators of $\mathbb{G}_1$ and h_1 be a random generator of $\mathbb{G}_2$. The q-SDH$^+$ problem is to compute a pair $\left((g_1 g_2^y g_3^z)^{1/(\gamma+x)}, (x,y,z)\right)$ for a given $(q+4)$-tuple $\left(g_1, g_2, g_3, g_1^{\gamma}, \ldots, g_1^{\gamma^q}, h_1, h_1^{\gamma}\right)$ where $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. We can show that this q-SDH$^+$ problem is efficiently reduced to the q-SDH problem as follows: For a q-SDH instance, a q-SDH solver generates a q-SDH$^+$ instance I by selecting $\alpha, \beta \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and adding $g_2 = g_1^{\alpha}$ and $g_3 = g_1^{\beta}$ to the given q-SDH instance. If the q-SDH$^+$ solver gets I and finally outputs (A,x,y,z), then the q-SDH solver outputs $(T = A^{(1+\alpha y+\beta z)^{-1}}, x)$ using (A,x,y,z). Obviously, (T,x) is a correct solution for the q-SDH problem, because $T = A^{(1+\alpha y+\beta z)^{-1}} = \left((g_1 g_2^y g_3^z)^{\frac{1}{\gamma+x}}\right)^{(1+\alpha y+\beta z)^{-1}} = g_1^{\frac{1}{\gamma+x}}$.

The Decision Linear Combination (DLC) Problem: A DLC parameter generator $\mathcal{IG}_{DLC}$ is a PPT algorithm that takes 1^{λ} as input security parameter, runs in polynomial time, and outputs a multiplicative group $\mathbb{G}_1$ of prime order p. The DLC problem is to decide if $w_1^a w_2^b \stackrel{?}{=} T$ for a given tuple $(u, v, w_1, w_2, u^a, v^b, T)$, where $a, b \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and $u, v, w_1, w_2, T \in \mathbb{G}_1$. Define the advantage of $\mathcal{A}$ with respect to $\mathcal{IG}_{DLC}$ by

$$\varepsilon_{\mathrm{DLC}} = \left|\Pr\left[\mathcal{A}(u, v, w_1, w_2, u^a, v^b, w_1^a w_2^b) = 1 : u, v, w_1, w_2 \stackrel{R}{\leftarrow} \mathbb{G}_1, a, b \stackrel{R}{\leftarrow} \mathbb{Z}_p^*\right] - \Pr\left[\mathcal{A}(u, v, w_1, w_2, u^a, v^b, \eta) = 1 : u, v, w_1, w_2, \eta \stackrel{R}{\leftarrow} \mathbb{G}_1, a, b \stackrel{R}{\leftarrow} \mathbb{Z}_p^*\right]\right|$$

$\mathcal{IG}_{DLC}$ is said to satisfy the *DLC assumption* if any PPT adversary $\mathcal{A}$ has a negligible advantage in solving the DLC problem.

We note that the DLC problem is a kind of double DDH problem. Decision Linear (DL) problem [6] is a special case of the DLC problem, that is, when $w_1 = w_2$. Using an idea similar to that of [6], we can show that the DLC assumption holds in generic bilinear groups. Refer to Appendix 8.

3.3. Linear Combination Encryption (LCE)

The Linear Combination Encryption (LCE) scheme is an extension of LE or ElGamal encryption, which is secure in groups where the DDH problem is easy but the CDH problem is hard. LCE can be viewed as a kind of double ElGamal encryption that performs ElGamal encryption twice sequentially.[1] In the LCE scheme, a user's public and secret keys are defined as $pk = (u, v, w_1 = u^x, w_2 = v^y)$ and $sk = (x,y)$, where $u, v \stackrel{R}{\leftarrow} \mathbb{G}_1$ and $x, y \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. Message M is encrypted to $(D_1 = u^a, D_2 = v^b, D_3 = M \cdot w_1^a w_2^b)$, using random $a, b \in \mathbb{Z}_p^*$. The original message M can be decrypted from the ciphertext (D_1, D_2, D_3) by $D_3 \cdot (D_1^x \cdot D_2^y)^{-1}$. It is obvious that LCE is *semantically secure* against chosen plaintext attacks (CPA), under the DLC assumption.

Next, we extend LCE and introduce Hybrid LCE (HLCE), which is crucially used for our construction. The HLCE scheme is described as follows:

- KeyGen: It picks $u, v \stackrel{R}{\leftarrow} \mathbb{G}_1$, $x_1, y_1, x_2, y_2 \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and then outputs a public key $pk = (u, v, w_1 = u^{x_1}, w_2 = v^{y_1}, d_1 = u^{x_2}, d_2 = v^{y_2})$ and its corresponding secret key (x_1, y_1, x_2, y_2).
- Enc: Given a public key pk and a message $M = (M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_1$, it picks $a, b \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and outputs a ciphertext $C = \left(D_1 = u^a, D_2 = v^b, D_3 = M_1 w_1^a w_2^b, D_4 = M_2 d_1^a d_2^b\right)$.
- Dec: Given a ciphertext $C = (D_1, D_2, D_3, D_4)$, compute a plaintext $M = (M_1, M_2)$ by $M_1 = D_3 \left(D_1^{x_1} D_2^{y_1}\right)^{-1}$ and $M_2 = D_4 \left(D_1^{x_2} D_2^{y_2}\right)^{-1}$.

We can show that the HLCE scheme is semantically secure against CPA under the DLC assumption via a standard hybrid argument.

4. ZKPK protocol for SDH$^+$

Before describing our CL–GS scheme in detail, we first introduce an honest-verifier zero-knowledge proof of knowledge (ZKPK) protocol for proving possession of an SDH$^+$ tuple. The public parameters for this protocol are $(\mathbf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, g_1, g_2, g_3, h_1, h_\theta, w_1, w_2, d_1, d_2, u, v)$, where $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing, and $w_1, w_2, d_1, d_2, u, v, g, g_1, g_2, g_3 \stackrel{R}{\leftarrow} \mathbb{G}_1$, $h_1 \stackrel{R}{\leftarrow} \mathbb{G}_2$ and $h_\theta = h_1^\theta$ for random $\theta \in \mathbb{Z}_p^*$. The protocol has two players, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Let $Z = g_2^y$ and $\widehat{Z} = g^y$ for some $y \in \mathbb{Z}_p^*$. Assume that the prover has a tuple (A,x,y,z) satisfying $A^{x+\theta} = g_1 Z^{-1} g_3^{-z}$ and $\log_{g_2} Z = \log_g \widehat{Z}$ where $A \in \mathbb{G}_1$ and $x, y, z \in \mathbb{Z}_p^*$. The protocol proves possession of the tuple (A,x,y,z). The concrete protocol is described as follows:

[1] We note that our notion is different from that of Double ElGamal Encryption introduced for achieving IND-CCA security in [20,28].

- (Prover) First, $\mathcal{P}$ picks $\alpha,\beta \xleftarrow{R} \mathbb{Z}_p^*$, and computes $D_1 = u^{\alpha}, D_2 = v^{\beta}, D_3 = Aw_1^{\alpha}w_2^{\beta}, D_4 = g^{y}d_1^{\alpha}d_2^{\beta} \in \mathbb{G}_1$, and $\gamma = x\alpha$, $\delta = x\beta \bmod p$. It also picks $r_{\alpha}, r_{\beta}, r_x, r_y, r_z, r_{\gamma}, r_{\delta} \xleftarrow{R} \mathbb{Z}_p^*$ and computes $R_1 = u^{r_{\alpha}}, R_2 = v^{r_{\beta}}$, $R_3 = \mathbf{e}(D_3,h_1)^{r_x}\mathbf{e}(w_1,h_0)^{-r_{\alpha}}\mathbf{e}(w_2,h_0)^{-r_{\beta}}\mathbf{e}(w_1,h_1)^{-r_{\gamma}}\mathbf{e}(w_2,h_1)^{-r_{\delta}}\mathbf{e}(g_2,h_1)^{r_y}\mathbf{e}(g_3,h_1)^{r_z}, R_4 = g^{r_y}d_1^{r_{\alpha}}d_2^{r_{\beta}}, R_5 = D_1^{r_x}u^{-r_{\gamma}}, R_6 = D_2^{r_x}v^{-r_{\delta}}$. Then, $(D_1,\ldots,D_4,R_1,\ldots,R_6)$ is sent to $\mathcal{V}$.
- (Verifier) $\mathcal{V}$ sends a random challenge value $c \xleftarrow{R} \mathbb{Z}_p^*$.
- (Prover) $\mathcal{P}$ computes $s_{\alpha} = r_{\alpha} + c\alpha$, $s_{\beta} = r_{\beta} + c\beta$, $s_x = r_x + cx$, $s_y = r_y + cy$, $s_z = r_z + cz$, $s_{\gamma} = r_{\gamma} + c\gamma$, $s_{\delta} = r_{\delta} + c\delta (\bmod p)$, and then sends $(s_{\alpha},s_{\beta},s_{\gamma},s_{\delta},s_x,s_y,s_z)$ to $\mathcal{V}$.
- (Verifier) $\mathcal{V}$ checks if (1) $R_1 \stackrel{?}{=} u^{s_{\alpha}}D_1^{-c}$, (2) $R_2 \stackrel{?}{=} v^{s_{\beta}} \cdot D_2^{-c}$, (3) $R_3 \stackrel{?}{=} \mathbf{e}(D_3,h_1)^{s_x}\mathbf{e}\left(w_1^{-s_{\alpha}}w_2^{-s_{\beta}},h_0\right)\mathbf{e}\left(w_1^{-s_{\gamma}}w_2^{-s_{\delta}},h_1\right)\mathbf{e}(g_2,h_1)^{s_y}\mathbf{e}(g_3,h_1)^{s_z}(\mathbf{e}(D_3,\ h_0)/\mathbf{e}(g_1,h_1))^{c}$, (4) $R_4 \stackrel{?}{=} g^{s_y}d_1^{s_{\alpha}}d_2^{s_{\beta}}D_4^{-c}$, (5) $R_5 \stackrel{?}{=} D_1^{s_x} \cdot u^{-s_{\gamma}}$, (6) $R_6 \stackrel{?}{=} D_2^{s_x} \cdot v^{-s_{\delta}}$. If all the six equations hold then output 1 (valid) and otherwise, 0 (invalid).

Using Lemmas 1–3, we can directly show that the above protocol is an honest-verifier ZKPK of an SDH$^+$ tuple.

Lemma 1. *The above protocol is complete.*

Proof. Suppose that a prover possess a valid SDH$^+$ tuple (A,x,y,z) and honestly executes the protocol. The following six relations hold: (1) $R_1 = u^{r_{\alpha}} = u^{r_{\alpha}}u^{c\alpha}(u^{\alpha})^{-c} = u^{s_{\alpha}}D_1^{-c}$, (2) $R_2 = v^{r_{\beta}} = v^{r_{\beta}}v^{c\beta}(v^{\beta})^{-c} = v^{s_{\beta}}D_2^{-c}$, (3) $R_3 = \mathbf{e}(D_3,h_1)^{r_x}\mathbf{e}(w_1,h_0)^{-r_{\alpha}}\mathbf{e}(w_2,h_0)^{-r_{\beta}}\mathbf{e}(w_1,h_1)^{-r_{\gamma}}\mathbf{e}(w_2,h_1)^{-r_{\delta}}\mathbf{e}(g_2,h_1)^{r_y}\mathbf{e}(g_3,h_1)^{r_z} = \mathbf{e}(D_3,h_1)^{cx}\mathbf{e}(w_1^{-\alpha}w_2^{-\beta},h_0h_1^{x})^{c}\mathbf{e}(g_2,h_1)^{cy}\mathbf{e}(g_3,h_1)^{cz}(\mathbf{e}(D_3,h_0)/\mathbf{e}(g_1,h_1))^{c}\mathbf{e}(D_3,h_1)^{r_x}\mathbf{e}(w_1,h_0)^{-r_{\alpha}}\mathbf{e}(w_2,h_0)^{-r_{\beta}}\mathbf{e}(w_1,h_1)^{-r_{\gamma}}\mathbf{e}(w_2,h_1)^{-r_{\delta}}\mathbf{e}(g_2,h_1)^{r_y}\mathbf{e}(g_3,h_1)^{r_z} = \mathbf{e}(D_3,h_1)_{c}^{s_x}\mathbf{e}(w_1,h_0)^{-s_{\alpha}}\mathbf{e}(w_2,h_0)^{-s_{\beta}}\mathbf{e}(w_1,h_1)^{-s_{\gamma}}\mathbf{e}(w_2,h_1)^{-s_{\delta}}\mathbf{e}(g_2,h_1)^{s_y}\mathbf{e}(g_3,h_1)^{s_z}(\mathbf{e}(D_3,h_0)/\mathbf{e}(g_1,h_1))^{c}$,(4) $R_4 = g^{r_y}d_1^{r_{\alpha}}d_2^{r_{\beta}} = g^{r_y}d_1^{r_{\alpha}}d_2^{r_{\beta}}\left(g^{y}d_1^{\alpha}d_2^{\beta}\right)^{c}\left(g^{y}d_1^{\alpha}d_2^{\beta}\right)^{-c} = g^{s_y}d_1^{s_{\alpha}}d_2^{s_{\beta}}D_4^{-c}$, (5) $R_5 = D_1^{r_x} \cdot u^{-r_{\gamma}} = D_1^{r_x} \cdot u^{-r_{\gamma}}(u^{\alpha})^{cx} \cdot u^{-c\alpha x} = D_1^{r_x} \cdot u^{-r_{\gamma}}D_1^{cx} \cdot u^{-c\gamma} = D_1^{s_x} \cdot u^{-s_{\gamma}}$, (6)$R_6 = D_2^{r_x} \cdot v^{-r_{\delta}} = D_2^{r_x} \cdot v^{-r_{\delta}}(v^{\beta})^{cx} \cdot v^{-c\alpha x} = D_2^{r_x} \cdot v^{-r_{\delta}}D_1^{cx} \cdot v^{-c\gamma} = D_2^{s_x} \cdot v^{-s_{\delta}}$.

Thus, all equations which the verifier checks hold and all transcripts are always accepted. □

Lemma 2. *The above protocol is sound.*

Proof. We show that there exists an extractor for the protocol. Suppose that an extractor can find a forking point in the protocol, where the point is just before the prover receives a challenge c. At the first run of the protocol, the prover sends $D_1, \ldots, D_4$ and $R_1, \ldots, R_6$ and then, to a challenge value c, the prover answers with $s_{\alpha}, s_{\beta}, s_x, s_y, s_z, s_{\gamma}, s_{\delta}$. To another challenge value $c' \neq c$, the prover answers with $s'_{\alpha},\ s'_{\beta},\ s'_x,\ s'_y,\ s'_z,\ s'_{\gamma}$, and s'_{δ}. If the prover passes the test, all six Eqs. (1)–(6) in the final verification step hold for each set of values. Let $\Delta c = c - c'$, $\Delta s_{\alpha} = s_{\alpha} - s'_{\alpha}$, and similarly for Δs_{β}, Δs_x, Δs_y, Δs_z, Δs_{γ}, and Δs_{δ}. Dividing the two instances of Eq. (1) (one instance using c and the other using c'), we have $u^{\Delta s_{\alpha}} = D_1^{\Delta c}$, i.e., $u^{\tilde{\alpha}} = D_1$ where $\tilde{\alpha} = \Delta s_{\alpha} \cdot (\Delta c)^{-1} (\bmod p)$. Similarly, from Eq. (2), we have $v^{\tilde{\beta}} = D_2$ where $\tilde{\beta} = \Delta s_{\beta} \cdot (\Delta c)^{-1} (\bmod p)$. Consider Eq. (4). Dividing the two instances of this equation gives $D_4^{\Delta c} = g^{\Delta s_y}d_1^{\Delta s_{\alpha}}d_2^{\Delta s_{\beta}}$ and so $D_4 = \left(g^{\Delta s_y}d_1^{\Delta s_{\alpha}}d_2^{\Delta s_{\beta}}\right)^{(\Delta c)^{-1}}$. We obtain $\tilde{y} = \Delta s_y(\Delta c)^{-1} (\bmod p)$ such that $D_4 = g^{\tilde{y}}d_1^{\tilde{\alpha}}d_2^{\tilde{\beta}}$. Next, consider Eq. (5). Dividing the two instances gives $D_1^{\Delta s_x} = u^{\Delta s_{\gamma}}$. Substituting $D_1 = u^{\tilde{\alpha}}$ gives $u^{\tilde{\alpha}\Delta s_x} = u^{\Delta s_{\gamma}}$, i.e., $\Delta s_{\gamma} = \tilde{\alpha}\Delta s_x (\bmod p)$. Similarly, from Eq. (6) we can deduce that $\Delta s_{\delta} = \tilde{\beta}\Delta s_x (\bmod p)$. Finally, dividing the two instances from Eq. (3), we obtain $(\mathbf{e}(g_1,h_1)/\mathbf{e}(D_3,h_0))^{\Delta c} = \mathbf{e}(D_3,h_1\ )^{\Delta s_x}\mathbf{e}(w_1^{-\Delta s_{\alpha}}w_2^{-\Delta s_{\beta}},h_0)\mathbf{e}(w_1^{-\Delta s_{\gamma}}w_2^{-\Delta s_{\delta}},h_1)\mathbf{e}(g_2,h_1)^{\Delta s_y}\mathbf{e}(g_3,h_1)^{\Delta s_z} = \mathbf{e}(D_3,h_1)^{\Delta s_x}\mathbf{e}(w_1^{-\Delta s_{\alpha}}w_2^{-\Delta s_{\beta}},h_0)\mathbf{e}\left(w_1^{-\tilde{\alpha}\Delta s_x}w_2^{-\tilde{\beta}\Delta s_x},h_1\right)\mathbf{e}(g_2,h_1)^{\Delta s_y}\mathbf{e}(g_3,h_1)^{\Delta s_z}$.

Let $\tilde{x} = \Delta s_x(\Delta c)^{-1} (\bmod p)$ and $\tilde{y} = \Delta s_y(\Delta c)^{-1} (\bmod p)$. Taking Δcth roots, we have $\mathbf{e}(g_1,h_1)/\mathbf{e}(D_3,h_0) = \mathbf{e}(D_3,h_1)^{\tilde{x}}\mathbf{e}\left(w_1^{-\tilde{\alpha}}w_2^{-\tilde{\beta}},h_0\right)\mathbf{e}\left(w_1^{-\tilde{\alpha}}w_2^{-\tilde{\beta}},h_1^{\tilde{x}}\right)\mathbf{e}(g_2,h_1)^{\tilde{y}}\mathbf{e}(g_3,h_1)^{\tilde{z}}$. This can be equivalently rewritten as $\mathbf{e}(g_1,h_1) = \mathbf{e}\left(\left(D_3w_1^{-\tilde{\alpha}}w_2^{-\tilde{\beta}}\right)^{\tilde{x}+\theta}g_2^{\tilde{y}}g_3^{\tilde{z}},h_1\right)$. Let $\widetilde{A} = D_3w_1^{-\tilde{\alpha}}w_2^{-\tilde{\beta}}$. Then we have $(\widetilde{A},\tilde{x},\tilde{y},\tilde{x})$ satisfying $\mathbf{e}(g_1,h_1) = \mathbf{e}(\widetilde{A}^{\tilde{x}+\theta}g_2^{\tilde{y}}g_3^{\tilde{z}},h_1)$. This implies $g_1g_2^{-\tilde{y}}g_3^{-\tilde{z}} = \widetilde{A}^{\tilde{x}+\theta}$ because $\mathbf{e}$ is non-degenerated and $\mathbf{e}(\cdot,h_1)$ for a fixed $h_1 \in \mathbb{G}_2$ is injective.

Consequently, the extractor obtains a valid SDH$^+$ tuple $(\widetilde{A},\tilde{x},\tilde{y},\tilde{z})$. Also, $\widetilde{A}$ and $g^{\tilde{y}}$ in this SDH$^+$ tuple are the same as those in the ciphertext (D_1,D_2,D_3,D_4). □

Lemma 3. *For an honest verifier, the above protocol can be simulated under the DLC assumption.*

Proof. We build a simulator that gives transcripts of the proposed ZKPK protocol for SDH$^+$. First, the simulator picks $A,\widehat{Z} \xleftarrow{R} \mathbb{G}_1$, and $\alpha,\beta \xleftarrow{R} \mathbb{Z}_p^*$. It sets $D_1 \leftarrow u^{\alpha},\ D_2 \leftarrow v^{\beta},\ D_3 \leftarrow Aw_1^{\alpha}w_2^{\beta}$, and $D_4 \leftarrow \widehat{Z}d_1^{\alpha}d_2^{\beta}$. Under the DLC assumption, the tuples (D_1,D_2,D_3,D_4) are drawn from a distribution that is indistinguishable from the distribution output by any prover.

Since the simulation does not require the knowledge of A, x, y, z, α or β, it can also be used even when D_1, D_2, D_3 and D_4 are pre-specified. When (D_1,D_2,D_3,D_4) is a random ciphertext of the HLCE scheme of some A, the remainder of the transcript is simulated perfectly as follows. The simulator first chooses a challenge $c \xleftarrow{R} \mathbb{Z}_p^*$ and response values $s_{\alpha},\ s_{\beta},\ s_x,\ s_y,\ s_z,\ s_{\gamma},\ s_{\delta} \xleftarrow{R} \mathbb{Z}_p^*$ uniformly at random. It then computes commitment values R_1, R_2, R_3, R_4, R_5, and R_6 as follows:

$R_1 \leftarrow u^{s_\alpha} \cdot D_1^{-c}$, $R_2 \leftarrow v^{s_\beta} \cdot D_2^{-c}$, $R_3 \leftarrow \mathbf{e}(D_3,h_1)^{s_x}\mathbf{e}(w_1,h_\theta)^{-s_\alpha}\mathbf{e}(w_2,h_\theta)^{-s_\beta}\mathbf{e}(d_1,g_2)^{-s_\gamma}\mathbf{e}(d_2,g_2)^{-s_\delta}\mathbf{e}(g_2,h_1)^{s_y}\mathbf{e}(g_3,h_1)^{s_z}(\mathbf{e}(D_3,h_\theta)/\mathbf{e}(g_1,h_1))^c$, $R_4 \leftarrow g^{s_y}d_1^{s_\alpha}d_2^{s_\beta}D_4^{-c}$, $R_5 \leftarrow D_1^{-s_x} \cdot u^{-s_\gamma}$, and $R_6 \leftarrow D_2^{-s_x} \cdot u^{-s_\delta}$. The simulator outputs the transcript $(D_1,\ldots,D_4,R_1,\text{-}D_1,\ldots,D_4,R_1,\ldots,R_6,c,s_\alpha,s_\beta,s_x,s_y,s_z,s_\gamma,s_\delta)$. Note that the resulting values satisfy Eqs. (1)–(6), and $R_1,\ldots,R_6$ are distributed as in a real transcript. As discussed above, this transcript is indistinguishable from transcripts of the ZKPK protocol for SDH$^+$ under the DLC assumption. □

5. Our construction

We construct a CL–GS scheme for dynamic membership. For convenience we call this scheme GS-L.

SetUp. It takes as input a security parameter 1^λ and generates a group public key *gpk* and its corresponding master private keys *mik*, *mok*, and *mlk* as follows. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be a tuple of groups of prime order p that defines a bilinear map $\mathbf{e}:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$. It picks random $h_1 \xleftarrow{R} \mathbb{G}_2$ and $g_1,g_2,g_3,g,u,v \xleftarrow{R} \mathbb{G}_1\setminus\{1_{\mathbb{G}_1}\}$ and $\eta_1,\eta_2,\xi_1,\xi_2,\theta \xleftarrow{R} \mathbb{Z}_p^*$ and computes $w_1 = u^{\eta_1}$, $w_2 = v^{\eta_2}$, $d_1 = u^{\xi_1}$, $d_2 = v^{\xi_2}$, $U = h_1^{\xi_1}$, $V = h_1^{\xi_2}$, and $h_\theta \leftarrow h_1^\theta$. Let $\mathcal{H}:\{0,1\}^* \to \mathbb{Z}_p^*$ be a cryptographic hash function. A group public key is $gpk = (\mathbf{e},\mathbb{G}_1,\mathbb{G}_2,g,h_1,h_\theta,u,v,w_1,w_2,d_1,d_2,\mathcal{H},g_1,g_2,g_3)$, and the master issuing key, $mik = \theta$, the master opening key, $mok = (\eta_1,\eta_2,\xi_1,\xi_2)$, and the master linking key, $mlk = (U,V)$. The group public key, or, more concretely the parameters g_1, g_2, and g_3 will be updated per revocation. For distinction, denote by gpk_0 the initial group key which is generated in this SetUp.

UserJoin/Issue. Two algorithms, UserJoin (run by a joining user) and Issue (run by the **Issuer**) interactively perform a protocol to generate a user signing key $\mathbf{usk}[i] = \left(\widetilde{A}_i = (\tilde{g}_1\tilde{g}_2^{-y_i}\tilde{g}_3^{-z_i})^{\frac{1}{\theta+x_i}},\ x_i,y_i,z_i,A_i = (g_1g_2^{-y_i}g_3^{-z_i})^{\frac{1}{\theta+x_i}}\right)$ where $\widetilde{A}_i = (\tilde{g}_1\tilde{g}_2^{-y_i}\tilde{g}_3^{-z_i})^{\frac{1}{\theta+x_i}}$ and $A_i = (g_1g_2^{-y_i}g_3^{-z_i})^{\frac{1}{\theta+x_i}}$ correspond to the current group public key *gpk* and the initial group public key gpk_0, respectively. Here, $z_i \in \mathbb{Z}_p^*$ is privately selected by UserJoin and $x_i,y_i \in \mathbb{Z}_p^*$ are selected by Issue. A_i will be used to update the current $\widetilde{A}_i$ in $\mathbf{usk}[i]$ whenever a revocation event occurs. Refer to Section 6.

GSig. It takes as input (current) *gpk*, a user signing key $\mathbf{usk}[i] = (\widetilde{A},x,y,z,A)$ corresponding to *gpk*, and a message M, and then proceeds as follows. First, it picks $\alpha,\beta \xleftarrow{R} \mathbb{Z}_p^*$ and computes $D_1 \leftarrow u^\alpha$, $D_2 \leftarrow v^\beta$, $D_3 \leftarrow \widetilde{A}w_1^\alpha w_2^\beta$, $D_4 \leftarrow g^y d_1^\alpha d_2^\beta$, and $\gamma \leftarrow x\alpha \bmod p$, $\delta \leftarrow x\beta \bmod p$. It also picks $r_\alpha,r_\beta,r_\gamma,r_\delta,r_x,r_y,r_z \xleftarrow{R} \mathbb{Z}_p^*$ and computes $R_1 \leftarrow u^{r_\alpha}$, $R_2 \leftarrow v^{r_\beta}$, $R_3 \leftarrow \mathbf{e}(D_3,h_1)^{r_x}\mathbf{e}(w_1,h_\theta)^{-r_\alpha}\mathbf{e}(w_2,h_\theta)^{-r_\beta}\cdot\mathbf{e}(w_1,h_1)^{-r_\gamma}\mathbf{e}(w_2,h_1)^{-r_\delta}\mathbf{e}(g_2,h_1)^{r_y}\mathbf{e}(g_3,h_1)^{r_z}$, $R_4 \leftarrow g^{r_y}d_1^{r_\alpha}d_2^{r_\beta}$, $R_5 \leftarrow D_1^{r_x}u^{-r_\gamma}$, and $R_6 \leftarrow D_2^{r_x}v^{-r_\delta}$. It computes $c \leftarrow \mathcal{H}(M,D_1,\ldots,D_4,R_1,\ldots,R_6) \in \mathbb{Z}_p$, and $s_\alpha \leftarrow r_\alpha + c\alpha$, $s_\beta \leftarrow r_\beta + c\beta$, $s_\gamma \leftarrow r_\gamma + c\gamma$, $s_\delta \leftarrow r_\delta + c\delta$, $s_x \leftarrow r_x + cx$, $s_y \leftarrow r_y + cy$, $s_z \leftarrow r_z + cz \bmod p$. Finally, it outputs a signature $\sigma = (D_1,D_2,D_3,D_4,c,s_\alpha,s_\beta,s_\gamma,s_\delta,s_x,s_y,s_z)$.

GVfy. It takes as input a message M, a signature $\sigma = (D_1,D_2,D_3,D_4,c,s_\alpha,s_\beta,s_\gamma,s_\delta,s_x,s_y,s_z)$, and *gpk* corresponding to σ, and then computes $R_1 \leftarrow u^{s_\alpha} \cdot D_1^{-c}$, $R_2 \leftarrow v^{s_\beta} \cdot D_2^{-c}$, $R_3 \leftarrow \mathbf{e}(D_3,h_1)^{s_x}\mathbf{e}\left(w_1^{-s_\alpha}w_2^{-s_\beta},h_\theta\right)\mathbf{e}\left(w_1^{-s_\gamma}w_2^{-s_\delta},h_1\right)\mathbf{e}(g_2,h_1)^{s_y}\mathbf{e}(g_3,h_1)^{s_z}(\mathbf{e}(D_3,h_\theta)/\mathbf{e}(g_1,h_1))^c$, $R_4 \leftarrow g^{s_y}d_1^{s_\alpha}d_2^{s_\beta}D_4^{-c}$, $R_5 \leftarrow D_1^{s_x} \cdot u^{-s_\gamma}$, and $R_6 \leftarrow D_2^{s_x} \cdot v^{-s_\delta}$. It then checks if $c \stackrel{?}{=} \mathcal{H}(M,D_1,D_2,D_3,D_4,R_1,R_2,R_3,R_4,R_5,R_6)$. If the equality holds then it outputs 1, and otherwise, 0.

Open. It takes as input a signature σ, message M, *gpk* corresponding to σ, and the master opening key $mok = (\eta_1,\eta_2,\xi_1,\xi_2)$. If the signature is not valid then return $\perp$. Otherwise, it proceeds as follows. It recovers $g^y \leftarrow D_4 \cdot \left(D_1^{\xi_1} \cdot D_2^{\xi_2}\right)^{-1}$ (and possibly $\widetilde{A} \leftarrow D_3 \cdot (D_1^{\eta_1} \cdot D_2^{\eta_2})^{-1}$). Using a binary search on the registration list **REG**, it finds i such that $\mathcal{H}(g^y) = \mathcal{H}(g^{y_i})$ in **REG**. If no such i exists then it returns $(i = 0, *)$. Otherwise, it finds its corresponding values $(\mathbf{upk}[i] = Z_i = g_3^{z_i})$, $Y_{1,i} = g_2^{y_i}$, $Y_{2,i} = h_1^{y_i}$, $X_{1,i} = g^{x_i}$, and $X_{2,i} = h_1^{x_i}$. Next it picks $r_1,r_2 \xleftarrow{R} \mathbb{Z}_p^*$ and computes $K_{12} \leftarrow D_1^{\eta_1}D_2^{\eta_2}$, $W_1 = u^{r_1}, W_2 = v^{r_2}, W_{12} = D_1^{r_1}D_2^{r_2}$, $c_{12} = \mathcal{H}(\sigma,u,v,K_{12},W_1,W_2,W_{12})$, and $s_1 = r_1 + c_{12}\eta_1 \bmod p$ and $s_2 = r_2 + c_{12}\eta_2 \bmod p$. Return a proof $(i,\tau = (K_{12},c_{12},s_1,s_2))$ and $(\sigma_{Judge,i},(\mathbf{upk}[i] = Z_i),Y_{1,i},Y_{2,i},X_{1,i},X_{2,i})$.

The $\sigma_{Judge,i}$, which is an ordinary signature generated by a signer i, is a kind of certificate guaranteeing that the signer is aware of x_i, y_i, and z_i. By the construction of the joining procedure in Section 6, we have $\mathbf{e}(X_{1,i},h_1) = \mathbf{e}(g,X_{2,i})$ and $\mathbf{e}(Y_{1,i},h_1) = \mathbf{e}(g_2,Y_{2,i})$.

Judge. It takes as input a signature σ, a message M, *gpk* corresponding to σ, and i, $\tau = (K_{12},c_{12},s_1,s_2)$, $(\sigma_{Judge,i}$, and $(\mathbf{upk}[i] = Z_i)$, $Y_{1,i}$, $Y_{2,i}$, $X_{1,i}$, $X_{2,i})$. If the signature is not valid then return $\perp$. Otherwise, it proceeds as follows. Let $\tilde{h}_1 = h_1^{\log_{g_1}\tilde{g}_1}$ where $\tilde{g}_1 \in gpk$ is an updated value of g_1 which was used to generate σ. If $i = 0$ then return 0. Otherwise, it checks if the following equalities hold: (1) $c_{12} \stackrel{?}{=} \mathcal{H}(\sigma,u,v,K_{12},u^{s_1}w_1^{-c_{12}},v^{s_2}w_2^{-c_{12}},D_1^{s_1}D_2^{s_2}K_{12}^{-c_{12}})$, (2) $\mathbf{e}\left(D_3(K_{12})^{-1},X_{2,i}h_\theta\right) \stackrel{?}{=} \mathbf{e}\left(g_1Y_{1,i}^{-1}Z_i^{-1},\tilde{h}_1\right)$, where g, g_2, h_1, and h_θ are in the initial group public key gpk_0. If all the equalities hold then return 1 (`valid`), and otherwise, 0.

Link. For two given pairs of signatures and messages, (σ,M) and (σ',M'), and the master linking key $mlk = (U,V)$, it first checks if the signatures are valid. If so, it computes $B_1 \leftarrow \mathbf{e}(D_4,h_1)[\mathbf{e}(D_1,U)\mathbf{e}(D_2,V)]^{-1}$ and $B_2 \leftarrow \mathbf{e}(D'_4,h_1)\left[\mathbf{e}(D'_1,U)\mathbf{e}(D'_2,V)\right]^{-1}$. If $B_1 = B_2$ then it outputs 1 (`linked`) and otherwise, 0 (`unlinked`). Alternatively, $\mathbf{e}(D_4/D'_4,h_1) \stackrel{?}{=} \mathbf{e}(D_1/D'_1,U)\mathbf{e}(D_2/D'_2,V)$ can be used for the above equality test.

Note that even after the public key update according to the revocation, Link algorithm is not effected by the change and so, a signature of a user is linked always.

5.1. Security

It is obvious that the proposed scheme GS-L is correct. Next we show that GS-L achieves security properties, i.e., anonymity (under CPA), traceabiltiy, non-frameability, and linkability in the security model in Section 2. In our security proofs, we consider the random oracle model, that is, the hash function $\mathcal{H}$ of GS-L is considered as a random function mapping an input to a random response chosen from its output domain.

Theorem 1 (*Anonymity under CPA*). GS-L *is CPA-anonymous under the DLC assumption in the random oracle model.*

Proof. We will construct an efficient algorithm $\mathcal{S}$ which breaks the LCE scheme using an adversary $\mathcal{F}$ who breaks the anonymity of GS-L under CPA.

Assume that a public key of the LCE scheme, $(\mathbb{G}_1, p, u, v, w_1, w_2)$ is given to $\mathcal{S}$. The group $\mathbb{G}_1$ is associated with a bilinear map $\mathbf{e}: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $\mathcal{S}$ generates the remaining parameters and keys according to SetUp. The group public key $gpk = (\mathbf{e}, \mathbb{G}_1, \mathbb{G}_2, g, h_1, h_\theta, u, v, w_1, w_2, d_1, d_2, \mathcal{H}, g_1, g_2, g_3)$ is given to $\mathcal{F}$. Note that ($\eta_1 = \log_u w_1$, $\eta_2 = \log_v w_2$) of the master opening key is not known to $\mathcal{S}$. By the definition of the experiment of E-Linkability, all keys except the master opening key are given to $\mathcal{F}$.

- **Query I.** $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries as follows:
  - $\mathcal{H}$: On a hash query, return a random value in $\mathbb{Z}_p^*$ if the query has not been issued. Assume that hash queries are never repeated.
  - Link: On query $(m_0, \sigma_0, m_1, \sigma_1)$, $\mathcal{S}$ responds with a correct result using the master linking key *mlk* and the Link algorithm.
  - SndToU, WReg, USK, and CrptU: On queries $\mathcal{S}$ can easily respond because $\mathcal{S}$ is aware of the master issuing key *mik* and can correctly generate a user signing key with Join. We omit the details here.
- **Challenge request.** If the adversary $\mathcal{F}$ submits two indices i_0 and i_1, and a message M as a challenge request for CPA-anonymity then $\mathcal{S}$ submits $(M_0 = A_{i_0}, M_1 = A_{i_1})$ corresponding to (i_0, i_1) as its challenge request for the semantic security of LCE. For a random bit $b \in \{0,1\}$, an LCE-ciphertext $(D_1 = u^\alpha, D_2 = v^\beta, D_3 = w_1^\alpha w_1^\beta M_b)$ for M_b is generated. After receiving (D_1, D_2, D_3) as a challenge, $\mathcal{S}$ picks a random bit $\hat{b} \in \{0,1\}$ and then generates $D_4 = D_1^{\xi_1} D_2^{\xi_2} g^{y_{i_{\hat{b}}}} = d_1^\alpha d_2^\beta g^{y_{i_{\hat{b}}}}$ using (ξ_1, ξ_2). Using the simulation method of Lemma 3, $\mathcal{S}$ creates $\sigma = (D_1, \ldots, D_4, R_1, \ldots, R_6, c, s_\alpha, s_\beta, s_\gamma, s_\delta, s_x, s_y, s_z)$ associated with the $(D_1, \ldots, D_4)$, even without knowledge of α, β, where c is selected uniformly at random in $\mathbb{Z}_p^*$. Then, $\mathcal{S}$ defines $\mathcal{H}(M, D_1, \ldots, D_4, R_1, \ldots, R_6)$ by c. This is valid because we consider the random oracle model. $\mathcal{S}$ returns σ to $\mathcal{F}$ as a challenge.
- **Query II.** Similar to the above query phase, $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries.
  Suppose that $\mathcal{F}$ finally outputs a bit $b_{\mathcal{F}}$. In turn, $\mathcal{S}$ outputs $b_{\mathcal{S}} = b_{\mathcal{F}}$ as the answer to its own challenge.
  In the above simulation, if $b = \hat{b}$ then σ is a valid group signature for $(A_{i_b}, x_{i_b}, y_{i_b}, z_{i_b})$, because $(D_1, \ldots, D_4)$ is a random hybrid LCE-ciphertext of the message tuple $(A_{i_b}, g^{y_{i_b}})$ and the remaining part of the transcript is distributed identically as in the real ZKPK protocol. In this case, $\mathcal{S}$'s guess is correct if and only if $\mathcal{F}$'s guess is correct. The probability of $b = \hat{b}$ is 1/2. Therefore, if $\mathcal{F}$ succeeds in breaking the anonymity of GS-L with a certain advantage ε then $\mathcal{S}$ succeeds in breaking the semantic security of the *LCE* scheme with $\varepsilon/2$. Thus, we have the desired result. □

Theorem 2 (*E-**linkability***). GS-L *is E-linkable under the discrete logarithm assumption on* $\mathbb{G}_1$ *in the random oracle model.*

Proof. We will construct an efficient algorithm $\mathcal{S}$ which solves the discrete logarithm (DL) problem in $\mathbb{G}_1$ using $\mathcal{F}$ attacking the E-linkability of GS-L. Let $(\hat{g}_1, \hat{g}_2) \in \mathbb{G}_1 \times \mathbb{G}_1$ be a random instance of the DL problem. The goal of $\mathcal{S}$ is to compute $\omega = \log_{\hat{g}_1} \hat{g}_2$. Running SetUp, $\mathcal{S}$ generates all parameters and keys and then picks $\tau_1, \tau_2, \zeta \xleftarrow{R} \mathbb{Z}_p^*$ and sets $g_1 = \hat{g}_1^{\tau_1}$, $g_2 = \hat{g}_2^{\tau_2}$, and $g_3 = g_1^\zeta$ for *gpk*. Note that the distribution of these parameters is completely identical to that of the corresponding real parameters generated in SetUp. All master secret keys are known to $\mathcal{S}$. By the definition of the experiment of E-Linkability, assume that *gpk*, *mok*, and *mlk* are given to $\mathcal{F}$.

- **Query.** $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries as follows:
  - $\mathcal{H}$: On a hash query, return a random value in $\mathbb{Z}_p^*$ if the query has not been issued. Assume that hash queries are never repeated.
  - SndToU, AddU, RReg, GSig, USK, and CrptU: Queries can be easily responded because $\mathcal{S}$ is aware of the master issuing key *mik*. We omit the details.

Suppose that an adversary finally outputs (M_0, σ_0), (M_1, σ_1) such that $(i_b, \tau_{i_b}) \leftarrow \mathsf{Open}(M_b, \sigma_b)$, and $1 = \mathsf{Judge}(gpk, \mathbf{upk}[i_b], M_b, \sigma_b, \tau_{i_b})$ for each $b = 0, 1$. Let $A_b = \left(g_1 g_2^{-y_b} g_3^{-z_b}\right)^{\frac{1}{\theta + x_b}}$ be a secret value associated with the proof τ_{i_b} for each $b = 0, 1$. We consider two cases according to the success conditions defined in the experiment of E-Linkability.

- The first case is that $0 = \mathsf{Link}(gpk, mlk, M_0, \sigma_0, M_1, \sigma_1)$ but $i_0 = i_1$. From the first condition, we have $\mathbf{e}(g^{y_0}, h_1) \neq \mathbf{e}(g^{y_1}, h_1)$ and so $y_0 \neq y_1$ because the signatures are valid and "0" output by Link means the values $\mathbf{e}(g^{y_0}, h_1)$, $\mathbf{e}(g^{y_1}, h_1)$ computed from the signatures differ. The condition $i_0 = i_1$ directly means $A_0 = A_1$. $\mathcal{S}$ outputs $\omega' = \frac{(\theta + x_0)(1 - z_1\zeta) - (\theta + x_1)(1 - z_0\zeta)}{\theta(y_1 - y_0) + x_0 y_1 - x_1 y_0} \pmod p$ as a solution to the given DL problem. Note that $g_1^{\omega'} = g_2$ holds, because $A_0 = A_1$, i.e., $\left(g_1 g_2^{-y_0} g_3^{-z_0}\right)^{\frac{1}{\theta + x_0}} = \left(g_1 g_2^{-y_1} g_3^{-z_1}\right)^{\frac{1}{\theta + x_1}}$ implies $(\theta + x_1)(1 - y_0\omega - z_0\zeta) = (\theta + x_0)(1 - y_1\omega - z_1\zeta) \pmod p$ and so $\omega = \frac{(\theta + x_0)(1 - z_1\zeta) - (\theta + x_1)(1 - z_0\zeta)}{\theta(y_1 - y_0) + x_0 y_1 - x_1 y_0} \pmod p$. For random $x_0, x_1, y_0, y_1, \theta \in \mathbb{Z}_p^*$, it is obvious that we have $\theta(y_1 - y_0) + x_0 y_1 - x_1 y_0 \neq 0 \bmod p$ with non-negligible probability.
- The second case is $1 = \mathsf{Link}(gpk, mlk, M_0, \sigma_0, M_1, \sigma_1)$ but $i_0 \neq i_1$. Using an argument similar to the first case, we have $\mathbf{e}(g^{y_0}, h_1) = \mathbf{e}(g^{y_1}, h_1)$ and so $y_0 = y_1$, and $A_0 \neq A_1$. The condition $A_0 \neq A_1$ implies $\left(g_1 g_2^{-y_0} g_3^{-z_0}\right)^{\frac{1}{\theta + x_0}} \neq \left(g_1 g_2^{-y_1} g_3^{-z_1}\right)^{\frac{1}{\theta + x_1}}$ and $(x_0, z_0) \neq (x_1, z_1)$. By construction of the scheme, g^y or $\mathbf{e}(g^y, h_1)$ computed from a valid signature is uniquely defined

for each user (or a signing key). Since y is selected from $\mathbb{Z}_p^*$ uniformly at random for each signing key, the uniqueness can be guaranteed with overwhelming probability. Accordingly, g^y indicates a unique A. By assumption, (M_0,σ_0) and (M_1,σ_1) are valid, i.e., $1 \leftarrow \mathsf{GVfy}(gpk,\sigma_b,M_b)$ and $1 = \mathsf{Judge}(gpk,\mathbf{upk}[i_b],M_b,\sigma_b,\tau_{i_b})$ for each $b = 0, 1$, where $(i_b,\tau_{i_b}) \leftarrow \mathsf{Open}(M_b,\sigma_b)$. Using the Open algorithm, g^{y_0} and g^{y_1} can be uniquely recovered from the signatures σ_0 and σ_1, respectively. Since $g^{y_0} = g^{y_1}$, their associated A_0 and A_1 must be the same, but this contradicts the assumption. □

Theorem 3 (*LO-**linkability***). GS-L *is LO-linkable under the DLC assumption in the random oracle model.*

Proof. In contrast to the anonymity game, we must consider an adversary who can use a linking key to link signatures while he cannot learn the real signer of the signatures, in the LO-linkability game. We will construct an efficient algorithm $\mathcal{S}$ which solves the DLC problem using the adversary $\mathcal{F}$ which breaks the LO-linkability of GS-L. Assume that a DLC problem $(\mathbb{G}_1,p,u,v,w_1,w_2,u^{a^*},v^{b^*},T)$ is given to $\mathcal{S}$. The goal of $\mathcal{S}$ is to guess whether $T = w_1^{a^*}w_2^{b^*}$ or not. Let $\mathbf{e}:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$ be a bilinear map associated with $\mathbb{G}_1$. The algorithm $\mathcal{S}$ generates the remaining parameters and keys except the master opening key *mok* according to SetUp. Note that $mok = (\eta_1 = \log_u w_1, \eta_2 = \log_v w_2, \xi_1, \xi_2)$ is implicitly defined in the above construction. Thus $\mathcal{S}$ cannot learn $\eta_1 = \log_u w_1$ and $\eta_2 = \log_v w_2$. However, $\mathcal{S}$ is aware of $mik = \theta$ and $mlk = (U = h_1^{\xi_1}, V = h_1^{\xi_2})$ because he picked θ, ξ_1, and ξ_2. By the definition of the experiment of LO-Linkability, assume that the group public key $gpk = (\mathbf{e},\mathbb{G}_1,\mathbb{G}_2,g,h_1,h_\theta,u,v,w_1,w_2,d_1,d_2,\mathcal{H},g_1,g_2,g_3)$ and *mlk* are given to $\mathcal{F}$.

- **Query I.** $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries as follows:
  - $\mathcal{H}$: On a hash query, return a random value in $\mathbb{Z}_p^*$ if the query has not been issued. Assume that hash queries are never repeated.
  - SndToU, AddU, GSig, Open, USK, and CrptU: Queries can be easily responded because $\mathcal{S}$ makes use of the master issuing key *mik* and signing keys generated with *mik*. We omit the details.
- **Challenge request.** Assume that $\mathcal{F}$ outputs two indices i_0 and i_1, and a message M as a challenge request for LO-linkability. Let $(A_{i_0},g^{y_{i_0}})$ and $(A_{i_1},g^{y_{i_1}})$ be the secret values corresponding to i_0 and i_1, respectively. The values y_{i_0} and y_{i_1} related to linking information are selected uniformly at random by construction. $\mathcal{S}$ picks a random bit $b\in\{0,1\}$ and generates a signature σ as follows: (1) Set $D_1 \leftarrow u^{a^*}$, $D_2 \leftarrow v^{b^*}$ and compute $D_3 \leftarrow A_{i_b}\cdot T$, $D_4 \leftarrow g^{y_{i_0}}\cdot D_1^{\xi_1}D_2^{\xi_2}$. (2) Pick $c \xleftarrow{R} \mathbb{Z}_p^*$ and $s_\alpha,s_\beta,s_\gamma,s_\delta,s_x,s_y,s_z \xleftarrow{R} \mathbb{Z}_p^*$. (3) Compute $R_1 \leftarrow u^{s_\alpha}\cdot D_1^{-c}$, $R_2 \leftarrow v^{s_\beta}\cdot D_2^{-c}$, $R_3 \leftarrow \mathbf{e}(D_3,h_1)^{s_x}\mathbf{e}(w_1,h_\theta)^{-s_\alpha}\mathbf{e}(w_2,h_\theta)^{-s_\beta}\mathbf{e}(d_1,g_2)^{-s_\gamma}\mathbf{e}(d_2,g_2)^{-s_\delta}\mathbf{e}(g_2,h_1)^{s_y}\mathbf{e}(g_3,h_1)^{s_z}(\mathbf{e}(D_3,h_\theta)/\mathbf{e}(g_1,h_1))^c$, $R_4 \leftarrow g^{s_y}d_1^{s_\alpha}d_2^{s_\beta}D_4^{-c}$, $R_5 \leftarrow D_1^{-s_x}\cdot u^{-s_\gamma}$, and $R_6 \leftarrow D_2^{-s_x}\cdot u^{-s_\delta}$. The signature $\sigma = (D_1,\ldots,D_4,R_1,\ldots,R_6,c,s_\alpha,s_\beta,s_x,s_y,s_z,s_\gamma,s_\delta)$ is given to $\mathcal{F}$ as a challenge.
- **Query II.** Similar to the above query phase, $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries.
  Suppose that $\mathcal{F}$ finally outputs a bit $b_{\mathcal{F}}$. If $b = b_{\mathcal{F}}$ then $\mathcal{S}$ outputs 1, which means $T = w_1^{a^*}w_2^{b^*}$, and else if $b \neq b_{\mathcal{F}}$ then output 0, which means T is random.
  If $T = w_1^{a^*}w_2^{b^*}$ then the presented simulation is perfect. This means that distinguishing the target indices is successful and the equality holds at least with the advantage of $\mathcal{F}$. On the other hand, if T was a random element then $\mathcal{F}$ would get a negligible advantage in distinguishing the target indices.
  Therefore, if $\mathcal{F}$ succeeds in breaking the LO-linkability of GS-L with a certain advantage then $\mathcal{S}$ succeeds in solving the *DLC* problem with at least the advantage. □

Theorem 4 (*JP-**unforgeability***). GS-L *is JP-unforgeable under the discrete logarithm assumption on* $\mathbb{G}_1$ *in the random oracle model.*

Proof. We will construct an efficient algorithm $\mathcal{S}$ which solves the DL problem in $\mathbb{G}_1$ using $\mathcal{F}$ to attack the JP-unforgeability of GS-L. Let $(u,w_1)\in\mathbb{G}_1\times\mathbb{G}_1$ be a random instance of the DL problem. The goal of $\mathcal{S}$ is to compute $\eta^* = \log_u w_1$. Except (u,w_1), the remaining parameters and keys are generated according to the procedure of SetUp. By the definition of the experiment of JP-unforgeability, $gpk = (\mathbf{e},\mathbb{G}_1,\mathbb{G}_2,g,h_1,h_\theta,u,v,w_1,w_2,d_1,d_2,\mathcal{H},g_1,g_2,g_3)$, $mik = \theta$, and $mlk = (U = h_1^{\xi_1}, V = h_1^{\xi_2})$ are given to $\mathcal{F}$. By construction, only $\eta_2 = \log_v w_2$ is known to $\mathcal{S}$.

- **Query I.** $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries as follows:
  - $\mathcal{H}$: On a hash query, return a random value in $\mathbb{Z}_p^*$ if the query has not been issued. Assume that hash queries are never repeated.
  - SndToU, WReg, GSig, Open, USK, and CrptU queries can be easily responded because $\mathcal{S}$ can make use of the master issuing key *mik* and signing keys $\mathbf{usk}[i]$ generated with *mik*. We omit the details.
- **Challenge request.** Assume that $\mathcal{F}$ outputs a message M^* and an index i^* as a challenge. Using $\mathbf{usk}[i^*]$, $\mathcal{S}$ generates a challenge signature $\sigma^* = (D_1^*,D_2^*,D_3^*,D_4^*,\ldots)$ and gives it to $\mathcal{F}$.

– **Query II.** Similar to the first query phase, $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries.
Finally, assume that $\mathcal{F}$ outputs a proof $i^*, \tau^* = (K_{12}^*, c_{12}^*, s_1^*, s_2^*)$ such that $1 \leftarrow \mathsf{Judge}(gpk, \mathbf{upk}[i], M^*, \sigma^*, \tau^*)$. By definition, ($i^*$,-$\tau^*$) satisfies the first verification condition $c_{12} = \mathcal{H}\left(\sigma, u, v, K_{12}^*, u^{s_1^*} w_1^{-c_{12}^*}, v^{s_2} w_2^{-c_{12}^*}, D_1^{*s_1^*} D_2^{*s_2^*} K_{12}^{*\,-c_{12}^*}\right)$.
Note that the given proof τ^* includes a non-interactive ZKPK of $\eta^* = \log_u w_1$ and η_2. Using a similar argument for the forking lemma [36] we can obtain another proof $\tau' = (K_{12}, c'_{12}, s'_1, s'_2)$. Now $\mathcal{S}$ computes $\eta = (s_1^* - s'_1)(c_{12}^* - c'_{12})^{-1}$. and outputs it to the given DL problem. □

Theorem 5 (*Traceability*). GS-L *is traceable under the q-SDH assumption.*

Proof. We will construct an efficient algorithm $\mathcal{S}$ that solves the q-SDH problem using $\mathcal{F}$ to attack the traceability of GS-L. Let $(\hat{g}_1, \hat{g}_1^{\theta}, \hat{g}_1^{\theta^2}, \ldots, \hat{g}_1^{\theta^q}, \hat{h}_1, \hat{h}_1^{\theta})$ be a random instance of the q-SDH problem, where $\theta \xleftarrow{R} \mathbb{Z}_p^*$. The goal of $\mathcal{S}$ is to find a pair $\left(x, g_1^{1/(\theta+x)}\right)$ for some $x \in \mathbb{Z}_p^*$.

To provide $\mathcal{F}$ with a simulation of the experiment for traceability, algorithm $\mathcal{S}$ sets up parameters as follows: $\mathcal{S}$ first picks $\tau, \tau_0, \tau_2, \tau_3, \tau_h \xleftarrow{R} \mathbb{Z}_p^*$ and $x_k, y_k \xleftarrow{R} \mathbb{Z}_p^*$ for $k = 1, \ldots, q$. Define $g_{1,i} = \hat{g}_1^{\theta^i} \in \mathbb{G}_2$, and $F(\theta) = \prod_{k=1}^{q}(\theta + x_k) \bmod p$ and $F_{-i}(\theta) = F(\theta)/(\theta + x_i) = \prod_{k=1, k\neq i}^{q}(\theta + x_k) \bmod p$, and $F_{-i-j}(\theta) = \frac{F(\theta)}{(\theta+x_i)(\theta+x_j)} = \prod_{k=1, k\neq i,j}^{q}(\theta + x_k) \bmod p$. Expanding $F(\theta)$, $F_{-i}(\theta)$, and $F_{-i-j}(\theta)$, we can rewrite $F(\theta) = \sum_{k=0}^{q} c_k \theta^k$, $F_{-i}(\theta) = \sum_{k=0}^{q-1} c_{ik}\theta^k$, and $F_{-i-j}(\theta) = \sum_{k=0}^{q-2} c_{ijk}\theta^k$, where $c_k, c_{ik}, c_{ijk} \in \mathbb{Z}_p$ can be efficiently computed using $x_1, \ldots, x_q$.It computes $g_1 = \left(\prod_{i=0}^{q} g_{1,i}^{c_i}\right)^{\tau} \prod_{i=1}^{q}\left(\prod_{j=0, j\neq i}^{q-1} g_{1,j}^{c_{ij}}\right)^{y_i} = \hat{g}_1^{\tau F(\theta) + \tau_2 \sum_{k=1}^{q} y_k F_{-k}(\theta)}$.It also computes $g = g_1^{\tau_0}$, $g_2 = \hat{g}_1^{\tau_2 \sum_{k=1}^{q} F_{-k}(\theta)}$, $g_3 = \hat{g}_1^{\tau_3 F(\theta)}$, $h_1 = \hat{h}_1^{\tau_h}$ and $h_\theta = \left(\hat{h}_1^{\theta}\right)^{\tau_h}$.$\mathcal{S}$ generates $(u, v, w_1 = u^{\eta_1}, w_2^{\eta_2}, d_1 = u^{\xi_1}, d_2 = v^{\xi_2})$ as the defined procedure.$gpk = (\mathbf{e}, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, g_3, g, h_1, h_\theta, u, v, \quad w_1, w_2, d_1, d_2, \mathcal{H})$ is published.Here, $mok = (\eta_1, \eta_2, \xi_1, \xi_2)$ and $mlk = \left(U = h_1^{\xi_1}, V = h_1^{\xi_2}\right)$ are known to $\mathcal{S}$ while $mik = \theta$ is unknown. By the definition of the experiment of Traceability, *gpk*, *mok*, and *mlk* are given to $\mathcal{F}$.

–**Query I.** $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries as follows:

- $\mathcal{H}$: On a hash query, return a random value in $\mathbb{Z}_p^*$ if the query has not been issued. Assume that hash queries are never repeated.
  Next we show how to respond to AddU oracle queries assuming that the adversary adds all users to a group.
- AddU. On query (i), $\mathcal{S}$ and $\mathcal{A}$ (on behalf of userJoin) proceed as follows:
  (*a*) Upon receipt of a join-request message $(\mathtt{Join\_Req}, ID_i, Z_i = g_3^{z_i}, \sigma_{1,i})$, $\mathcal{S}$ picks $r \xleftarrow{R} \mathbb{Z}_p^*$, computes $R_i = \hat{g}_1^{\tau_3 F_{-i}(\theta) r} = \left(g_3^{\frac{1}{\theta+x_i}}\right)^r$ and sends R_i to $\mathcal{A}$.
  (*b*) $\mathcal{A}$ picks $z_i \leftarrow \mathbb{Z}_p^*$ and computes $\widehat{Z}_i = R_i^{z_i}$, $\widehat{T}_{Join-Eq,i} \leftarrow \mathsf{NIZKPEq2DL}(Z_i, g_3, R_i, \widehat{Z}_i)$, and sends $(\widehat{Z}_i, \widehat{T}_{Join-Eq,i})$ to $\mathcal{S}$.
  (*c*) Upon receipt of valid $(\widehat{Z}_i, \widehat{T}_{Join-Eq,i})$, $\mathcal{S}$ computes $V = \left(R_i^{z_i}\right)^{r^{-1}} = g_3^{\frac{z_i}{\theta+x_i}}$ and $A_i = \hat{g}_1^{\tau F_{-i}(\theta) + \tau_2 \sum_{k=1}^{q}(y_k - y_i) F_{-k-i}(\theta)} V^{-1} = (g_1 g_2^{-y_i} g_3^{-z_i})^{\frac{1}{\theta+x_i}}$, and outputs $(A_i, Y_{1,i} = g_2^{y_i}, X_{2,i} = h_1^{x_i})$.
  (*d*) The remaining steps can be finished in an obvious manner.
- SndToI, RReg, USK, and CrptU: Queries can be easily responded through the construction of the AddU oracle queries. We omit the details.

Assume that $\mathcal{F}$ finally outputs a forgery σ^* on M^*, and $(i^*, \tau^*) \leftarrow \mathsf{Open}(gpk, mok, \mathbf{REG}, M^*, \sigma^*)$. Let g^{y^*} and A^* be values recovered from σ^*. Consider the two cases according to the definition of the traceability game, i.e., (1) $i^* = 0$ and (2) $i^* = i \in \{1, \ldots, q\}$ but $0 \leftarrow \mathsf{Judge}(gpk, \mathtt{upk}[i] = Z_i (= g_3^{z_i}), M^*, \sigma^*, \tau^*)$. Next we show that we can solve the q-SDH problem for each case.

- First, assume that i does not indicate any user in **REG**, i.e., $i^* = 0$.
  (1) Assume that $A^* \neq A_i$ for all $i \in \{1, \ldots, q\}$.
    (a) Assume that $x^* \neq x_i$ for all $i \in \{1, \ldots, q\}$. First, using a similar argument for the forking lemma [6,19], we obtain (x^*, y^*, z^*) corresponding to A^*. Let $Q(\theta) = (\tau - z^*\tau_3)F(\theta) + \tau_2 \sum_{k=1}^{q}(y_k - y^*)F_{-k}(\theta)$. Note that $A^* = \left(g_1 g_2^{-y^*} g_3^{-z^*}\right)^{\frac{1}{\theta+x^*}} = \left(\hat{g}_1^{(\tau - z^*\tau_3)F(\theta) + \tau_2 \sum_{k=1}^{q}(y_k - y^*)F_{-k}(\theta)}\right)^{\frac{1}{\theta+x^*}} = \hat{g}_1^{\frac{Q(\theta)}{\theta+x^*}}$. Here $\frac{Q(\theta)}{\theta+x^*}$ can be expressed as $\frac{Q(\theta)}{\theta+x^*} = Q'(\theta) + \frac{C'}{\theta+x^*}$ where $C' = Q(-x^*) \in \mathbb{Z}_p^*$, and $Q'(\theta)$ is a polynomial of degree $q-1$ and its coefficients can be publicly computed. $\mathcal{S}$ computes $\hat{g}_1^{\frac{1}{\theta+x^*}} = \left(A^* \hat{g}_1^{-Q'(\theta)}\right)^{C'^{-1}}$ and outputs $\left(\hat{g}_1^{\frac{1}{\theta+x^*}}, x^*\right)$ as a solution of the$q$-SDH problem.
    (b) Assume that $x^* = x_i$ for some $i \in \{1, \ldots, q\}$. We have two possible cases for z^* and z_i, as follows:

(1) Assume that $z^* \neq z_i$. First, compute $T = \left(A_i^{z^*} A^{*-z_i}\right)^{(z^*-z_i)^{-1}} = \left((g_1 g_2^{-y_i} g_3^{-z_i})^{\frac{z^*}{\theta+x_i}}\right) \left((g_1 g_2^{-y^*} g_3^{-z^*})^{\frac{-z_i}{\theta+x_i}}\right)^{(z^*-z_i)^{-1}} = \left(g_1 g_2^{\frac{-y_i z^* + y^* z_i}{z^*-z_i}}\right)^{\frac{1}{\theta+x_i}} = \left(\hat{g}_1^{\tau F(\theta) + \tau_3 \sum_{k=1}^{q} y_k F_{-k}(\theta)} \hat{g}_1^{\frac{-y_i z^* + y^* z_i}{z^*-z_i} \tau_2 \sum_{k=1}^{q} F_{-k}(\theta)}\right)^{\frac{1}{\theta+x_i}}$. Let $\rho = \frac{-y_i z^* + y^* z_i}{z^*-z_i} \tau_2$ and $Q(\theta) = \tau F(\theta) + \sum_{j=1}^{q} (\tau_3 y_j + \rho) F_{-j}(\theta)$. We have $T = \hat{g}_1^{\frac{Q(\theta)}{\theta+x_i}}$. Here, $Q(\theta)/(\theta + x_i) = \tau F_{-i}(\theta) + \left(\sum_{j=1, j\neq i}^{q} (\tau_3 Q_j + \rho) F_{-j-i}(\theta)\right) + \frac{F_{-i}(\theta)}{\theta+x_i}$ can be expressed as $Q'(\theta) + \frac{C'}{\theta+x_i}$ where $C' = Q(-x_i)(\neq 0) \in \mathbb{Z}_p^*$, and $Q'(\theta)$ is a polynomial of degree $q-1$ and, its coefficients and C' can be publicly computed. Thus we can compute $\hat{g}_1^{\frac{1}{\theta+x_i}} = (T \hat{g}_1^{-Q'(\theta)})^{C'^{-1}}$ and output $(\hat{g}_1^{\frac{1}{\theta+x_i}}, x_i)$ as a solution of the q-SDH problem.

(2) Assume that $z^* = z_i$. Note that $y^* \neq y_i$ due to $A^* \neq A_i$ and $x^* = x_i$. Compute $T = (A_i A^{*-1})^{(-y_i+y^*)^{-1}} = \left(\left(g_2^{-y_i+y^*}\right)^{\frac{1}{\theta+x_i}}\right)^{(-y_i+y^*)^{-1}} = g_2^{\frac{1}{\theta+x_i}} = \left(\hat{g}_1^{\tau_2 \sum_{k=1}^{q} F_{-k}(\theta)}\right)^{\frac{1}{\theta+x_i}}$. Let $Q(\theta) = \tau_2 \sum_{k=1}^{q} F_{-k}(\theta)$. We have $\hat{g}_1^{\frac{Q(\theta)}{\theta+x_i}}$. Here, $Q(\theta)/(\theta + x_i) = \left(\sum_{k=1, k\neq i}^{q} (\tau_2) F_{-k-i}(\theta)\right) + \frac{F_{-i}(\theta)}{\theta+x_i}$ can be expressed as $Q'(\theta) + \frac{C'}{\theta+x_i}$ where $C' = Q(-x_i)(\neq 0) \in \mathbb{Z}_p^*$, and $Q'(\theta)$ is a polynomial of degree $q-1$, and its coefficients and C' can be publicly computed. Thus, we can compute $\hat{g}_1^{\frac{1}{\theta+x_i}} = \left(T \hat{g}_1^{-Q'(\theta)}\right)^{C'^{-1}}$ and output $\left(\hat{g}_1^{\frac{1}{\theta+x_i}}, x_i\right)$ as a solution of the q-SDH problem.

(3) Assume that $A^* = A_i$ for some i.

(a) If $x^* \neq x_i$, then $\mathcal{S}$ solves the q-SDH problem using a similar method presented in the case (a).

(b) If $x^* = x_i$, $y^* \neq y_i$ and $z^* \neq z_i$ then $\left(g_1 g_2^{-y^*} g_3^{-z^*}\right)^{\frac{1}{\theta+x^*}} = A^* = A_i = \left(g_1 g_2^{-y_i} g_3^{-z_i}\right)^{\frac{1}{\theta+x_i}}$ and so $g_2^{-y^*} g_3^{-z^*} = g_2^{-y_i} g_3^{-z_i}$ holds. Thus, this implies $g_2 = g_3^{(-z_i+z^*)/(-y^*+y_i)}$ and $(-z_i + z^*)/(-y^* + y_i) = \log_{g_3} g_2 = \frac{\tau_2 \sum_{k=1}^{q} F_{-k}(\theta)}{\tau_3 F(\theta)}$ and so $\frac{-z_i+z^*}{-y^*+y_i} \tau_3 F(\theta) - \tau_2 \sum_{k=1}^{q} F_{-k}(\theta) = 0$. For any i, let $Q(\theta) = \frac{-z_i+z^*}{-y^*+y_i} \tau_3 F(\theta) - \tau_2 \sum_{k=1}^{q} F_{-k}(\theta)$ and $\frac{Q(\theta)}{(\theta+x_i)} = \frac{-z_i+z^*}{-y^*+y_i} \tau_3 F_{-i}(\theta) - (\tau_2 \sum_{k=1, k\neq i}^{q} F_{-k-i}(\theta)) + \frac{F_{-k-i}(\theta)}{\theta+x_i}$. This $\frac{Q(\theta)}{(\theta+x_i)}$ can be expressed as $Q'(\theta) + \frac{C'}{\theta+x_i}$ where $C'(\neq 0) \in \mathbb{Z}_p^*$, and $Q'(\theta)$ is a polynomial of degree $q-1$ and its coefficients and C' can be publicly computed. Since $Q(\theta) = 0$, we have $\frac{1}{\theta+x_i} = -C'^{-1} Q'(\theta)$. Thus, we can compute $\hat{g}_1^{\frac{1}{\theta+x_i}} = \left(\hat{g}_1^{Q'(\theta)}\right)^{-C'^{-1}}$ and output $\left(\hat{g}_1^{\frac{1}{\theta+x_i}}, x_i\right)$ as a solution of the q-SDH problem.

- Second, assume that i^* indicates a user in **REG**, i.e., $(A^*, x^*, y^*, z^*) = (A_i, x_i, y_i, z_i)$ for some $i \in \{1, \ldots, q\}$ but $0 \leftarrow \text{Judge}(gpk, \text{upk}[i] = Z_i (= g_3^{z_i}), M^*, \sigma^*, \tau^*)$.

Let $\sigma^* = \left(D_1^*, \ldots, D_4^*, R_1^*, \ldots, R_6^*, c^*, s_\alpha^*, s_\beta^*, s_\gamma^*, s_\delta^*, s_x^*, s_y^*, s_z^*\right)$ and $\tau^* = \left((K_{12}^*, c_{12}^*, s_1^*, s_2^*), (Z_i, Y_{1,i}, Y_{2,i}, X_{1,i}, X_{2,i})_{sign_i}\right)$. It is obvious that the first verification of the Judge algorithm always holds because τ^* was generated according to the opening procedure and so is correct. Suppose that the second condition does not hold; that is, $e\left(D_3 (K_{12})^{-1}, X_{2,i} h_\theta\right) \neq e\left(g_1 Y_{1,i}^{-1} Z_i^{-1}, h_1\right)$, and so $e\left(D_3^* (K_{12})^{-1}, h_1^{\theta+x_i}\right) \neq e\left(g_1 g_2^{-y_i} g_3^{-z_i}, h_1\right)$. This implies that $D_3^* (K_{12})^{-1} = D_3^* (D_1^{*\eta_1} D_2^{*\eta_2})^{-1} \neq \left(g_1 g_2^{-y_i} g_3^{-z_i}\right)^{\frac{1}{\theta+x_i}} = A_i$. Let $D_1 = u^{\alpha^*}$ and $D_2 = v^{\beta^*}$ for some $\alpha^*, \beta^* \in \mathbb{Z}_p^*$. By the proof of Lemma 2, we have $A_i = D_3^* \left(w_1^{\alpha^*} w_2^{\beta^*}\right)^{-1}$. Note that $w_1^{\alpha^*} w_2^{\beta^*} = u^{\eta_1 \alpha^*} w_2^{\eta_2 \beta^*} = D_1^{*\eta_1} D_2^{*\eta_2}$. Thus, these two results contradict the assumption. □

Informally, a user generates its own secret exponent during the joining protocol and only the corresponding commitment is known to the trusted authority Issuer. It is hard to forge a signature for the user without knowledge of the secret exponent because a signature involves a proof of knowledge for the exponent. Next, we formally prove that GS-L is non-frameable under the hardness of the DL problem in $\mathbb{G}_1$.

Theorem 6 (*Non-frameabiltiy*). GS-L *is non-frameable under the discrete logarithm assumption on* $\mathbb{G}_1$ *in the random oracle model.*

Proof. Suppose that there is an adversary $\mathcal{F}$ which breaks the non-frameability of GS-L. We will construct an efficient algorithm $\mathcal{S}$ that solves the DL problem using $\mathcal{F}$ as a sub-algorithm. Let $(\mathbb{G}_1, g_3, Z^* = g_3^z)$ be a random instance of the DL problem in $\mathbb{G}_1$. The goal of $\mathcal{S}$ is to find $z = \log_{g_3} Z^*$.

The simulator $\mathcal{S}$ sets up group public parameters *gpk* according to the procedure of SetUp. Thus, $\mathcal{S}$ is aware of all master secret keys, $mik = \theta$, $mok = (\eta_1, \eta_2, \xi_1, \xi_2)$, and $mlk = (U = h_1^{\xi_1}, V = h_2^{\xi_2})$. By the definition of the experiment of non-frameability, *gpk*, *mik*, *mok*, *mlk* and $\{(A_i, x_i, y_i) | i = 1, \ldots, q + q'\}$ are given to the adversary. Let $q + q'$ be the number of all user keys that are generated during the attack, where q is the number of user keys known to the adversary via corruption or user join queries, and q' is the number of all honest user keys that are not known to the adversary. $\mathcal{S}$ randomly picks $\alpha \in \{1, \ldots, q + q'\}$ such that the adversary will finally output a forged signature of a user with index α. Let **USK** $= \{(A_i, x_i, y_i, z_i) | i = 1, \ldots, q + q'\}$ denote the ordered set of all user keys that are generated and registered during the attack simulation. We note that z_α is unknown to $\mathcal{S}$ during the attack. Without loss of generality, assume that $\{z_{i_1}, \ldots, z_{i_q}\}$ is known to $\mathcal{F}$.

The solver $\mathcal{S}$ will simulate an attack environment for the adversary through oracle queries as defined in Section 2.

– **Query.** $\mathcal{S}$ answers to $\mathcal{F}$'s oracle queries as follows:
- $\mathcal{H}$: On a hash query, return a random value in $\mathbb{Z}_p^*$ if the query has not been issued. Assume that hash queries are never repeated.
- USK: On query (i), If $i \neq \alpha$, return $\mathbf{gsk}[i] = (A_i, x_i, y_i, z_i)$ and $\mathbf{usk}[i] = z_i$ in **REG**. Otherwise, i.e., if $i = \alpha$, abort.
- GSig: On query (i,m), If $i \neq \alpha$, then generate σ using the secret signing key $\mathbf{gsk}[i]$, and return it. Otherwise, i.e., if $i = \alpha$ then generate σ using the simulation technique of Lemma 4, and return it.
- SndToU, WReg, and CrptU queries can be easily responded using the master issuing key *mik*. We omit the details.

Assume that the adversary $\mathcal{F}$ finally outputs (m^*, σ^*, i^*) that satisfies the forgery conditions of non-frameability, (1) $1 \leftarrow \mathsf{GVfy}(gpk, m^*, \sigma^*)$, (2) $i^* \in \mathbf{HU}$, $\mathbf{gsk}[i^*] \neq \varepsilon$, (3) $(\mathbf{upk}[i^*], \tau^*) \leftarrow \mathsf{Open}(gpk, mok, \mathbf{REG}, M, \sigma)$, $1 \leftarrow \mathsf{Judge}(gpk, i^*, \mathbf{upk}[i^*], m^*, \sigma^*, \tau^*)$, and (4) $\mathcal{F}$ has not queried $\mathsf{USK}(i)$ or $\mathsf{GSig}(i^*, m^*)$. Let $\sigma^* = \left(D_1^*, \ldots, D_4^*, R_1^*, \ldots, R_6^*, c^*, s_\alpha^*, s_\beta^*, s_\gamma^*, s_\delta^*, s_x^*, s_y^*, s_z^*\right)$. Using a similar argument for the forking lemma technique of [19,4], we can obtain another signature $\hat{\sigma}^* = (D_1^*, \ldots, D_4^*, R_1^*, \ldots, R_6^*, \hat{c}^*, \hat{s}_\alpha^*, \ldots, \hat{s}_\delta^*, \hat{s}_x^*, \hat{s}_y^*, \hat{s}_z^*)$, where $c^* \neq \hat{c}^* \bmod p$. Hence we can compute $z^* = (c^* - \hat{c}^*)^{-1}(s_z^* - \hat{s}_z^*) \bmod p$. Note that the above simulation is perfect unless an abortion occurs. If α is correctly guessed then there is no abortion. Hence, with probability $1/q$, we succeed in solving the given DL problem. □

6. Join and revocation

We present our join and revocation algorithms. Let the initial group public key be $gpk_0 = (\chi, g_1, g_2, g_3)$ where $\chi = (\mathrm{e}, \mathbb{G}_1, \mathbb{G}_2, g, h_1, h_\theta, u, v, w_1, w_2, d_1, d_2, \mathcal{H})$. χ does not change regardless of revocation.

6.1. Join

Our join algorithm makes use of a standard signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$. We assume that, using KGen, each user ID_i obtains a signing key sk_i and its corresponding public verification key vk_i before joining.

Join. Two algorithms, UserJoin (run by a joining user) and Issue (run by the **Issuer**) interactively perform the following protocol. Let the initial group key be $gpk_0 = (\chi, g_1, g_2, g_3)$.

(1) UserJoin picks $z_i \xleftarrow{R} \mathbb{Z}_p^*$ and computes $\mathtt{upk}[i] = Z_i = g_3^{z_i} \in \mathbb{G}_1$ and $\sigma_{1,i} \leftarrow \mathsf{Sign}_{\mathsf{sk}_i}(\mathtt{Join\_Req}, ID_i, Z_i)$, and sends $(\mathtt{Join\_Req}, ID_i, Z_i, \sigma_{1,i})$.
(2) If $\sigma_{1,i}$ is valid, then Issue sends a random $R_i \xleftarrow{R} \mathbb{G}_1$.
(3) UserJoin computes $\widehat{Z}_i = R_i^{z_i} \xleftarrow{R} \mathbb{Z}_p^*$ and generates $\widehat{T}_{Join-Eq,i} \leftarrow \mathsf{NIZKPEq2DL}(Z_i, g_3, R_i, \widehat{Z}_i)$, which is a non-interactive zero-knowledge proof for $\log_{g_3} Z_i = \log_{R_i} \widehat{Z}_i$. It then sends $(\widehat{Z}_i, \widehat{T}_{Join-Eq,i})$.
(4) Upon receipt of $(\widehat{Z}_i, \widehat{T}_{Join-Eq,i})$, Issue checks the validity of $\widehat{T}_{Join-Eq,i}$ and $\widehat{Z}_i \in \mathbb{G}_1$. If successful, then it checks if there exists $(\mathcal{H}(g^{y_i}), ID_i, y_i, \ldots)$ in the list **REG**. If such i exists, it picks $x_i \xleftarrow{R} \mathbb{Z}_p^*$ and computes $A_i = \left(g_1 g_2^{-y_i} Z_i^{-1}\right)^{\frac{1}{\theta + x_i}} \in \mathbb{G}_1$. Otherwise, it picks $x_i, y_i \xleftarrow{R} \mathbb{Z}_p^*$ and computes $Y_{1,i} = g_2^{y_i}$, $X_{2,i} = h_1^{x_i}$, and $A_i = \left(g_1 g_2^{-y_i} Z_i^{-1}\right)^{\frac{1}{\theta + x_i}} \in \mathbb{G}_1$ and then sends $(A_i, Y_{1,i}, X_{2,i})$.
(5) Upon receipt of $(A_i, Y_{1,i}, X_{2,i})$, UserJoin checks $A_i \in \mathbb{G}_1$ and the equality $\mathrm{e}(A_i, X_{2,i} h_\theta) \stackrel{?}{=} \mathrm{e}\left(g_1 Y_{1,i}^{-1} g_3^{-z_i}, h_1\right)$. If successful, then it generates a signature $\sigma_{2,i} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(A_i, Y_{1,i}, X_{2,i}, Z_i)$ and sends it.
(6) If $\sigma_{2,i}$ is valid then Issue sends (x_i, y_i).
(7) Upon receipt of (x_i, y_i), UserJoin updates A_i to $\widetilde{A}_i$ by calling uskUpdate. It then keeps $\mathtt{usk}[i]_\kappa = (\widetilde{A}_i, x_i, y_i, z_i, A_i)$ secret as a user signing key corresponding to the current group public key gpk_κ. Finally, it generates a signature $\sigma_{Judge,i} \leftarrow \mathsf{Sign}_{\mathsf{sk}_i}(\mathtt{upk}[i], Y_{1,i} = g_2^{y_i}, Y_{2,i} = h_1^{y_i}, X_{1,i} = g^{x_i}, X_{2,i} = h_1^{x_i})$ and sends $\sigma_{Judge,i}$, $(\mathbf{upk}[i], Y_{2,i}, X_{1,i})$.
(8) After receiving the final message from UserJoin, if $\mathbf{e}(X_{1,i}, h_1) = \mathbf{e}(g, X_{2,i})$, $\mathbf{e}(Y_{1,i}, h_1) = \mathbf{e}(g_2, Y_{2,i})$ then Issue appends $(\mathcal{H}(g^{y_i}), ID_i, y_i, A_i, \mathtt{upk}[i], Y_{1,i}, Y_{2,i}, X_{1,i}, X_{2,i}, \sigma_{Judge,i})$ to **REG**.

Using a similar method based on non-interactive ZK proof techniques in [19], we can construct a parallel join.

6.2. Revocation

The revocation algorithm consists of two sub-algorithms, gpkUpdate, which updates a group public key, and uskUpdate, which updates a user's signing key. For clarity, we define session number κ which is incremented by one whenever a revocation session occurs. We assume that a set of keys may be revoked for each session. Let the initial group public key be $gpk_0 = (\chi, g_1, g_2, g_3)$.

Revoke. Let the current group public key $gpk_{\kappa-1} = (\chi, \tilde{g}_1, \tilde{g}_2, \tilde{g}_3)$ be given. Assume that a list of r_κ keys to be revoked, $\mathcal{RI} = \{(T_{1,i} = g_1^{\frac{1}{\theta + x_{\kappa,i}}}, T_{2,i} = g_2^{\frac{1}{\theta + x_{\kappa,i}}}, T_{3,i} = g_3^{\frac{1}{\theta + x_{\kappa,i}}}, x_{\kappa,i}) | i = 1, \ldots, r_\kappa\}$ is given. Then, gpkUpdate and uskUpdate algorithms are executed as follows:

- gpkUpdate. It updates the current $gpk_{\kappa-1}$ to gpk_κ as follows. $\tilde{g}_1' \leftarrow \tilde{g}_1 \prod_{i=1}^{r_\kappa} T_{1,i} = g_1^{1+\zeta}$, $\tilde{g}_2' \leftarrow \tilde{g}_2 \prod_{i=1}^{r_\kappa} T_{2,i} = g_2^{1+\zeta}$, and $\tilde{g}_3' \leftarrow \tilde{g}_3 \prod_{i=1}^{r_\kappa} T_{3,i} = g_3^{1+\zeta}$, where $\zeta = \sum_{j=1}^{\kappa} \sum_{i=1}^{r_j} \frac{1}{\theta + x_{j,i}}$. Return $gpk_\kappa = (\chi, \tilde{g}_1', \tilde{g}_2', \tilde{g}_3')$.
- uskUpdate. It takes as input gpk_κ and $\mathcal{RI}$, and updates a user signing key from $\mathtt{usk}_{\kappa-1} = (\widetilde{A}, x, y, z, A)$ to $\mathtt{usk}_{\kappa} = (\widetilde{A}', x, y, z, A)$. Assume that $x \neq x_i$ for any $i = 1, \ldots, r_\kappa$. It then computes $X_{\kappa,i} \leftarrow \left[\left(T_{1,i} T_{2,i}^{-y} T_{3,i}^{-z}\right) A^{-1}\right]^{\frac{1}{x - x_{\kappa,i}}}$ and $\widetilde{A}' \leftarrow \widetilde{A} \prod_{i=1}^{r_\kappa} X_{\kappa,i} = \left(\tilde{g}_1' (\tilde{g}_2')^{-y} (\tilde{g}_3')^{-z}\right)^{\frac{1}{\theta + x}}$. An updated user signing key is $\mathtt{usk}_{\kappa} = (\widetilde{A}', x, y, z, A)$ and it corresponds to the updated group public key gpk_κ.

It is easy to see that the above revocation method is correct because $X_{\kappa,i} = \left[\left(T_{1,i} T_{2,i}^{-y} T_{3,i}^{-z}\right) A^{-1}\right]^{\frac{1}{x - x_{\kappa,i}}} = \left[\left(g_1^{\frac{1}{\theta + x_{\kappa,i}}} g_2^{\frac{-y}{\theta + x_{\kappa,i}}} g_3^{\frac{-z}{\theta + x_{\kappa,i}}}\right) \left(g_1^{\frac{1}{\theta + x}} g_2^{\frac{-y}{\theta + x}} g_3^{\frac{-z}{\theta + x}}\right)^{-1}\right]^{\frac{1}{x - x_{\kappa,i}}} = \left(g_1^{\frac{1}{\theta + x_{\kappa,i}}} g_2^{\frac{-y}{\theta + x_{\kappa,i}}} g_3^{\frac{-z}{\theta + x_{\kappa,i}}}\right)^{\frac{1}{\theta + x}}$ and $\widetilde{A}' = \widetilde{A} \cdot \prod_{i=1}^{r_\kappa} X_{\kappa,i} = \left[(g_1 g_2^{-y} g_3^{-z})^{\frac{1}{\theta + x}} \cdot \prod_{j=1}^{\kappa-1} \left(\prod_{i=1}^{r_j} \left(g_1^{\frac{1}{\theta + x_{j,i}}} g_2^{\frac{-y}{\theta + x_{j,i}}} g_3^{\frac{-z}{\theta + x_{j,i}}}\right)^{\frac{1}{\theta + x}}\right)\right] \cdot \left(\prod_{i=1}^{r_\kappa} \left(g_1^{\frac{1}{\theta + x_{j,i}}} g_2^{\frac{-y}{\theta + x_{j,i}}} g_3^{\frac{-z}{\theta + x_{j,i}}}\right)^{\frac{1}{\theta + x}}\right) = \left[\left(g_1^{1+\zeta}\right) \left(g_2^{1+\zeta}\right)^{-y} \left(g_3^{1+\zeta}\right)^{-z}\right]^{\frac{1}{\theta + x}} = \left(\tilde{g}_1' (\tilde{g}_2')^{-y} (\tilde{g}_3')^{-z}\right)^{\frac{1}{\theta + x}}$.

7. Performance analysis

We analyze the performance of our scheme in terms of signature size and computation overhead, and provide simulation results. This analysis includes a comparison between the best-known GS scheme [6], referred to as BBS, and our own.

Signature length. Let ℓ_p and $\ell_{\mathbb{G}_1}$ be the bit-length of the order of $\mathbb{G}_1$ and an element of $\mathbb{G}_1$, respectively. Our signature consists of four elements of $\mathbb{G}_1$ and eight elements of $\mathbb{Z}_p$ and so its bit-length is $4\ell_{\mathbb{G}_1} + 8\ell_p$. The BBS signature consists of three elements of $\mathbb{G}_1$ and six elements of $\mathbb{Z}_p$ and so its bit-length is $3\ell_{\mathbb{G}_1} + 6\ell_p$. If $\mathbb{G}_1$ is a bilinear group of prime order p on an MNT curve where ℓ_p and $\ell_{\mathbb{G}_1}$ are almost the same, then our signature length is only 33.3% longer than the BBS signature. If $\ell_p = 170$ and $\ell_{\mathbb{G}_1} = 171$ are assumed, our signature length is about 2044 bits or 256 bytes.

Amount of computation. In our scheme, all the pairings $\mathbf{e}(w_1, h_\theta)$, $\mathbf{e}(w_2, h_\theta)$, $\mathbf{e}(w_1, h_1)$, $\mathbf{e}(w_2, h_1)$, $\mathbf{e}(g_2, h_1)$, and $\mathbf{e}(g_3, h_1)$ can be precomputed, and can be included in the group public key gpk or cached by users.

- Signing and verifying: A signer can cache $\mathbf{e}(A, h_1)$ instead of evaluating a pairing for each generation of a signature. Thus, $\mathbf{e}(D_3, h_1)$ can be computed by $\mathbf{e}(A, h_1)\mathbf{e}(w_1, h_1)^{\alpha}\mathbf{e}(w_2, h_1)^{\beta}$ where $Aw_1^{\alpha}w_2^{\beta}, h_1 = D_3$. Accordingly, signing requires no pairing computation and only two exponentiations in $\mathbb{G}_1$ and ten multi-exponentiations. A verifier can derive R_3 by merging $\mathrm{e}(D_3, h_1)^{s_x}$ and $\mathbf{e}(D_3, h_\theta)^{c}$ into $\mathrm{e}\left(D_3, h_1^{s_x} h_\theta^{c}\right)$ and evaluating one pairing. Thus, verifying requires seven multi-exponentiations and one pairing computation. Since all bases for multi-exponentiation in signing are fixed, further speedup by precomputation is possible.
- Opening: To open a signature, computing a relative value, that is, A in the BBS scheme and g^y in our scheme requires only one multi-exponentiation in $\mathbb{G}_1$ and a binary search.
  If a key-update by revocation is considered then the BBS scheme needs an additional extraction of the original A from an opened value $\widehat{A}$ to trace the corresponding signer (see Chapter 7 in [6]). With the help of the Issuer using the master issuing key that the Opener cannot access, this extraction is possible with one exponentiation in $\mathbb{G}_1$. However, without direct use of the master issuing key, the Opener should update all of the secret keys for each revocation. This will require an amount of computation proportional to the total number of group members. In contrast, in our scheme, since the value g^y does not change regardless of revocation, we can directly decide the corresponding signer and thus minimize the opening computation. Judging a proof output by our opening algorithm requires three multi-exponentiations and two pairing computations.
- Linking: The presented linking algorithm computes 6 pairing maps for two given signatures $\sigma = (D_1, \ldots, D_4, \ldots)$ and $\sigma' = (D_1', \ldots, D_4', \ldots)$. The alternative method, i.e., $\mathrm{e}(D_4/D_4', h_1) = \mathrm{e}(D_1/D_1', U)\,\mathrm{e}(D_2/D_2', V)$, requires three pairing computations. If the linking test needs to be executed for a fixed link index $\mathbf{e}(D_4, h_1)$, then only two pairing computations for $\mathrm{e}(D_1', U)\,\mathrm{e}(D_2', V)$ are required for each new signature.
- Key-updating: In both the BBS and our scheme, all but revoked users must update their secret signing keys. Let r be the number of revoked users. This updating requires $3r$ multi-exponentiations in $\mathbb{G}_1$ in our scheme while r exponentiations in $\mathbb{G}_1$ in [6].

Next, we provide some empirical results regarding computation overhead. The test for these results was performed on an Intel Core2 Quad Q9550 model CPU clocked at 2.83 GHz. This test makes use of the d159 curve from the PBC library [35] running on top of Gnu GMP [27] on Linux 2.6.25-14 (Fedora Core 9). Every test result of Tables 1 and 2 is the average of 1,000 tests. For convenience the scheme of [19] is called DP hereafter.

In Table 1, our CL–GS scheme requires 42% and 28% more time than the BBS scheme for generating and verifying a signature, respectively, but the differences are only about 10 ms. $\mathsf{Open}^{\mathsf{NoRev}}$ and $\mathsf{Open}^{10^4\mathsf{Rev}}$ represent the opening procedure after key-updates with 0 and 10,000 revoked users, respectively. For the BBS and DP schemes, we assume that $\mathsf{Open}^{10^4\mathsf{Rev}}$ performs a single exponentiation with the help of the Issuer, that is, using the master issuing key. In presence of 10,000 revoked keys,

Table 1
Performance comparison between BBS [6], DP [19], and our scheme.

	Ours	BBS04 [6]	DP06 [19]
GSig/GVfy	37.76/55.84	26.40/43.34	21.20/39.25
$\mathsf{Open}^{\mathsf{NoRev}}/\mathsf{Open}^{10^4\mathsf{Rev}}$	1.83/1.93	1.95/6.08	1.92/6.29
Open/Judge	12.13/28.86	N/A	N/A
Link	34.19	N/A	N/A

Time: ms.

Table 2
Revocation performance of our scheme.

# of revoked keys	1	10	100
gpkUpdate in our scheme	39.01	39.70	40.23
uskUpdate in our scheme	13.35	32.52	213.12

Time: ms.

Table 3
Comparison.

	An	CL	NF	Tr	Rev.	Pairing type [23]
Ours	CPA	O	O	O	O	Type 1, 2, 3
[6]	CPA	–	O	O	O	Type 1, 2, 3
[19]	CCA	–	O	O	O	Type 2
[5]	CCA	–	O	O	–	Type 1, 2, 3
[22]	CPA	–	–	O	–	Non-pairing based

our opening algorithm is three times faster than that of the BBS and DP schemes, though that gap is not significant. However, note that the opening time for the BBS and DP schemes was computed assuming that the Opener can access the master issuing key, which sometimes might not be the case in real world applications. In principle, our opening algorithm does not need the master issuing key but has a constant performance time. Without of the use of the master issuing key, our opening algorithm significantly outperforms that of the BBS and DP schemes, as mentioned above. 'Open' in the second row of Table 1 represents our opening procedure, which computes g^y and outputs the corresponding proof τ for judgment. Note that the BBS scheme does not have a judge and a link procedure. Since there is no explicit judge algorithm in [19], we cannot test the Judge algorithm of the DP scheme. Finally, our Judge and Link algorithms are also reasonably fast.

Table 2 shows that the running time of uskUpdate increases in proportion to the number of revoked keys in both our scheme and the BBS scheme. In this table, our uskUpdate uses a multi-exponentiation with three group elements and can be improved with simultaneous exponentiations with more than three group elements.

Finally, we compare our scheme with other recent group signatures [6,19,22,5] in terms of security properties, anonymity (An), controllable linkability (CL), non-frameab ility (NF), traceability (Tr), and revocability (Rev). Table 3 shows that our scheme is comparable to these schemes with respects to these properties.

In the above table, since [22] is based on lattices, the type of pairing maps is not considered for [22].

8. Conclusion

We have introduced a novel and useful notion, controllable linkability for a GS scheme with dynamic membership. Through controllable linkability, a linker can always link signatures from a signer even when the signer is revoked and re-joins. The feature is very versatile and useful in many privacy preserving data mining applications. We concretely constructed a GS scheme with the linkability and proved that the proposed scheme achieves not only the basic security properties, anonymity, traceability, non-frameability but also three security requirements regarding linkability, that is, link-only linkability, judge-proof unforgeability, and enforced linkability. In addition, we presented an efficient revocation method and an opening algorithm that can significantly save or minimize computation overhead in the opening procedure even without the master issuing key.

Acknowledgements

This work was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology in Korea (Grant No. 2012-0006493).

Appendix A. The DLC Assumption in generic bilinear groups

We prove that the DLC assumption introduced in Section 3 holds in generic (bilinear) groups [32]. The underlying idea for the proof is similar to that of [6]. We consider four oracles that represent the basic group operations in each of (multiplicative) groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ (possibly $\mathbb{G}_1 = \mathbb{G}_2$), and a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The opaque encoding of elements in each group is modeled using an injective function which maps each $a \in \mathbb{Z}_p$ to the string representation $\xi(a)$ for the group. An adversary can obtain only ξ-representations of group elements through the oracles and so confirm some relations on exponents induced by queries. For the DLC problem, suppose x, y, x_1, y_1, α, β, $c \xleftarrow{R} \mathbb{Z}_p$, $T_0 \leftarrow g^{x_1\alpha+y_1\beta}$, $T_1 \leftarrow g^c$, and $b \xleftarrow{R} \{0,1\}$. We show that no generic algorithm $\mathcal{A}$ that is given the encodings of $u = g^x$, $v = g^y$, $w_1 = g^{x_1}$, $w_2 = g^{y_1}$, $g^{x\alpha}$, $g^{y\beta}$, T_b, T_{1-b} and makes up to q oracle queries can guess the bit, b with probability greater than $\frac{1}{2} + \varepsilon$, where ε is negligible.

Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$. $\mathcal{B}$ maintains three lists, $\mathcal{L}_k = \{(F_{k,i}, \xi_{k,i}) : i = 0, \cdots, \tau_k - 1\}$ $(k = 1, 2, T)$, under the invariant that, at step τ, $\tau_1 + \tau_2 + \tau_T = \tau + 11$. Here, the $F_{\star,\star} \in \mathbb{Z}_p[X, Y, X_1, Y_1, A, B, T_0,$ and $T_1]$ are polynomials in the indeterminates X, Y, X_1, Y_1, A, B, T_0, T_1 with coefficients in $\mathbb{Z}_p$. The $\xi_{\star,\star} \in \{0,1\}^*$ are arbitrary distinct strings. The lists are initialized at step $\tau = 0$ by initializing $\tau_1 \leftarrow 10$, $\tau_2 \leftarrow 1$, $\tau_T \leftarrow 0$, and setting $F_{1,0} = 1$, $F_{1,1} = X$, $F_{1,2} = Y$, $F_{1,3} = X_1$, $F_{1,4} = Y_1$, $F_{1,5} = XA$, $F_{1,6} = YB$, $F_{1,7} = T_0$, $F_{1,8} = T_1$, and $F_{2,0} = 1$. The corresponding strings are set to arbitrary distinct strings. Note that $\mathcal{B}$ can determine the index i of any given string $\xi_{j,i}$ in L_j $(j = 1, 2, T)$. $\mathcal{B}$ provides $\mathcal{A}$ with the encodings $\xi_{1,0}$, $\xi_{1,1}$, …, $\xi_{1,8}$, $\xi_{2,0}$, and responds to $\mathcal{A}$'s queries as follows:

- **Basic group operations** $(\times, \div)$. Given a bit denoting multiplication or division of group $\mathbb{G}_k$ $(k = 1, 2, T)$ and two operands $\xi_{k,i}$ and $\xi_{k,j}$ with $0 \leqslant i, j < \tau_k$, if $b = 1$ then compute $F_{1,\tau_k} \leftarrow F_{k,i} + F_{k,j}$; otherwise, i.e., $b = 0$, $F_{k,\tau_1} \leftarrow F_{k,i} - F_{k,j}$. If $F_{k,\tau_k} = F_{k,l}$ for some $l < \tau_k$, set $\xi_{k,\tau_k} \leftarrow \xi_{k,l}$; otherwise, set ξ_{k,τ_k} to a string in $\{0,1\}^*$ distinct from $\xi_{k,0}, \cdots, \xi_{k,\tau_k-1}$. Add $(F_{k,\tau_k}, \xi_{k,\tau_k})$ to $\mathcal{L}_k$ and return ξ_{k,τ_k}. Then, increment τ_k by one.
- **Pairing**. Given two operands $\xi_{1,i}$ and $\xi_{2,j}$ with $0 \leqslant i < \tau_1$ and $0 \leqslant j < \tau_2$, compute $F_{T,\tau_T} \leftarrow F_{1,i}F_{2,j}$. If $F_{T,\tau_T} = F_{T,l}$ for some $l < \tau_T$, set $\xi_{T,\tau_T} \leftarrow \xi_{T,l}$; otherwise, set ξ_{T,τ_T} to a string in $\{0,1\}^*$. Return ξ_{T,τ_T} and increment τ_T by one.

The total degree of any polynomial in each of the lists is bounded as follows: $\deg(F_{1,i}) \leqslant 2$, $\deg(F_{2,i}) = 0$ (or $\deg(F_{2,i}) \leqslant 2$ if $\mathbb{G}_1 = \mathbb{G}_2$), and $\deg(F_{T,i}) \leqslant 2$ (or $\deg(F_{T,i}) \leqslant 4$ if $\mathbb{G}_1 = \mathbb{G}_2$). After at most q queries, $\mathcal{A}$ terminates and returns a guess $b \in \{0,1\}$. At this point, $\mathcal{B}$ chooses $x, y, x_1, y_1, \alpha, \beta, c \xleftarrow{R} \mathbb{Z}_p$. Consider $t_b \leftarrow x_1\alpha + y_1\beta$ and $t_{1-b} \leftarrow c$, for both choices of $b \in \{0,1\}$. The simulation provided by $\mathcal{B}$ is perfect and reveals nothing to $\mathcal{A}$ about b unless the chosen random values for the indeterminates give rise to a non-trivial equality relation between the simulated group elements that was not revealed to $\mathcal{A}$, i.e., when we assign $X \leftarrow x$, $Y \leftarrow y$, $X_1 \leftarrow x_1$, $Y_1 \leftarrow y_1$, $A \leftarrow \alpha$, $B \leftarrow \beta$ and either $T_0 \leftarrow x_1\alpha + y_1\beta$, $T_1 \leftarrow c$ or the converse $T_0 \leftarrow c$, $T_1 \leftarrow x_1\alpha + y_1\beta$. The condition of such abortion happens only if for some i, j one of the following four cases holds: (1) $f_{1,i} = f_{1,j}$ but the polynomials $F_{1,i}$ and $F_{1,j}$ are not same, i.e., $F_{1,i} \neq F_{1,j}$, (2) $f_{2,i} = f_{2,j}$ but $F_{2,i} \neq F_{1,j}$, (3) $f_{T,i} = f_{T,j}$ but $F_{T,i} \neq F_{1,j}$, (4) any relation similar to the above in which $x_1\alpha + y_1\beta$ and c_1 have been exchanged, where $f_{k,i} = F_{k,i}(x, y, x_1, y_1, \alpha, \beta, x_1\alpha + y_1\beta, c_1)$ for $k = 1, 2, T$.

Next, we show that it is hard to find any of the above four cases. Observe that $\mathcal{A}$ can only manipulate the polynomials on the three lists through additions and subtractions (shown as multiplications and divisions in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$) as well as multiplications between polynomials which are shown as pairings between elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. Notice that in the initial lists, the variable A only appears within the monomial X_1A, and the variable B within the monomial Y_1B. Given the available operations, it is easy to see the following arguments hold in the three group representations: (1) It is hard to generate a polynomial that contains the pair (mX_1A, mY_1B) for any non-zero integer m, which is a prerequisite to compute a multiple of $X_1A + Y_1B$ in $\mathbb{G}_1$ or $\mathbb{G}_2$. The maximum degree in those groups is 2; (2) It is hard to generate the terms FX_1A and FY_1B for any non-zero monomial F of a maximum degree of 2, which is a prerequisite to compute a multiple of $X_1A + Y_1B$ in $\mathbb{G}_T$. The maximum degree in this group is 4.

Since in the above polynomial differences all arguments to the polynomials are independent except for $x_1\alpha + y_1\beta$, it is obvious that the adversary will not be able to cause any of them to cancel identically and non-trivially without knowledge of a multiple of $X_1A + Y_1B$. The adversary is thus reduced to find a numeric cancelation from random assignments of the variables. We compute the probability of a random occurrence of a non-trivial numeric cancelation. Since $F_{1,i} - F_{1,j}$ for fixed i and j is a polynomial of a maximum degree of 2, it vanishes for random assignment of the indeterminates in $\mathbb{Z}_p$ with a probability of at most $2/p$. Similarly, for fixed i and j, the second case occurs with probability 0 (or $\leqslant 2/p$ when $\mathbb{G}_1 = \mathbb{G}_2$), and the third with probability $\leqslant 2/p$ (or $\leqslant 4/p$ when $\mathbb{G}_1 = \mathbb{G}_2$). The same probabilities are found in the analogous cases where $x_1\alpha + y_1\beta$ and c_1 have been exchanged. Absent any of the above events, the distribution of the bit d in $\mathcal{A}$'s view is independent, and $\mathcal{B}$'s probability of making a correct guess is exactly 1/2. By summing over all valid pairs i, j in each case, we find that $\mathcal{A}$ makes a correct guess with advantage $\epsilon \leqslant 2 \cdot \left(\left(\frac{\tau_1(\tau_1-1)}{2}\right)\frac{2}{p} + \left(\frac{\tau_2(\tau_2-1)}{2}\right)\frac{2}{p} + \left(\frac{\tau_T(\tau_T-1)}{2}\right)\frac{4}{p} \right)$. Since $\tau_1 + \tau_2 + \tau_T \leqslant q + 11$, we have $\epsilon \leqslant (q + 11)^2/p$, as required.

References

[1] G. Ateniese, J. Camenisch, S. Hohenberger, B. de Medeiros, Practical group signatures without random oracles, ePrint Archive, Report 2005/385, 2005.
[2] G. Ateniese, J. Camenisch, M. Joye, G. Tsudik, A practical and provably secure coalition-resistant group signature scheme, Crypto 2000, vol. 1880, Springer-Verlag, 2000, pp. 255–270.
[3] G. Ateniese, D.X. Song, G. Tsudik, Quasi-efficient revocation in group signatures, FC 2002, vol. 2357, Springer-Verlag, 2003, pp. 183–197.

[4] D. Boneh, X. Boyen, Short signatures without random oracles and the SDH assumption in bilinear groups, Journal of Cryptology 21 (2) (2008) 149–177.
[5] P. Bichsel, J. Camenisch, G. Neven, N.P. Smart, B. Warinschi, Get Shorty via Group Signatures without Eencryption, SCN 2010, vol. 6280, Springer-Verlag, 2010, pp. 381–398.
[6] D. Boneh, X. Boyen, H. Shacham, Short group signatures, Crypto 2004, vol. 3152, Springer-Verlag, 2004, pp. 41–55.
[7] X. Boyen, C. Delerablee, Expressive subgroup signatures, SCN 2008, vol. 5229, Springer-Verlag, 2008, pp. 185–200.
[8] M. Bellare, D. Micciancio, B. Warinschi, Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions, Eurocrypt 2003, vol. 2656, Springer-Verlag, 2003, pp. 614–629.
[9] M. Bellare, H. Shi, C. Zang, Foundations of group signatures: the case of dynamic groups, CT-RSA 2005, vol. 3376, Springer-Verlag, 2004, pp. 136–153.
[10] D. Boneh, H. Shacham, Group signatures with verifier-local revocation, in: ACM CCS'04, ACM press, 2004, pp. 168–177.
[11] X. Boyen, B. Waters, Compact group signatures without random oracles, Eurocrypt 2006, vol. 4004, Springer-Verlag, 2006, pp. 427–444.
[12] X. Boyen, B. Waters, Full-domain subgroup hiding and constant-size group signatures, PKC 2007, vol. 4450, Springer, 2007, pp. 1–15.
[13] J. Camenisch, Efficient and generalized group signatures, Eurocrypt 1997, vol. 1233, Springer-Verlag, 1997, pp. 465–479.
[14] J. Camenisch, J. Groth, Group signatures: better efficiency and new theoretical aspects, SCN 2004, vol. 3352, Springer, Verlag, 2005, pp. 120–133.
[15] J. Camenisch, T. Gro, Efficient attributes for anonymous credentials, in: ACM CCS 2004, ACM press, 2008, pp. 345–356.
[16] J. Camenisch, A. Lysyanskaya, Dynamic accumulators and application to efficient revocation of anonymous credentials, Crypto 2002, vol. 2442, Springer-Verlag, 2002, pp. 61–76.
[17] J. Camenisch, A. Lysyanskaya, Signature schemes and anonymous credentials from bilinear maps, Crypto 2004, vol. 3152, Springer-Verlag, 2004, pp. 56–72.
[18] D. Chaum, E. van Heyst, Group signatures, Eurocrypt 1991, vol. 547, Springer-Verlag, 1991, pp. 257–265.
[19] C. Delerablee, D. Pointcheval, Dynamic fully anonymous short group signatures, Vietcrypt 2006, vol. 4341, Springer, 2006, pp. 193–210.
[20] P.-A. Fouque, D. Pointcheval, Threshold cryptosystems secure against chosen ciphertext attacks, Asiacrypt 2001, vol. 2248, Springer-Verlag, 2001, pp. 351–368.
[21] S. Gallbraith, Pairings, Advances in Elliptic Curve Cryptography, vol. 317, Cambridge University Press, 2005, pp. 183–213 (Chapter IX).
[22] S.D. Gordon, J. Katz, V. Vaikuntanathan, A group signature scheme for lattice assumptions, Asiacrypt 2010, vol. 6477, Springer-Verlag, 2010, pp. 395–412.
[23] S. Gallbraith, K.G. Paterson, N.P. Smart, Pairings for cryptographers, Discrete Applied Mathematics, vol. 156, Elsevier, 2008, pp. 3113–3121.
[24] J. Groth, Simulation-sound NIZK proofs for a practical language and constant size group signatures, Asiacrypt 2006, vol. 4284, Springer-Verlag, 2006, pp. 444–459.
[25] J. Groth, Fully anonymous group signatures without random oracles, Asiacrypt 2007, vol. 4833, Springer-Verlag, 2007, pp. 164–180.
[26] J. Guo, J. Baugh, S. Wang, A group signature based secure and privacy-preserving vehicular communication framework, in: Proc. of the Mobile Networking for Vehicular Environments Workshop in Conjunction with IEEE INFOCOM, 2007, pp. 103–108.
[27] The GNU Multiple Precision Arithmetic Library. <http://gmplib.org>.
[28] E. Hufschmitt, J. Traoré, Fair blind signatures revisited, Pairing-based Cryptography 2007, vol. 4575, Springer, 2007, pp. 268–292.
[29] IETF, RFC5636 Traceable Anonymous Certificate 2009.
[30] J. Ki, J.Y. Hwang, D. Nyang, B.-H. Chang, D.H. Lee, J. Lim, Constructing strong identity-based designated verifier signatures with self-unverifiability, ETRI Journal 34 (2) (2012) 235–244.
[31] A. Kiayias, M. Yung, Group signatures with efficient concurrent join, Eurocrypt 2005, vol. 3494, Springer-Verlag, 2005, pp. 198–214.
[32] V. Shoup, Lower bounds for discrete logarithms and related problems, Eurocrypt 1997, vol. 1233, Springer-Verlag, 1997, pp. 256–266.
[33] C.M. Park, H.-S. Lee, Pairing-friendly curves with minimal security loss by Cheon's algorithm, ETRI Journal 33 (4) (2011) 656–659.
[34] K. Paterson, Cryptography From Pairings, Advances in Elliptic Curve Cryptography, vol. 317, Cambridge University Press, 2005 (Chapter X).
[35] PBC (Pairing-based Cryptography) Library. <http://crypto.stanford.edu/pb//c>.
[36] D. Pointcheval, J. Stern, Security arguments for digital signatures and blind signatures, Journal of Cryptology 13 (3) (2000) 361–369.
[37] Y. Tseng, J. Jan, A novel ID-based group signature, Information Sciences 120 (1–4) (1999) 131–141.
[38] Q. Wu, J. Domingo-Ferrer, U. Gonzalez-Nicolas, Balanced trustworthiness, safety, and privacy in vehicle-to-vehicle communications, IEEE Transactions on Vehicular Technology 59 (2) (2010) 559–573.