Short Dynamic Group Signature Scheme Supporting Controllable Linkability

Jung Yeon Hwang, Liqun Chen, Hyun Sook Cho, and DaeHun Nyang

***Abstract*— The controllable linkability of group signatures introduced by Hwang *et al.* enables an entity who has a linking key to find whether or not two group signatures were generated by the same signer, while preserving the anonymity. This functionality is very useful in many applications that require the linkability but still need the anonymity, such as sybil attack detection in a vehicular ad hoc network and privacy-preserving data mining. In this paper, we present a new group signature scheme supporting the controllable linkability. The major advantage of this scheme is that the signature length is very short, even shorter than this in the best-known group signature scheme without supporting the linkability. We have implemented our scheme in both a Linux machine with an Intel Core2 Quad and an iPhone4. We compare the results with a number of existing group signature schemes. We also prove security features of our scheme, such as anonymity, traceability, nonframeability, and linkability, under a random oracle model.**

***Index Terms*— Anonymity, privacy, group signature, linkability.**

I. Introduction

PERSONAL information is more and more publicly accessible due to modern technologies, and accordingly, privacy is increasingly becoming an important security property. Based on [11], privacy is characterized by two fundamental notions, anonymity and unlinkability. Intuitively, anonymity means that a user's identity or identifiable information is concealed in authentication messages. Unlinkability means that given two authentication messages, an unauthorized entity cannot tell whether they are generated by the same user or not. Intrinsically, linkable information may be (strictly) weaker than identifiable information with respect to exposure, because identifiable information can always be used as linkable information, but not the reverse.

Generally speaking, for accessing a service, users prefer to preserve their privacy, but the service provider may want to relax their privacy to gain sufficient user information. For an application environment, privacy needs to be adjusted according to the desired policy or reasonable expectation of profit. If the requirements of privacy for both the users and service providers are properly balanced, privacy will be attractive for both of them. Consider an example of data mining [55]. Various personalized services combined with data mining make crucial use of customer behaviors. That is, linkability is the key feature required in data mining. However, anonymity is necessary for privacy. It is possible to hide an identity or identifiable information from transactions while revealing still linkable information. As another example, consider a recommendation system such as the one at Amazon.com. Customers might be happy to participate in the system only if their anonymity is kept and the linkability is given only to their service provider. Customers will feel assured if their buying pattern is revealed only to the service provider and their identities have not been revealed to anyone.

It is well-known in the literature that two cryptographic solutions have been widely used to preserve privacy, a pseudonym system [37] and group signatures (GS) [21]. The pseudonym system supports anonymity, but a signer cannot avoid being linked by anyone who obtains their signatures. A group signature (GS) scheme is considered as one of the most versatile primitives for anonymity. However, following the concept of a traditional GS [21] (or referred to as a normal GS), the linkability is given only to an opener, who is not usually a service provider but a special group manager. A verifier or a service provider who needs individual statistics on transactions, such as a consumer's buying pattern, based on the linkability of the transactions, cannot obtain these statistics without the help of an opener. Involving an opener for linkability introduces a bottleneck that is not only require an extra on-line process but also require a strong trusted relationship between the user and the opener, since the opener breaks the anonymity.

A privacy-protecting signature scheme was recently introduced to provide elaborate privacy controls [35]. Conceptually, it resides between pseudonym systems and normal GS schemes. Neither information identifying a signer nor information linking signatures is revealed explicitly from signatures. However, the anonymity and unlinkability can be controlled by keys. That is, the corresponding signer identity and linkage information can be revealed by an opening key and a linking key, respectively. Using a trapdoor-based approach on these two privacy notions, one can establish a two-level access hierarchy on signer privacy. As mentioned in [35], this feature can be applied to diverse environments that the pseudonym systems and GS schemes might not be fit for. To be more descriptive, this **P**rivacy-protecting **S**ignature scheme with both **O**pening and **L**inking capabilities in a controllable

Manuscript received August 19, 2014; revised November 20, 2014; accepted January 5, 2015. Date of publication January 12, 2015; date of current version April 13, 2015. This work was supported by Inha University. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Willy Susilo. *(Corresponding author: DaeHun Nyang.)*

J. Y. Hwang and H. S. Cho are with the Electronics and Telecommunication Research Institute, Daejeon 305-700, Korea (e-mail: videmot@etri.re.kr; hscho@etri.re.kr).

L. Chen is with Hewlett-Packard Labs, Bristol BS12 6QZ, U.K. (e-mail: liqun.chen@hp.com).

D. Nyang is with the School of Computer and Information Engineering, Inha University, Incheon 402-752, Korea (e-mail: nyang@inha.ac.kr).

Digital Object Identifier 10.1109/TIFS.2015.2390497

1556-6013 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

manner is referred to as a PS-OL scheme for short. A PS-OL scheme supports two seemingly-incompatible properties, that is, privacy and data mining versatility by selectively providing linkability and anonymity. A PS-OL scheme has benefits in flexibly organizing participants over a normal GS, considering that a linker can be built up separately from an opener. This separation enables a bottleneck (strong trusted relationship and on-line processing) to be removed in an anonymous system. In [35], a PS-OL scheme is constructed from a linear combination encryption with many parameters. Since the underlying structure is quite complex, the system requires heavy operations, and its signature length is also relatively long.

A. Our Contributions

In this paper, we construct a PS-OL scheme for a dynamic membership, where group signatures can be anonymously linked, but the corresponding linkage information can only be revealed with a linking key. The linking key is secretly managed by a privileged party called a linker who is delegated the link capability by the opener. Note that the capability of linking signatures is placed below the capability of opening the signer identity of the signatures. We can achieve a stepwise access control on anonymity by adding this Controllable Linkability (CL) to the controllable anonymity that can identify a signer from signatures using an opening key.

Our linking capability differs from the tracing capability of a traceable signature scheme [39]. The traceable signature scheme enables a tracer to trace only a specific user's signatures, not other users. In contrast, a linker of our scheme can deal with every user's linkage information with a key. Though a traceable signature scheme can be used for our linkability, it involves complex computation. For example, for n signatures and m tracing keys, $n \times m$ computation is required for a traceable signature scheme while n computation is required for our scheme.

The proposed PS-OL scheme supports a dynamic group membership where a user can join or leave a group. Leaving a group is also referred as to be revoked. However, the linking capability can be consistently preserved regardless of changes to the membership status of the signer. In addition, the CL property does not expose the history of the joining and revocation.

Despite the additional functionality of CL, our scheme has a compact structure to yield a very short signature that is one group element shorter than the best-known GS [25] 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 only 1193 bits or about 150 bytes. As shown in our comparison results, our scheme is comparable to that in [4] and [25] in terms of the amount of computation for basic operations such as signing, verification, and key updates caused by revocation. We note that, since our linkability for dynamic membership enables an opener to obtain a static linking tag uniquely associated with a signer, computation overhead for opening can be significantly reduced.

B. Related Works

In the literature, signature schemes with various privacy concerns have been extensively studied.

GS schemes provide controllable anonymity such that a signer can be identified from a signature by a trusted group manager. It provides unlinkability on signatures against all users except the group manager. Since the introduction of GS in [21], a number of GS schemes have been presented to address various features; for example, considering performance of computation or communications [1], [3], formal definitions of security properties [10], [12], concurrent joins [25], [38], CCA-anonymity [25], a general framework [16], and revocation for dynamic membership [3], [25]. The applications of a GS scheme for wireless communication environments such as VANET were suggested [31], [49]. Some GS schemes have been suggested without random oracles [14], [15], [30]. Recently, GS schemes have been suggested using lattice [40].

Direct Anonymous Attestation (DAA) has been proposed for the remote anonymous authentication of a trusted platform module [4]. While DAA guarantees complete anonymity, i.e., no party can reveal a signer's identity from a signature, it provides signer-controllable linkability, i.e., a signer can generate an anonymous signature with a tag, which is linkable to another signature from the same signer. Compared with the linkability solution in this paper and [35], DAA's linkability does not include a linking key, so anybody can see if two signatures were created by the same signer or not. Recent improvements on performance and security for a DAA have been widely studied from various mathematical assumptions, see [5], [6], [9], [22].

There are variants of a GS scheme to alleviate the centralized group manager's rights that can reveal a signer's identity. In the Democratic GS scheme [43], the group membership is controlled jointly and equally by all group members. A signer of a signature can be identified by a member, in other words, the signer's anonymity can be provided only against non-members. In the tracing-by-linking GS scheme [56], no signer can be identified by any authority if he or she signs only once per event.

Some schemes with controllable or revocable anonymity provide linkability by adding a tag to a signature [20], [41], [51]. Using the tag associated with a signature, one can check the linkability on signatures easily and explicitly. For example, a linkable democratic GS scheme [44] is a variant of a democratic GS scheme to support the tag-based linkability. A message-linkable GS scheme was suggested to resist Sybil attacks in VANET [57]. The message-linkable property means that given two anonymous signatures on the same message, one can easily decide whether they are generated by the same signer or not. To provide this property, the scheme uses a static tag generated with a message and a secret key.

There are other related works that measures privacy [11], [26], [33]. However, metrics defined in these works

are probabilistic and are not proper to treat privacy definitions in a cryptographic sense. For a brief survey, refer to [11].

C. Organization

The remainder of this paper is organized as follows. In Section II, we present a security model for a PS-OL scheme that captures a revocation method using a key-update. In Section III, we describe computational assumptions and an honest-verifier ZKPK protocol. Using this protocol, in Section IV, we propose a dynamic PS-OL scheme to yield short signatures and formally prove its security in Section V. We analyze the performance of the proposed PS-OL scheme in Section VI. We present some variations of our scheme in Section VII and finally conclude the paper in Section VIII.

II. Model

We present a security model for a PS-OL scheme by modifying the model of [35]. This model assumes three authorities, Issuer, Opener, and Linker, who have their independent privileges and a certain level of trust as in [35]. A linker is assumed to behave honestly but curiously, and so it can try to find passively user's identity only with signatures collected.

In contrast to the original model, our model explicitly considers a revocation algorithm that performs the update of keys. For the revocation, it makes use of a revocation list, denoted by **RL**. An entry of **RL** consists of an index and private information for a user who has been revoked. It is managed by the Issuer and initially set to be empty. The list is used to update a user signature key and a group public key. In this model, whenever the information of keys to be revoked are given according to a pre-defined policy, **RL** is immediately updated to include them and entries are arranged to the latest revocation index. One can publicly access the list.

A user signature key includes a non-negative integer λ, called a revocation index, to indicate that the key has been updated up to the λ^{th} entry of **RL**. Let $\hat{\lambda} > \lambda$ be the most up-to-date number of revoked keys in **RL**. For generation of a signature, the user signature key is updated up to the $\hat{\lambda}^{\text{th}}$ entry of **RL**. The generated signature includes $\hat{\lambda}$ to indicate that the signature was generated by the key which has been updated up to the $\hat{\lambda}^{\text{th}}$ entry of **RL**. It can be verified with the group public key that has been updated up to the $\hat{\lambda}^{\text{th}}$ entry of **RL**.

The PS-OL scheme uses a registration list $\mathbf{REG} = (\mathsf{REG}[1], \ldots, \mathsf{REG}[n])$. $\mathsf{REG}[i]$ contains private information for the i-th registered user. We assume that registered users are all different. **REG** is managed by Issuer and can be accessed by Opener to identify a signer.

A PS-OL scheme consists of the following algorithms.

- SetUp: On input a security parameter, it outputs an initial group public key, gpk_0 and private master keys, mik, mok, and mlk.
- (UserJoin, Issue): UserJoin with index i and a secret key corresponding to a public key of a user, **upk**$[i]$, and Issue with mik interactively run a protocol. After completion of this protocol, UserJoin generates a (full) user signature key **usk**$[i]$ corresponding to **upk**$[i]$. Issue updates or adds the corresponding record of the user in the registration list **REG**.
- GSig: On input **usk**$[i]$, a message $M \in \{0,1\}^*$, it calls Update-usk to obtain gpk' and **usk**$[i]'$ that are updated up to the most-up-to-date revocation index in **RL**, and then outputs a signature σ.
- GVfy: On input (M, σ), it calls Update-gpk to obtain a group public key gpk' that is updated up to the index of the signature σ and then outputs 1 (valid) or 0 (invalid).
- Open: On input (mok, **REG**, M, σ), it outputs i (or a signer's public key **upk**$[i]$) and a judge-proof τ.
- Judge: On input (M, σ, **upk**$[i]$, τ), it outputs 1 or 0. Here 1 means that σ was generated with the secret key corresponding to the public key **upk**$[i]$ for user i.
- Link: On input (mlk, (M_0, σ_0), (M_1, σ_1), it outputs 1 or 0. Here, 1 means the signatures are generated by a user.
- Revocation: This algorithm consists of two sub-algorithms, Update-usk and Update-gpk.
  - Update-usk: On input (**usk**$[i]$, **RL**), it outputs a group public key gpk' and **usk**$[i]'$ that are updated up to the most-up-to-date revocation index of **RL**.
  - Update-gpk: On input (gpk_0, ρ, **RL**), it outputs a group public key gpk_ρ, which is updated up to the ρ^{th} revocation index of **RL**.

Fields of **REG** corresponding to revoked users are immediately set to be empty whenever **RL** is updated.

Denote by **HU** and **CU** the lists of honest and corrupted users, respectively. **GSet** is the set of message-signature pairs. $\mathbf{REG}_i$ is the user i's field on **REG**. The following oracle queries are used by an adversary in experiments (See Figs. 2 and 3). On an $\texttt{AddU}(i)$ query, user i is added to the group as a member of the **HU** list, given **usk**$[i]$ on this oracle. On a $\texttt{CrptU}(i, \mathbf{upk}[i])$ query, a new user i with his/her identifiable information, **upk**$[i]$ is added to **CU**. On a $\texttt{SndToI}(i, M)$ query, a corrupted user i sends M to the honest Issuer. On a $\texttt{SndToU}(i, M)$ query, the (corrupted) Issuer sends a message M to an honest user. On a $\texttt{USK}(i)$ query, **usk**$[i]$ is revealed and an honest user i is turned into a corrupted user. On a $\texttt{RReg}(i)$ query, a read access to $\mathbf{REG}_i$ is given. On a $\texttt{WReg}(i, \rho)$ query, ρ is written or overwritten to $\mathbf{REG}_i$. On a $\texttt{GSig}(i, m)$ query, a group signature of an honest user i is given for a message m. On a $\texttt{GSig}^*(m)$ query, a group signature of a random user is given for a message m. On a $\texttt{Ch}_b(i_0, i_1, m)$ query, the oracle outputs i_b's signature σ_{i_b} for a random bit b and records (σ_{i_b}, m) in the **GSet**. On a $\texttt{Open}(m, \sigma)$ query, the Open algorithm returns the corresponding signer's identity. On a $\texttt{Link}(M_0, \sigma_0, M_1, \sigma_1)$ query, the Link algorithm checks and returns whether the two signatures are generated by a user as shown in Figure 1.

$\texttt{Link}(M_0, \sigma_0, M_1, \sigma_1)$
If $(M_0, \sigma_0) \in$ **GSet** or $(M_1, \sigma_1) \in$ **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.

Security Notions: For a PS-OL scheme $\mathcal{GS}$, correctness and anonymity, traceability, non-frameability, and linkability are defined by the following games.

Correctness: 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 $\mathsf{Exp}^{corr}_{\mathcal{GS},\mathcal{A}}(k)$ as in Figure 2 and say that $\mathcal{GS}$ is *correct* if $\Pr[\mathsf{Exp}^{corr}_{\mathcal{GS},\mathcal{A}}(k) = 1] = 0$.

Anonymity: 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 $\mathsf{Exp}^{anon-b}_{\mathcal{GS},\mathcal{A}}(k)$ in Figure 2 and $\mathsf{Adv}^{anon}_{\mathcal{GS},\mathcal{A}}(k) = |\Pr[\mathsf{Exp}^{anon-1}_{\mathcal{GS},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{anon-0}_{\mathcal{GS},\mathcal{A}}(k) = 1]|$, and say that $\mathcal{GS}$ is *CPA-anonymous* if $\mathsf{Adv}^{anon}_{\mathcal{GS},\mathcal{A}}(\cdot) = 0$ is negligible.

Traceability: Given a valid signature σ, the origin of the signature should not be identified, i.e., a (public) proof to check the origination should not be forged. To any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we define $\mathsf{Exp}^{trace}_{\mathcal{GS},\mathcal{A}}(k)$ in Figure 2 and say that $\mathcal{GS}$ is *traceable* if $\Pr[\mathsf{Exp}^{trace}_{\mathcal{GS},\mathcal{A}}(k) = 1]$ is negligible.

Non-Frameability: Colluding users with even trusted authorities should not forge a signature of an honest user. To any adversary $\mathcal{A}$ and $k \in \mathbb{N}$, we define $\mathsf{Exp}^{nf}_{\mathcal{GS},\mathcal{A}}(k)$ in Figure 2, and say that $\mathcal{GS}$ is *non-frameable* if $\Pr[\mathsf{Exp}^{nf}_{\mathcal{GS},\mathcal{A}}(k) = 1]$ is negligible.

Linkability: We consider three notions for linkability, called *E-linkability* (Enforced linkability), *JP-Unforgeability* (Judge-Proof Unforgeability), and *LO-linkability* (Link-Only linkability), respectively.

- 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 Figure 3, and say that $\mathcal{GS}$ is *E-linkable* if $\Pr[\mathsf{Exp}^{E-link}_{\mathcal{GS},\mathcal{A}}(k) = 1]$ is negligible.
- 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 $\mathsf{Exp}^{JPUF}_{\mathcal{GS},\mathcal{A}}(k)$ in Figure 3 and say that $\mathcal{GS}$ is *JP-unforgeable* if $\Pr[\mathsf{Exp}^{JPUF}_{\mathcal{GS},\mathcal{A}}(k) = 1]$ is negligible.
- 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 $\mathsf{Exp}^{LO-link}_{\mathcal{GS},\mathcal{A}}(k)$ as described in Figure 3 and say that $\mathcal{GS}$ is *LO-linkable* if $\Pr[\mathsf{Exp}^{LO-link}_{\mathcal{GS},\mathcal{A}}(k) = 1]$ is negligible. Here, the adversary's capabiliy 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 $\mathsf{Exp}^{corr}_{\mathcal{GS},\mathcal{A}}(k)$ 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 $\mathsf{GVf}(gpk, M, \sigma) = 0$ then 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

$(gpk, mik, mok, mlk) \leftarrow \mathsf{SetUp}(1^k)$; $\mathbf{CU} \leftarrow \emptyset$; $\mathbf{HU} \leftarrow \emptyset$; $\mathbf{GSet} \leftarrow \emptyset$;
$(i_0, i_1, m) \leftarrow \mathcal{A}(gpk, mik : \mathtt{SndToU}, \mathtt{Link}, \mathtt{WReg}, \mathtt{USK}, \mathtt{CrptU})$;
$d \leftarrow \mathcal{A}(gpk, mik, \sigma_{i_b} : \sigma_{i_b} \leftarrow \mathtt{Ch}_b(i_0, i_1, M)$;
$\mathtt{SndToU}, \mathtt{Link}, \mathtt{WReg}, \mathtt{USK}, \mathtt{CrptU})$,
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 are all true then return 1, else return 0:
- $i \in \mathbf{HU}$ and $\mathbf{usk}[i] \neq \varepsilon$; $(i, \tau) \leftarrow \mathsf{Open}(gpk, mok, \mathbf{REG}, M, \sigma)$;
- $\mathsf{Judge}(gpk, i, \mathbf{upk}[i], M, \sigma, \tau) = 1$; $\mathcal{A}$ did not query $\mathtt{USK}(i)$ or $\mathtt{GSig}(i, M)$

Fig. 2. Experiments for Correctness, Anonymity, Traceability, and Non-Frameability.

III. Preliminaries

We now review bilinear maps and computational assumptions for our scheme. Denote by $s \xleftarrow{R} S$ the operation that picks an element s of a set S uniformly at random.

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

$(gpk, mik, mok, mlk) \leftarrow \mathsf{SetUp}(1^k)$; **CU** $\leftarrow \emptyset$; **HU** $\leftarrow \emptyset$; **GSet** $\leftarrow \emptyset$;
$(M_0, \sigma_0, M_1, \sigma_1) \leftarrow \mathcal{A}(gpk, mok, mlk : \mathtt{SndToU}, \mathtt{RReg}, \mathtt{GSig}, \mathtt{USK}, \mathtt{CrptU})$;
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 $\mathsf{Judge}(gpk, \mathbf{upk}[i_0], M_0, \sigma_0, \tau_{i_0}) = 0$ or
$\mathsf{Judge}(gpk, \mathbf{upk}[i_1], M_1, \sigma_1, \tau_{i_1}) = 0$ 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;
elseif $i_0 = i_1$ and $0 = \mathsf{Link}(gpk, mlk, M_0, \sigma_0, M_1, \sigma_1)$
then return 1;else return 0

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

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

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

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

Fig. 3. Security Experiments for Linkability.

A. Bilinear Maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be multiplicative groups of prime order p and $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ 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 [28], [45], 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.

B. 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 EXternal Diffie-Hellman (XDH) Assumption: Assume that three groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are associated with a bilinear pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Briefly, the XDH assumption is that the DDH problem is hard in $\mathbb{G}_1$ but easy in $\mathbb{G}_2$. The XDH assumption was introduced in [3] and [17], and used to construct a fully-anonymous GS scheme with short signatures in [25]. For more details refer to [3], [17], and [25].

The q-Strong Diffie-Hellman (SDH) Problem [2], [3]: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ 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 $(g_1^{1/(\gamma+x)}, x)$ for a given $(q+3)$-tuple $(g_1, g_1^{\gamma}, \ldots, g_1^{\gamma^q}, h_1, h_1^{\gamma})$ 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 w 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 $((g_1 g_2^y w^z)^{1/(\gamma+x)}, x, y, z)$ for a given $(q+5)$-tuple $(g_1, g_2, w, g_1^{\gamma}, \ldots, g_1^{\gamma^q}, h_1, h_1^{\gamma})$ where $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and $y, z \in \mathbb{Z}_p^*$. 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 $w = g_1^{\beta}$ 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 $T = A^{(1+\alpha y+\beta z)^{-1}} = ((g_1 g_2^y w^z)^{\frac{1}{\gamma+x}})^{(1+\alpha y+\beta z)^{-1}} = g_1^{\frac{1}{\gamma+x}}$.

C. ZKPK Protocol for SDH$^+$

We present an honest-verifier zero-knowledge proof of knowledge (ZKPK) protocol called ZKPK$^{\mathsf{SDH}^+}$ which our scheme relies on. This protocol proves possession of an SDH$^+$ tuple and the equality of two discrete logarithms related to this tuple.[1] Next the concrete protocol is given.

Let $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing, and $g, u, w, d, g_1, g_2 \stackrel{R}{\leftarrow} \mathbb{G}_1$, $h_1 \stackrel{R}{\leftarrow} \mathbb{G}_2$, and $h_\theta = h_1^{\theta}$ for random $\theta \in \mathbb{Z}_p^*$. The public parameters of ZKPK$^{\mathsf{SDH}^+}$ are $(\mathbf{e}, g, h_1, h_\theta, g_1, g_2, u, w, d)$.

Assume that the prover has an SDH$^+$-tuple, (A, x, y, z) where $A^{x+\theta} = g_1 Y^{-1} w^{-z} \in \mathbb{G}_1$ for $x, y, z \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. Let $Y = g_2^y$ and $\hat{Y} = g^y$. The goal of prover $\mathcal{P}$ is to show that $A^{x+\theta} = g_1 Y^{-1} w^{-z}$ and $\log_{g_2} Y = \log_g \hat{Y}$ hold, i.e., generate a proof, $PK[(A, x, y, z) : A^{x+\theta} = g_1 g_2^{-y} w^{-z} \wedge y = \log_g \hat{Y}]$.

Prover $\mathcal{P}$ performs the following protocol with verifier $\mathcal{V}$.

- (Prover) $\mathcal{P}$ picks $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computes $D_1 = u^{\alpha}$, $D_2 = A w^{\alpha}$, $D_3 = g^y d^{\alpha} \in \mathbb{G}_1$, and $\gamma = x\alpha - z \mod p$. It also picks $r_\alpha, r_x, r_\gamma, r_y \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computes $R_1 = u^{r_\alpha}$, $R_2 = \mathbf{e}(D_2, h_1)^{r_x} \mathbf{e}(w, h_\theta)^{-r_\alpha} \mathbf{e}(w, h_1)^{-r_\gamma} \mathbf{e}(g_2, h_1)^{r_y}$, $R_3 = g^{r_y} d^{r_\alpha}$. $(D_1, D_2, D_3, R_1, R_2, R_3)$ is sent to $\mathcal{V}$.

[1] This protocol is different from the ZKPK protocol in [35] which is based on the so-called linear combination encryption.

- (Verifier) $\mathcal{V}$ picks $c \xleftarrow{R} \mathbb{Z}_p^*$ and sends it as a random challenge to $\mathcal{P}$.
- (Prover) $\mathcal{P}$ computes $s_\alpha = r_\alpha + c\alpha, s_x = r_x + cx$, $s_\gamma = r_\gamma + c\gamma$, and $s_y = r_y + cy$ under modulo p, and then sends $(s_\alpha, s_x, s_\gamma, s_y)$ to $\mathcal{V}$.
- (Verifier) $\mathcal{V}$ checks if the followings hold: $R_1 \stackrel{?}{=} u^{s_\alpha} D_1^{-c}$, $R_2 \stackrel{?}{=} \mathbf{e}(D_2, h_1)^{s_x} \mathbf{e}(w, h_\theta)^{-s_\alpha} \mathbf{e}(w, h_1)^{-s_\gamma}$ $\mathbf{e}(g_2, h_1)^{s_y} (\mathbf{e}(D_2, h_\theta)/\mathbf{e}(g_1, h_1))^c$, and $R_3 \stackrel{?}{=} g^{s_y} d^{s_\alpha} D_3^{-c}$. If all equality tests hold then output 1 (valid), and otherwise, 0 (invalid).

In the above protocol, the prover and the verifier execute a proof of knowledge of $(\alpha, \gamma, x, y, z)$ which satisfy the following equations:

(1) $u^\alpha = D_1$,
(2) $\mathbf{e}(D_2, h_1)^x \mathbf{e}(w, h_\theta)^{-\alpha} \mathbf{e}(w, h_1)^{-\gamma} \mathbf{e}(g_2, h_1)^y = \mathbf{e}(g_1, h_1)$ $\mathbf{e}(D_2, h_\theta)^{-1}$,
(3) $g^y d^\alpha = D_3$.

We show that the above protocol ZKPK$^{\mathsf{SDH}^+}$ satisfies the completeness, the soundness, and the zero-knowledge property.

Lemma 1: ZKPK$^{SDH^+}$ is complete.

Proof: Suppose that the protocol with a valid SDH$^+$ tuple (A, x, y, z) is honestly executed. We have the following relations: $\gamma = x\alpha - z \bmod p$ and

$$\begin{aligned}
R_1 &= u^{r_\alpha} = u^{r_\alpha} u^{c\alpha} (u^\alpha)^{-c} = u^{s_\alpha} D_1^{-c}, \\
R_2 &= \mathbf{e}(D_2, h_1)^{r_x} \mathbf{e}(w, h_\theta)^{-r_\alpha} \mathbf{e}(w, h_1)^{-r_\gamma} \mathbf{e}(g_2, h_1)^{r_y} \\
&= \mathbf{e}(D_2, h_1)^{r_x + cx} \mathbf{e}(w, h_\theta)^{-r_\alpha - c\alpha} \mathbf{e}(w, h_1)^{-r_\gamma - c\gamma} \\
&\quad \times \mathbf{e}(g_2, h_1)^{r_y + cy} \cdot \mathbf{e}(D_2, h_1)^{-cx} \mathbf{e}(w^\alpha, h_\theta h_1^x)^c \\
&\quad \times [\mathbf{e}(g_1 g_2^{-y} w^{-z}, h_1)]^c \mathbf{e}(g_1, h_1)^{-c} \\
&= \mathbf{e}(D_2, h_1)^{s_x} \mathbf{e}(w, h_\theta)^{-s_\alpha} \mathbf{e}(w, h_1)^{-s_\gamma} \\
&\quad \times \mathbf{e}(g_2, h_1)^{s_y} (\mathbf{e}(D_2, h_\theta)/\mathbf{e}(g_1, h_1))^c, \\
R_3 &= g^{r_y} d^{r_\alpha} = g^{r_y + cy} d^{r_\alpha + c\alpha} (g^y d^\alpha)^{-c} = g^{s_y} d^{s_\alpha} D_3^{-c}.
\end{aligned}$$

These relations show that a prover and a verifier computes the same R_1, R_2, and R_3 in an execution of the protocol. It is easy to see that all transcripts correctly generated by the prover are always accepted by the verifier.

Lemma 2: ZKPK$^{\mathsf{SDH}^+}$ *is sound.*

Proof: We show that we can construct an efficient knowledge extractor for the ZKPK$^{\mathsf{SDH}^+}$ protocol. Suppose that an extractor can find a forking point, which is just before a prover receives a challenge c. At the first run of the protocol, the prover sends $(D_1, D_2, D_3, R_1, R_2, R_3)$ and then, to a challenge value c, the prover responds with $s_\alpha, s_x, s_\gamma, s_y$. To another challenge value $c' \neq c$, the prover responds with $s'_\alpha, s'_x, s'_\gamma, s'_y$. If the prover passes the verification test, all three equations (1)-(3) in the final verification step hold for each set of response values. Let $\Delta c = c - c'$, $\Delta s_\alpha = s_\alpha - s'_\alpha$, $\Delta s_x = s_x - s'_x$, $\Delta s_\gamma = s_\gamma - s'_\gamma$, and $\Delta s_y = s_y - s'_y$. Dividing the two instances of Equation (1), that is, 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 $\tilde{\alpha} = \Delta s_\alpha \cdot (\Delta c)^{-1} \pmod p$. Consider Equation (3). Dividing the two instances of this equation we have $D_3^{\Delta c} = g^{\Delta s_y} d^{\Delta s_\alpha}$ and so $D_3 = (g^{\Delta s_y} d^{\Delta s_\alpha})^{\Delta c^{-1}}$. We obtain $\tilde{y} = \Delta s_y (\Delta c)^{-1} \pmod p$ such that $D_3 = g^{\tilde{y}} d_1^{\tilde{\alpha}}$. Finally, dividing the two instances of Equation (2), we obtain $(\mathbf{e}(g_1, h_1)/\mathbf{e}(D_2, h_\theta))^{\Delta c} = \mathbf{e}(D_2, h_1)^{\Delta s_x}$ $\mathbf{e}(w, h_\theta)^{-\Delta s_\alpha} \mathbf{e}(w, h_1)^{-\Delta s_\gamma} \mathbf{e}(g_2, h_1)^{\Delta s_y}$. Let $\tilde{x} = \Delta s_x \cdot (\Delta c)^{-1} \pmod p$, $\tilde{\gamma} = \Delta s_\gamma \cdot (\Delta c)^{-1} \pmod p$, and $\tilde{y} = \Delta s_y \cdot (\Delta c)^{-1} \pmod p$. Taking Δc^{th} roots on both sides of the equation, we have $\mathbf{e}(g_1, h_1)/\mathbf{e}(D_2, h_\theta) = \mathbf{e}(D_2, h_1)^{\tilde{x}} \mathbf{e}(w, h_\theta)^{-\tilde{\alpha}}$ $\mathbf{e}(w, h_1)^{-\tilde{\gamma}} \mathbf{e}(g_2, h_1)^{\tilde{y}}$.

Let $\tilde{z} = \tilde{\alpha}\tilde{x} - \tilde{\gamma} \pmod p$. We have $\tilde{\gamma} = \tilde{\alpha}\tilde{x} - \tilde{z} \pmod p$. One can get $\mathbf{e}(g_1, h_1) = \mathbf{e}(D_2, h_1^{\tilde{x}+\theta}) \mathbf{e}(w^{-\tilde{\alpha}}, h^{\theta+\tilde{x}})$ $\mathbf{e}(w^{\tilde{z}}, h_1) \mathbf{e}(g_2^{\tilde{y}}, h_1)$ by substituting $\tilde{\gamma}$ with $\tilde{\alpha}\tilde{x} - \tilde{z}$ and arranging the above equation. This can be equivalently rewritten as $\mathbf{e}(g_1, h_1) = \mathbf{e}((D_2 w^{-\tilde{\alpha}})^{\tilde{x}+\theta} g_2^{\tilde{y}} w^{\tilde{z}}, h_1)$. Let $\tilde{A} = D_2 w^{-\tilde{\alpha}}$. Then we have $(\tilde{A}, \tilde{x}, \tilde{y}, \tilde{z})$ satisfying $\mathbf{e}(g_1, h_1) = \mathbf{e}(\tilde{A}^{\tilde{x}+\theta} g_2^{\tilde{y}} w^{\tilde{z}}, h_1)$. This implies $g_1 g_2^{-\tilde{y}} w^{-\tilde{z}} = \tilde{A}^{\tilde{x}+\theta}$ because the bilinear map $\mathbf{e}$ is non-degenerated and $\mathbf{e}(\cdot, h_1)$ for a fixed $h_1 \in \mathbb{G}_2$ is injective. Therefore the extractor obtains a valid SDH$^+$ tuple $(\tilde{A}, \tilde{x}, \tilde{y}, \tilde{z})$. In addition, $\tilde{A}$ and $g^{\tilde{y}}$ in this SDH$^+$ tuple are the same as those in the ciphertext (D_1, D_2, D_3).

Lemma 3: For an honest verifier, ZKPK$^{SDH^+}$ can be simulated under the XDH assumption.

Proof: We show that we can build a polynomial-time simulator that outputs transcripts of the proposed ZKPK$^{\mathsf{SDH}^+}$ protocol. First, the simulator picks $A, \hat{Y} \xleftarrow{R} \mathbb{G}_1$, and $\alpha \xleftarrow{R} \mathbb{Z}_p^*$. It sets $D_1 \leftarrow u^\alpha$, $D_2 \leftarrow A w^\alpha$, and $D_3 \leftarrow \hat{Y} d^\alpha$. Under the XDH assumption and a standard hybrid argument, the tuple (D_1, D_2, D_3) is drawn from a distribution that is indistinguishable from the distribution of transcripts output by a prover.

Since the simulation does not require the knowledge of A, x, y, z or α, it can also be used even when D_1, D_2, D_3 are pre-specified. When (D_1, D_2, D_3) is a random ciphertext of $(A, \hat{Y})$, 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 $s_\alpha, s_\gamma, s_x, s_y \xleftarrow{R} \mathbb{Z}_p^*$. It then computes commitment values, $R_1 \leftarrow u^{s_\alpha} D_1^{-c}$, $R_2 \leftarrow \mathbf{e}(D_2, h_1)^{s_x} \mathbf{e}(w, h_\theta)^{-s_\alpha} \mathbf{e}(w, h_1)^{-s_\gamma} \mathbf{e}(g_2, h_1)^{s_y} (\mathbf{e}(D_2, h_\theta)/$ $\mathbf{e}(g_1, h_1))^c$, and $R_3 \leftarrow g^{s_y} d^{s_\alpha} D_3^{-c}$. The simulator outputs the transcript $(D_1, D_2, D_3, R_1, R_2, R_3, c, s_\alpha, s_\gamma, s_x, s_y)$. Note that the resulting values satisfy Equations (1)-(3). Hence the overall transcript is drawn from the distribution that is indistinguishable from the distribution of transcripts of the ZKPK$^{\mathsf{SDH}^+}$ under the XDH assumption.

IV. Our Short PS-OL Scheme

In this section we present a PS-OL scheme which yields short signatures. Basically our construction relies on ZKPK$^{\mathsf{SDH}^+}$. A signer obtains a valid SDH$^+$ tuple, $(x, y, z, A = (g_1 g_2^{-y} w^{-z})^{\frac{1}{\theta+x}})$ as a signing key at the joining process. To generate a signature, the signer proves possession of a valid SDH$^+$ tuple and also the equality of two exponents, i.e., $\log_{g_2} g_2^y$ and $\log_g g^y$. In other words, a signature includes encryption of a SDH$^+$ tuple and a linking value, y associated with the SDH$^+$ tuple. No information to identify a user or link signatures is not revealed publicly from a signature. However, the information can be appropriately recovered with an opening key or a linking key.

Our scheme consists of a setup algorithm, a group membership issuing algorithm which involves two sub-algorithms UserJoin and Issue to generate a user signature key, and signing, verifying, opening, linking, and revocation algorithms. SetUp. For given a security parameter, proceed as follows. Generate a tuple of algebraic groups of prime order p, i.e., $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and a bilinear map $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Pick random $h_1 \stackrel{R}{\leftarrow} \mathbb{G}_2 \setminus \{1_{\mathbb{G}_2}\}$ and $g, g_1, g_2, u \stackrel{R}{\leftarrow} \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$, and $\eta, \xi, \theta \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. Compute $w = u^{\eta}$, $d = u^{\xi}$, $L = h_1^{\xi}$, and $h_\theta = h_1^{\theta}$. Generate a cryptographic hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.

An initial group public key is $gpk_0 = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, g_1, g_2, u, w, d)$, and the master issuing key, $mik = \theta$, the master opening key, $mok = (\eta, \xi)$, and the master linking key, $mlk = L$. In the group public key, g_1, g_2, u, w, d will be updated per revocation. UserJoin/Issue. Two algorithms, UserJoin, which is run by a joining user, and Issue, which is run by the Issuer, interactively perform the following protocol to generate a user signature key $\mathbf{usk}[i] = (x_i, y_i, z_i, A_i = (g_1 g_2^{-y_i} w^{-z_i})^{\frac{1}{\theta + x_i}})$.

- UserJoin. Pick a secret key $\mathbf{sk}_{ID} = z \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and compute its corresponding public key $Z = \mathbf{upk}_{ID} = w^z$. Pick $\tilde{r} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and compute $\tilde{W} = w^{\tilde{r}}$. Compute $\tilde{c} = \mathcal{H}(M_{JR}, w, Z, \tilde{W})$ and $\tilde{s} = \tilde{r} + \tilde{c}z \pmod p$ where M_{JR} is a pre-defined message that is used to request a joining process. Let $\tilde{T}_{ID} = (Z, \tilde{s}, \tilde{c})$. Send (JOIN-REQUEST, ID, $\tilde{T}_{ID}$) to Issue. Assume that $Z = \mathbf{upk}_{ID}$ is cryptographically bounded with the identity ID through an additional technique such as a certificate.
- Issue. Upon the receipt of the join-request message (JOIN-REQUEST, ID, $\tilde{T}_{ID} = (Z, \tilde{s}, \tilde{c})$), compute $\tilde{c}' = \mathcal{H}(M_{JR}, w, Z, w^{\tilde{s}} Z^{-\tilde{c}})$ and check if $\tilde{c} = \tilde{c}'$. If the equality holds then check if ID is registered, that is, there exists j with $\mathsf{REG}[j] = (ID, \ldots)$ on the registration list $\mathbf{REG} = (\mathsf{REG}[1], \ldots, \mathsf{REG}[n-1])$. If no such j exists then pick $x, y \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and compute $A = (g_1 g_2^{-y} w^{-z})^{\frac{1}{\theta + x}}$. Otherwise, find y from $\mathsf{REG}[j] = (ID, g^y, A, x, y, \mathbf{upk}[j])$, pick $x \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, and compute $A = (g_1 g_2^{-y} Z^{-1})^{\frac{1}{\theta + x}}$, and then make $\mathsf{REG}[j]$ empty. Add $\mathsf{REG}[n] = (ID, g^y, A, x, y, \mathbf{upk}[n] = \mathbf{upk}_{ID}, Y_n = g_2^y, X_n = h_1^x)$ to **REG**, and send (n, A, x, y) to UserJoin.
- UserJoin. After receiving the message (n, A, x, y) for some non-negative integer n, if the equality $e(A, h_\theta h_1^x) = e(g_1 g_2^{-y} w^{-z}, h_1)$ holds then UserJoin keeps $\mathbf{usk}[i] = (\lambda = 0, x, y, z, A = (g_1 g_2^{-y} w^{-z})^{\frac{1}{\theta + x}})$ secret as a user signature key.

GSig. For given a user signature key $\mathbf{usk}[i] = (\lambda, x, y, z, A)$ and a message M, proceed as follows:

- Call Update-usk of the revocation algorithm with $\mathbf{usk}[i]$ and obtain $gpk_{\hat{\lambda}} = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, \hat{g}_1, \hat{g}_2, \hat{u}, \hat{w}, \hat{d})$ and its corresponding user signature key $\mathbf{usk}[i] = (\hat{\lambda}, x, y, z, \hat{A})$.
- Pick $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computes $\gamma = x\alpha - z \pmod p$, $D_1 = \hat{u}^{\alpha}$, $D_2 = \hat{A}\hat{w}^{\alpha}$, and $D_3 = g^y \hat{d}^{\alpha}$.
- Pick $r_\alpha, r_x, r_\gamma, r_y \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and compute $R_1 = \hat{u}^{r_\alpha}$, $R_2 = \mathbf{e}(D_2, h_1)^{r_x} \mathbf{e}(\hat{w}, h_\theta)^{-r_\alpha} \mathbf{e}(\hat{w}, h_1)^{-r_\gamma} \mathbf{e}(\hat{g}_2, h_1)^{r_y}$, and $R_3 = g^{r_y} \hat{d}^{r_\alpha}$.
- Compute $c = \mathcal{H}(M, D_1, D_2, D_3, R_1, R_2, R_3)$, and $s_\alpha = r_\alpha + c\alpha \pmod p$, $s_x = r_x + cx \pmod p$, $s_\gamma = r_\gamma + c\gamma \pmod p$, and $s_y = r_y + cy \pmod p$.
- Finally, output a signature $\sigma = (\hat{\lambda}, D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$.

GVfy. For given $(M, \sigma = (\lambda', D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y))$, proceed as follows:

- Call Update-gpk of the revocation algorithm with gpk_0, $(\lambda', \mathbf{RL})$ and obtain $gpk_{\lambda'} = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, \hat{g}_1, \hat{g}_2, \hat{u}, \hat{w}, \hat{d})$.
- Compute $R_1 = \hat{u}^{s_\alpha} D_1^{-c}$, $R_2 = \mathbf{e}(D_2, h_1)^{s_x} \mathbf{e}(\hat{w}, h_\theta)^{-s_\alpha} \mathbf{e}(\hat{w}, h_1)^{-s_\gamma} \cdot \mathbf{e}(\hat{g}_2, h_1)^{s_y} (\mathbf{e}(D_2, h_\theta)/\mathbf{e}(\hat{g}_1, h_1))^c$, and $R_3 = g^{s_y} \hat{d}^{s_\alpha} D_3^{-c}$.
- If $c = \mathcal{H}(M, D_1, D_2, D_3, R_1, R_2, R_3)$ then output 1 (valid), and otherwise, 0 (invalid).

Open. For given $mok = (\eta, \xi)$, **REG** and valid (σ, M), proceed as follows:

- Call GVfy to check if the given signature σ is valid.
- If the signature is not valid then output $\perp$. Otherwise, proceed as follows. Recover $g^y = D_3 \cdot D_1^{-\xi}$ (and alternatively $A = D_2 \cdot D_1^{-\eta}$). Using a binary search on the registration list **REG**, find i such that $g^y = g^{y_i}$. If such i does not exist then return $(i = 0, *)$. Otherwise, find the corresponding $\mathbf{upk}[i] = Z_i = w^{z_i}$ and $\mathsf{PI}_i = (Y_i = g_2^{y_i}, X_i = h_1^{x_i})$. Next, pick $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and compute $K_{\mathsf{open}} = D_1^{\eta}$, $V_1 = u^r$, $V_2 = D_1^r$, $c_{\mathsf{open}} = \mathcal{H}(\sigma, g, K_{\mathsf{open}}, V_1, V_2)$, and $s_{\mathsf{open}} = r + c_{\mathsf{open}}\eta \pmod p$. Output i and a proof $\tau = ((K_{\mathsf{open}}, c_{\mathsf{open}}, s_{\mathsf{open}}), \mathsf{PI}_i)$ where $\mathsf{PI}_i = (Y_i, X_i)$.

Assume that PI_i is cryptographically bounded with a registered identity through an additional technique such as a certificate. Judge. For valid (σ, M), $\mathbf{upk}[i] = Z_i$, a proof $\tau = ((K_{\mathsf{open}}, c_{\mathsf{open}}, s_{\mathsf{open}}), \mathsf{PI}_i = (Y_i, X_i))$, proceed as follows.

- Assume that $i \neq 0$ and $\sigma = (\lambda', D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$.
- Call Update-gpk of the revocation algorithm with $(gpk_0, \lambda', \mathbf{RL})$ and obtain $gpk_{\lambda'} = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, \hat{g}_1, \hat{g}_2, \hat{u}, \hat{w}, \hat{d})$.
- Let $\hat{h}_1 = h_1^{\upsilon} \in \mathbb{G}_2$ where $\upsilon = \log_{g_1} \hat{g}_1$ and $g_1 \in gpk_0$. If $c_{\mathsf{open}} = \mathcal{H}(\sigma, g, K_{\mathsf{open}}, u^{s_{\mathsf{open}}} w^{-c_{\mathsf{open}}}, D_1^{s_{\mathsf{open}}} K_{\mathsf{open}}^{-c_{\mathsf{open}}})$ and $\mathbf{e}(D_2 K_{\mathsf{open}}^{-1}, X_i h_\theta) = \mathbf{e}(g_1 Y_i^{-1} Z_i^{-1}, \hat{h}_1)$ then output 1 (valid), and otherwise, 0 (invalid).

Link. For the master linking key $mlk = L$ and two given pairs of signatures and messages, (σ', M') and (σ'', M''), proceed as follows: Let $\sigma' = (\lambda', D_1', D_2', D_3', c', s_\alpha', s_x', s_\gamma', s_y')$ and $\sigma'' = (\lambda'', D_1'', D_2'', D_3'', c'', s_\alpha'', s_x'', s_\gamma'', s_y'')$.

- Call GVfy to check if the given signatures are valid.
- If any of them is not valid then output $\perp$. Otherwise, compute $B_1 = \mathbf{e}(D_3', h_1)\mathbf{e}(D_1', L)^{-1}$ and $B_2 = \mathbf{e}(D_3'', h_1)\mathbf{e}(D_1'', L)^{-1}$. If $B_1 = B_2$ then output 1

(i.e., linked) and otherwise, 0 (i.e., unlinked). Alternatively, $\mathbf{e}(D_3'/D_3'', h_1) \stackrel{?}{=} \mathbf{e}(D_1'/D_1'', L)$ can be used for the above test.[2]

Revocation. The revocation algorithm consists of two sub-algorithms, Update-gpk which updates a group public key, and Update-usk which updates a user signature key. Let the initial group public key be $gpk_0 = (\mathbf{T}, g_1, g_2, u, w, d)$ where $\mathbf{T} = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H})$.

Define $\upsilon_k = (\theta + x_{j_1})(\theta + x_{j_2}) \cdots (\theta + x_{j_k}) \mod p$ where k is a positive integer. Let the revocation list denote $\mathbf{RL} = \{(S_{1,k}, S_{2,k}, S_{3,k}, S_{4,k}, S_{5,k}, x_{j_k}) | k = 1, \ldots, \hat{\lambda}\}$, where $S_{1,k} = g_1^{1/\upsilon_k}$, $S_{2,k} = g_2^{1/\upsilon_k}$, $S_{3,k} = u^{1/\upsilon_k}$, $S_{4,k} = w^{1/\upsilon_k}$, and $S_{5,k} = d^{1/\upsilon_k}$, and x_{j_k} corresponds to the j_k^{th} registered but revoked user.

- Update-usk. For given $\mathbf{usk}[i] = (\lambda, x, y, z, A)$, proceed as follows: Assume that $x \neq x_{j_k}$ for any $k = 1, \ldots, \hat{\lambda}$.
  - If $\hat{\lambda} = \lambda$ then return (λ, x, y, z, A).
  - Else if $\hat{\lambda} > \lambda$, update the key as follows. Compute $gpk_{\hat{\lambda}}$ by calling Update-gpk with $(gpk_0, -1, \mathbf{RL})$.
  - Let $\pi = \hat{\lambda} - \lambda$ and $t_k = j_{\lambda+k}$ for $k = 1, \ldots, \pi$.
  - Compute $\hat{A} = A^{\frac{(-1)^\pi}{\delta_\pi}} \prod_{i=1}^{\pi} [S_{1,i} S_{2,i}^{-y} S_{4,i}^{-z}]^{(-1)^{\pi-i}\delta_{i-1}/\delta_\pi}$, where $\delta_0 = 1$ and $\delta_i = \prod_{k=1}^{i}(x - x_{t_k}) \mod p$. Return $(\hat{\lambda}, x, y, z, \hat{A})$.
- Update-gpk. Given a tuple $(gpk_0, \rho, \mathbf{RL})$, proceed as follows.
  - If $\hat{\lambda} \leq \rho$ then abort.
  - Else if $\rho = -1$ then update gpk_0 to the latest revoked key as follows. Set $\hat{g}_1 \leftarrow S_{1,j_{\hat{\lambda}}}$, $\hat{g}_2 \leftarrow S_{2,j_{\hat{\lambda}}}$, $\hat{u} \leftarrow S_{3,j_{\hat{\lambda}}}$, $\hat{w} \leftarrow S_{4,j_{\hat{\lambda}}}$, and $\hat{d} \leftarrow S_{5,j_{\hat{\lambda}}}$ and output $gpk_{\hat{\lambda}} = (\mathbf{T}, \hat{g}_1, \hat{g}_2, \hat{u}, \hat{w}, \hat{d})$ where $\mathbf{T} = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H})$.
  - If $0 < \rho < \hat{\lambda}$ then update gpk_0 up to the ρ^{th} revoked key as follow. Set $\hat{g}_1 \leftarrow S_{1,\rho}$, $\hat{g}_2 \leftarrow S_{2,\rho}$, $\hat{u} \leftarrow S_{3,\rho}$, $\hat{w} \leftarrow S_{4,\rho}$, and $\hat{d} \leftarrow S_{5,\rho}$. Output the updated $gpk_\rho = (\mathbf{T}, \hat{g}_1, \hat{g}_2, \hat{u}, \hat{w}, \hat{d})$.

V. Security of the PS-OL Scheme

We show that the proposed scheme PS-OL is correct. First, it is obvious that the signing/verifying algorithms are correct because of Lemma 1. Next we show that the open/judge algorithms are correct. We have $V_1 = u^r = u^{r+c_{\mathsf{open}}\eta}(u^\eta)^{-c_{\mathsf{open}}} = u^{s_{\mathsf{open}}} w^{-c_{\mathsf{open}}}$ and $V_2 = D_1^r = D_1^{r+c_{\mathsf{open}}\eta}(D_1^\eta)^{-c_{\mathsf{open}}} = D_1^{s_{\mathsf{open}}} K_{\mathsf{open}}^{-c_{\mathsf{open}}}$, and so $\mathcal{H}(\sigma, g, K_{\mathsf{open}}, u^{s_{\mathsf{open}}} w^{-c_{\mathsf{open}}}, D_1^{s_{\mathsf{open}}} K_{\mathsf{open}}^{-c_{\mathsf{open}}})$ is equal to $\mathcal{H}(\sigma, g, K_{\mathsf{open}}, V_1, V_2)$. We can also show that $D_2 K_{\mathsf{open}}^{-1} = (\hat{A}\hat{w}^\alpha)(\hat{u}^\alpha)^{-\eta} = \hat{A}$ and thus we have $\mathbf{e}(D_2 K_{\mathsf{open}}^{-1}, X_i h_\theta) = \mathbf{e}(\hat{A}, h_1^{x_i+\theta}) = \mathbf{e}((\hat{g}_1 \hat{g}_2^{-y} \hat{w}^{-z})^{\frac{1}{\theta+x_i}}, h_1^{x_i+\theta}) = \mathbf{e}(g_1 Y_i^{-1} Z_i^{-1}, \hat{h}_1)$ because $\upsilon = \log_{h_1} \hat{h}_1 = \log_{g_1} \hat{g}_1 = \log_{g_2} \hat{g}_2 = \log_w \hat{w}$.

The link algorithm correctly works because, for two valid signatures $\sigma' = (\lambda', D_1', D_2', D_3', c', s_{\alpha'}', \cdot, \cdot, \cdot)$ and $\sigma'' = (\lambda'', D_1'', D_2'', D_3'', c'', s_{\alpha''}'', \cdot, \cdot, \cdot)$, we have the following relations: $B_1 = \mathbf{e}(D_3', h_1)\mathbf{e}(D_1', L)^{-1} = \mathbf{e}(D_3'', h_1)\mathbf{e}(D_1'', L)^{-1} = B_2$.

Finally, we show that the revocation algorithm is correct as follows. Let $\mathbf{RL} = \{(S_{1,k}, S_{2,k}, S_{3,k}, S_{4,k}, S_{5,k}, x_{j_k}) | k = 1, \ldots, \hat{\lambda}\}$, where $S_{1,k} = g_1^{1/\upsilon_k}$, $S_{2,k} = g_2^{1/\upsilon_k}$, $S_{3,k} = u^{1/\upsilon_k}$, $S_{4,k} = w^{1/\upsilon_k}$, and $S_{5,k} = d^{1/\upsilon_k}$. If the initial group public key gpk_0 is assumed to be updated up to the $\rho^{\text{th}}(\rho \leq \hat{\lambda})$ revoked key then g_1, g_2, u, w, d in gpk_0 are updated to $\hat{g}_1 = S_{1,\rho} = g_1^{1/\prod_{t=1}^{\rho}(\theta+x_{j_t})}$, $\hat{g}_2 = S_{2,\rho} = g_2^{1/\prod_{t=1}^{\rho}(\theta+x_{j_t})}$, $\hat{u} = S_{3,\rho} = u^{1/\prod_{t=1}^{\rho}(\theta+x_{j_t})}$, $\hat{w} = S_{4,\rho} = w^{1/\prod_{t=1}^{\rho}(\theta+x_{j_t})}$, and $\hat{d} = S_{5,\rho} = d^{1/\prod_{t=1}^{\rho}(\theta+x_{j_t})}$. Assume that a user signature key, i.e., $A = (g_1 g_2^{-y} w^{-z})^{\frac{1}{\theta+x}}$ is updated up to the π^{th} revoked key. Using the mathematical induction, we show that

$$A^{\frac{1}{(\theta+x_{j_1})\cdots(\theta+x_{j_\pi})}} = A^{\frac{(-1)^\pi}{\delta_\pi}} \prod_{i=1}^{\pi} [S_{1,i} S_{2,i}^{-y} S_{4,i}^{-z}]^{\frac{(-1)^{\pi-i}\delta_{i-1}}{\delta_\pi}},$$

where $\delta_0 = 1$ and $\delta_i = \prod_{k=1}^{i}(x - x_{j_k}) \mod p$.

First, the equality holds for $\pi = 1$ because we have

$$\begin{aligned} &A^{\frac{-1}{x-x_{j_1}}} (S_{1,1} S_{2,1}^{-y} S_{4,1}^{-z})^{\frac{1}{x-x_{j_1}}} \\ &= [g_1^{\frac{1}{\theta+x_{j_1}}} g_2^{\frac{-y}{\theta+x_{j_1}}} w^{\frac{-z}{\theta+x_{j_1}}} (g_1^{\frac{1}{\theta+x}} g_2^{\frac{-y}{\theta+x}} w^{\frac{-z}{\theta+x}})^{-1}]^{\frac{1}{x-x_{j_1}}} \\ &= [g_1^{\frac{1}{\theta+x}} (g_2^{\frac{1}{\theta+x}})^{-y} (w^{\frac{1}{\theta+x}})^{-z}]^{\frac{1}{\theta+x_{j_1}}} = A^{\frac{1}{\theta+x_{j_1}}}. \end{aligned}$$

Assume that by the mathematical induction, the equality holds for $\pi - 1$, that is,

$$A^{\frac{1}{(\theta+x_{j_1})\cdots(\theta+x_{j_{\pi-1}})}} = A^{\frac{(-1)^{\pi-1}}{\delta_{\pi-1}}} \prod_{i=1}^{\pi-1} [S_{1,i} S_{2,i}^{-y} S_{4,i}^{-z}]^{\frac{(-1)^{\pi-1-i}\delta_{i-1}}{\delta_{\pi-1}}}.$$

We can show that the equality holds for ω as follows:

$$\begin{aligned} &A^{\frac{(-1)^\pi}{\delta_\pi}} \prod_{i=1}^{\pi} [S_{1,i} S_{2,i}^{-y} S_{4,i}^{-z}]^{\frac{(-1)^{\pi-i}\delta_{i-1}}{\delta_\pi}} \\ &= \left(\left((g_1 g_2^{-y} w^{-z})^{\frac{-1}{\theta+x}} (g_1 g_2^{-y} w^{-z})^{\frac{1}{\theta+x_{j_\pi}}} \right)^{\frac{1}{x-x_{j_\pi}}} \right)^{\frac{1}{(\theta+x_{j_1})\cdots(\theta+x_{j_{\pi-1}})}} \\ &= \left((g_1 g_2^{-y} w^{-z})^{\frac{1}{\theta+x}} \right)^{\frac{1}{(\theta+x_{j_1})\cdots(\theta+x_{j_{\pi-1}})(\theta+x_{j_\pi})}} \\ &= A^{\frac{1}{(\theta+x_{j_1})\cdots(\theta+x_{j_{\pi-1}})(\theta+x_{j_\pi})}}. \end{aligned}$$

Next we prove that PS-OL achieves CPA-anonymity, traceabiltiy, non-frameability, and three requirements for linkability, that is, E-linkability, JP-Unforgeability, and LO-linkability.

Theorem 1: PS-OL is CPA-anonymous under the XDH assumption.

Proof: We will construct an efficient algorithm $\mathcal{S}$ which solves the XDH problem using an adversary $\mathcal{F}$ who breaks the CPA-anonymity of PS-OL.

Let $(\mathbb{G}_1, p, \tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c)$ be a XDH problem given to $\mathcal{S}$. The goal of $\mathcal{S}$ is to decide if $c = ab \pmod p$ holds. We assume that $\mathbb{G}_1$ is associated with a bilinear map $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. To generate the remaining parameters and keys (according to SetUp) for PS-OL, $\mathcal{S}$ proceeds as follows. It first sets $u = \tilde{g}$ and $w = \tilde{g}^a = u^a$, and then selects ξ,

[2]The first algorithm extracts linking indices from group signatures and compares them. If some memory space is available for storing linking indices, many group signatures can be easily handled with the first one. In contrast, the second algorithm directly checks an equality for linking. If a specific signature should be linked (or traced) quickly for a small number of group signatures, the second one is preferable because of low computation.

$\theta \xleftarrow{R} \mathbb{Z}_p$ and $h_1 \xleftarrow{R} \mathbb{G}_2$ and $g_1, g_2 \xleftarrow{R} \mathbb{G}_1$ (uniformly at random). It then computes $d = u^{\xi}$ and $h_\theta = h_1^{\theta}$. The initial group public key $gpk = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, g_1, g_2, u, w, d)$ is given to $\mathcal{F}$. Note that the master opening key, mok is $\eta = a = \log_u w$ which is unknown to $\mathcal{S}$ in the above set up process. By the definition, all keys except mok, that is, $mik = \theta$ and $mlk = h_1^{\xi}$ are given to $\mathcal{F}$.

– *Query I:* Simulator $\mathcal{S}$ 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 simply never repeated by the adversary.
- `Link`: On query $(m_0, \sigma_0, m_1, \sigma_1)$, $\mathcal{S}$ responds with a correct result running Link with the master linking key mlk.
- `SndToU`, `WReg`, `USK`, and `CrptU`: On queries, $\mathcal{S}$ can easily respond because $\mathcal{S}$ is aware of the master issuing key $mik = \theta$ and can correctly generate a user signature key with Join. We omit the details.

– *Challenge:* Adversary $\mathcal{F}$ submits two indices i_0 and i_1, and a message M as a challenge request for CPA-anonymity. Upon receiving (i_0, i_1, M), $\mathcal{S}$ picks a random bit $b \in \{0, 1\}$ and then generates a challenge signature as follows. Let $\mathbf{usk}[i_b] = (\lambda, x, y, z, A)$ be a user signature key corresponding to index i_b. Assume that **RL** be the list of revoked uses that the adversary has picked in an adaptive way. Let $\hat{\lambda}$ be the most-up-to-date revocation index of **RL**.

- Call Update-usk with $\mathbf{usk}[i_b]$ to obtain a updated user signature key denoted by $\mathbf{usk}[i_b] = (\hat{\lambda}, x, y, z, A)$. Also, call Update-gpk with gpk_0 and $(\hat{\lambda}, \mathbf{RL})$ to obtain a updated group public key $gpk_{\hat{\lambda}} = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, \hat{g}_1, \hat{g}_2, \hat{u}, \hat{w}, \hat{d})$ corresponding the updated user signature key.
- Compute $D_1 \leftarrow \tilde{g}^b = u^b$, $D_2 \leftarrow A\tilde{g}^c = Au^c$, $D_3 \leftarrow g^y(\tilde{g}^b)^{\xi} = g^y(\tilde{g}^{\xi})^b = g^y d^b$.
- Pick $s_\alpha, s_x, s_\gamma, s_y, c \xleftarrow{R} \mathbb{Z}_p$.
- Compute $R_1 \leftarrow \hat{u}^{s_\alpha} D_1^{-c}$, $R_2 \leftarrow \mathbf{e}(D_2, h_1)^{s_x}\, \mathbf{e}(\hat{w}, h_\theta)^{-s_\alpha}\, \mathbf{e}(\hat{w}, h_1)^{-s_\gamma} \cdot \mathbf{e}(\hat{g}_2, h_1)^{s_y}\, (\mathbf{e}(D_2, h_\theta)/\mathbf{e}(\hat{g}_1, h_1))^c$, and $R_3 \leftarrow g^{s_y} \hat{d}^{s_\alpha} D_3^{-c}$.
- Set $\mathcal{H}(M, D_1, D_2, D_3, R_1, R_2, R_3)$ to be equal to c.
- Finally, output a signature $\sigma = (\hat{\lambda}, D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$.

Simulator $\mathcal{S}$ returns σ to $\mathcal{F}$ as a challenge.

– *Query II:* Similar to the above query phase, $\mathcal{S}$ responds to $\mathcal{F}$'s oracle queries.

Suppose that $\mathcal{F}$ finally outputs a guess bit $b_{\mathcal{F}}$. In turn, $\mathcal{S}$ outputs $b_{\mathcal{S}} = b_{\mathcal{F}}$ as the solution to the XDH problem.

In the above simulation, if $g^c = g^{ab}$ then σ is a valid group signature (for $\mathbf{usk}[i_b]$) because $D_2 = Au^c = Au^{ab} = Aw^b$ and thus D_1, D_2, and D_3 have the correct form. It is obvious to see that $(c, s_\alpha, s_x, s_\gamma, s_y)$ of the challenge signature is correct because the distribution of these parameters is completely identical to that of the parameters generated in SetUp of the real protocol. Since the XDH instance is turned into a challenge group signature for user i_b, $\mathcal{S}$'s guess will be correct whenever $\mathcal{F}$'s guess is correct. Therefore, if $\mathcal{F}$ succeeds in breaking the anonymity of PS-OL with a certain advantage ε then $\mathcal{S}$ succeeds in solving the XDH problem with the same advantage. Thus, we have the desired result. □

Next we show that PS-OL achieves all requirements for (controllable) linkability, that is, E-linkability, JP-Unforgeability, and LO-linkability.

Theorem 2: PS-OL is E-linkable under the discrete logarithm (DL) assumption.

Proof: We will construct an efficient algorithm $\mathcal{S}$ which solves the discrete logarithm (DL) problem in $\mathbb{G}_1$ using $\mathcal{F}$ who attacks the E-Linkability of PS-OL. Let $(f_1, f_2) \in \mathbb{G}_1 \times \mathbb{G}_1$ be a random instance of the DL problem. The goal of $\mathcal{S}$ is to compute $\chi = \log_{f_1} f_2 \pmod p$. For an initial group public key gpk_0, $\mathcal{S}$ generates related parameters as follows. Let $g_1 = f_1$ and $g_2 = f_2$. It picks $\zeta, \theta, \eta, \xi \xleftarrow{R} \mathbb{Z}_p^*$ and $h_1 \xleftarrow{R} \mathbb{G}_2$, and computes $u = g_1^{\zeta}$, $w = u^{\eta}$, $d = u^{\xi}$, $L = h_1^{\xi}$, and $h_\theta = h_1^{\theta}$. It also generates a cryptographic hash function $\mathcal{H} : \{0, 1\}^* \to \mathbb{Z}_p^*$. Note that the distribution of these parameters is completely identical to that of the real parameters generated in SetUp. All the master secret keys are well defined by $mik = \theta$, $mok = (\eta, \xi)$, and $mlk = L$. The initial group public key $gpk_0 = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, g_1, g_2, u, w, d)$, and 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 simply never repeated by the adversary.
- `SndToU`, `AddU`, `RReg`, `GSig`, `USK`, `CrptU`, and `Revoc`: Queries can be easily responded because $\mathcal{S}$ is aware of the master issuing key mik. We omit the details.

Assume that the adversary $\mathcal{F}$ outputs two tuples (M_0, σ_0) and (M_1, σ_1) such that $(i_b, \tau_{i_b}) \leftarrow \mathsf{Open}(M_b, \sigma_b)$ and $1 = \mathsf{Judge}(M_b, \sigma_b, \mathbf{upk}[i_b], \tau_{i_b})$ for each $b = 0, 1$. Note that i_0 and i_1 indicate signers who are registered in **REG** and their corresponding initial secret values (x_b, y_b, z_b, A_b) have been well defined. Let $A_b = (g_1 g_2^{-y_b} w^{-z_b})^{\frac{1}{\theta + x_b}}$ for each $b = 0, 1$. Next we consider two cases according to the E-linkability game.

- The first case is that $0 = \mathsf{Link}(mlk, M_0, \sigma_0, M_1, \sigma_1)$ but $i_0 \neq i_1$. By the definition of Link, we have $\mathbf{e}(g^{y_0}, h_1) \neq \mathbf{e}(g^{y_1}, h_1)$ and so $y_0 \neq y_1$. The condition of $i_0 = i_1$ implies $w^{z_0} = \mathbf{upk}[i_0] = \mathbf{upk}[i_1] = w^{z_1}$ and so $z_0 = z_1$. Since a single A_b is generated and uniquely assigned for each $\mathbf{upk}[i_b]$, we also have $A_0 = A_1$. $\mathcal{S}$ outputs $\chi' = \frac{(x_0 - x_1)(1 - z_1\zeta\eta)}{\theta(y_1 - y_0) + x_0 y_1 - x_1 y_0} \pmod p$ as a solution to the given DL problem. Note that $f_1^{\chi'} = f_2$ holds, because $A_0 = A_1$, i.e., $(g_1 g_2^{-y_0} w^{-z_0})^{\frac{1}{\theta + x_0}} = (g_1 g_2^{-y_1} w^{-z_1})^{\frac{1}{\theta + x_1}}$ implies $(\theta + x_1)(1 - y_0\chi' - z_0\zeta\eta) = (\theta + x_0)(1 - y_1\chi' - z_1\zeta\eta) \pmod p$ and so we have $\chi' = \frac{(\theta + x_0)(1 - z_1\zeta\eta) - (\theta + x_1)(1 - z_0\zeta\eta)}{\theta(y_1 - y_0) + x_0 y_1 - x_1 y_0} = \frac{(x_0 - x_1)(1 - z_1\zeta\eta)}{\theta(y_1 - y_0) + x_0 y_1 - x_1 y_0} \pmod p$. For random $y_0, y_1, x_0, x_1 \in \mathbb{Z}_p^*$, we have $\theta(y_1 - y_0) + x_0 y_1 - x_1 y_0 \neq 0 \pmod p$ with non-negligible probability.
- The second case is $1 = \mathsf{Link}(mlk, M_0, \sigma_0, M_1, \sigma_1)$ but $i_0 \neq i_1$. By the definition of the Link algorithm, we have $\mathbf{e}(g^{y_0}, h_1) = \mathbf{e}(g^{y_1}, h_1)$ and so $y_0 = y_1$.

By the construction of the scheme, y_b is selected from $\mathbb{Z}_p^*$ uniformly at random and uniquely assigned to each user signature key and A_b. Thus, since $y_0 = y_1$, A_0 and A_1 must be the same with overwhelming probability. On the other hand, the condition of $i_0 \neq i_1$ implies $w^{z_0} = \mathbf{upk}[i_0] \neq \mathbf{upk}[i_1] = w^{z_1}$ and so $z_0 \neq z_1$. We can also have $x_0 \neq x_1$ because, if $x_0 = x_1$ then $A_0 = (g_1 g_2^{-y_0} w^{-z_0})^{\frac{1}{\theta+x_0}} \neq (g_1 g_2^{-y_1} w^{-z_1})^{\frac{1}{\theta+x_1}} = A_1$ under $y_0 = y_1$ and $z_0 \neq z_1$, and this would contradict the prior condition $A_0 = A_1$. Finally, $\mathcal{S}$ outputs $\chi' = \frac{(\theta+x_0)(1-z_1\zeta\eta)-(\theta+x_1)(1-z_0\zeta\eta)}{y_1(x_0-x_1)} \pmod p$ as a solution to the given DL problem. Note that $f_1^{\chi'} = f_2$ holds, because $A_0 = A_1$, i.e., $(g_1 g_2^{-y_0} w^{-z_0})^{\frac{1}{\theta+x_0}} = (g_1 g_2^{-y_1} w^{-z_1})^{\frac{1}{\theta+x_1}}$ implies $(\theta + x_1)(1 - y_0\chi' - z_0\zeta\eta) = (\theta + x_0)(1 - y_1\chi' - z_1\zeta\eta) \pmod p$ and so we have $\chi' = \frac{(\theta+x_0)(1-z_1\zeta\eta)-(\theta+x_1)(1-z_0\zeta\eta)}{\theta(y_1-y_0)+x_0y_1-x_1y_0} = \frac{(\theta+x_0)(1-z_1\zeta\eta)-(\theta+x_1)(1-z_0\zeta\eta)}{x_0y_1-x_1y_0} \pmod p$. Note that $x_0y_1 - x_1y_0 = y_1(x_0 - x_1) \neq 0 \pmod p$ because $x_0 \neq x_1 \pmod p$ and $y_0(= y_1) \neq 0 \pmod p$ by the construction of PS-OL. □

Theorem 3: PS-OL is JP-Unforgeable under the DL assumption.

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 PS-OL. Let $(u, w) \in \mathbb{G}_1 \times \mathbb{G}_1$ be a random instance of the DL problem. The goal of $\mathcal{S}$ is to compute $\eta^* = \log_u w$. Except (u, w), $\mathcal{S}$ generates the remaining parameters and keys according to the procedure of SetUp. Assume that an initial group public key $gpk_0 = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, g_1, g_2, u, w, d)$, and $mik = \theta$ and $mlk = U(= h_1^\xi)$ 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 simply never repeated by the adversary.
- SndToU, WReg, GSig, Open, USK, CrptU, and Revoc queries can be easily responded because $\mathcal{S}$ can make use of the master issuing key mik to generate a user signature key $\mathbf{usk}[i]$. We omit the details.

– *Challenge:* 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 $\sigma^* = (\lambda^*, D_1^*, D_2^*, D_3^*, c, s_\alpha, s_x, s_\gamma, s_y)$ 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 $(i^*, \tau^* = (K^*_{\mathsf{open}}, c^*_{\mathsf{open}}, s^*_{\mathsf{open}})$ such that $\mathsf{Judge}(M^*, \sigma^*, \mathbf{upk}[i], \tau^*) = 1$. By definition, (i^*, τ^*) satisfies the first verification condition of the judge algorithm as follows:

$$c^*_{\mathsf{open}} = \mathcal{H}(\sigma, u, K^*_{\mathsf{open}}, u^{s^*_{\mathsf{open}}} w^{-c^*_{\mathsf{open}}}, (D_1^*)^{s^*_{\mathsf{open}}} {K^*_{\mathsf{open}}}^{-c^*_{\mathsf{open}}}).$$

Note that the proof τ^* returned from the opening algorithm includes a (non-interactive) ZKPK of the discrete logarithm $\eta^* = \log_u w$. Using a similar argument for the forking lemma [48] we can obtain another proof $\tau' = (K^*_{\mathsf{open}}, c'_{\mathsf{open}}, s'_{\mathsf{open}})$. $\mathcal{S}$ then computes $\eta^* = (s^*_{\mathsf{open}} - s'_{\mathsf{open}})(c^*_{\mathsf{open}} - c'_{\mathsf{open}})^{-1} \pmod p$ and outputs η as the answer to the given DL problem. It is obvious to see $\eta^* = \log_u w$. Even when u, w are updated to u', w' by a revocation we have $\eta^* = \log_u w = \log_{u'} w'$ because exponentiation with a same exponent is done on both u and w through the computation of the revocation. □

Theorem 4: PS-OL is LO-linkable under the XDH assumption.

Proof: We will construct an efficient algorithm $\mathcal{S}$ which solves the XDH problem using an adversary $\mathcal{F}$ who breaks the LO-linkability of PS-OL. Assume that an XDH problem $(\mathbb{G}_1, p, u, w, u^{a^*}, T)$ is given to $\mathcal{S}$ where $a^* \xleftarrow{R} \mathbb{Z}_p^*$. The goal of $\mathcal{S}$ is to guess whether $T = w^{a^*}$ 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. In the above construction $\mathcal{S}$ picked $mik = \theta$ and $mlk = U(= h_1^\xi)$, in other words, is aware of mik and mlk. But $mok = (\eta = \log_u w, \xi)$ is implicitly defined in the above set-up and so $\mathcal{S}$ cannot learn $\eta = \log_u w$. The initial group public key $gpk_0 = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, g_1, g_2, u, w, d)$, and mik 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 $\mathcal{S}$ is aware of the master issuing key mik and is able to generate any signing key with mik. We omit the details.

– *Challenge:* 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 initial (secret) values corresponding to i_0 and i_1, respectively. Assume that keys have been updated according to the revocation list $\mathbf{RL}$ that the adversary adaptively makes. Denote by $\hat{A}_{i_0}$ and $\hat{A}_{i_1}$ the updated values of A_{i_0} and A_{i_1}, which can be obtained by the key update with $\mathbf{RL}$, respectively. Let $\varpi = \log_{A_{i_b}} \hat{A}_{i_b}$. The updated group public key is $gpk_{\hat{\lambda}} = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, \hat{g}_1, \hat{g}_2, \hat{u}, \hat{w}, \hat{d})$. The values y_{i_0} and y_{i_1} related to linking information are selected uniformly at random by construction. Simulator $\mathcal{S}$ picks a random bit $b \in \{0, 1\}$ and proceeds as follows: (1) Set $D_1 \leftarrow u^{a^*}$, and compute $D_2 \leftarrow \hat{A}_{i_b} \cdot T$ and $D_3 \leftarrow g^{y_{i_0}} \cdot D_1^\xi$. (2) Pick $c \xleftarrow{R} \mathbb{Z}_p^*$ and $s_{\alpha(=\varpi^{-1}a^*)}, s_x, s_\gamma, s_y \xleftarrow{R} \mathbb{Z}_p^*$. (3) Compute $R_1 \leftarrow \hat{u}^{s_\alpha} D_1^{-c}$, $R_2 \leftarrow \mathbf{e}(D_2, h_1)^{s_x} \mathbf{e}(\hat{w}, h_\theta)^{-s_\alpha} \mathbf{e}(\hat{w}, h_1)^{-s_\gamma} \mathbf{e}(\hat{g}_2, h_1)^{s_y} (\mathbf{e}(D_2, h_\theta)/\mathbf{e}(\hat{g}_1, h_1))^c$, and $R_3 \leftarrow g^{s_y} \hat{d}^{s_\alpha} D_3^{-c}$. The signature $\sigma = (\hat{\lambda}, D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$ 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 $T = w^{a^*}$, and else if $b \neq b_\mathcal{F}$ then output 0, which means T is random.

We can show that, if $T = w^{a^*}$ then the presented simulation is perfect because $D_1 = u^{a^*} = (u^\varpi)^{\varpi^{-1}a^*} = (u^\varpi)^\alpha$ and

s_α is defined as a random value which is independent of $\alpha = \varpi^{-1}a^* \pmod p$. 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 randomly picked, $\mathcal{F}$ would get a negligible advantage in distinguishing the target indices. Therefore, if $\mathcal{F}$ succeeds in breaking the LO-linkability of PS-OL with a certain advantage then $\mathcal{S}$ succeeds in solving the XDH problem with at least the advantage. □

Theorem 5: PS-OL is traceable under the q-SDH assumption.

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

First, $\mathcal{S}$ generates the parameters of PS-OL as follows. First, it picks $\nu, \nu_0, \nu_2, \nu_3, \nu_h, \eta, \xi \xleftarrow{R} \mathbb{Z}_p^*$ 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 coefficients $c_k, c_{ik}, c_{ijk} \in \mathbb{Z}_p$ can be efficiently computed using $x_1, \ldots, x_q$. To set g_1, g_2, g, u, h_1, it computes them as follows: $g_1 = \left(\prod_{i=0}^{q} g_{1,i}^{c_i}\right)^{\nu} \prod_{i=1}^{q} (\prod_{j=0, j\neq i}^{q-1} g_{1,j}^{c_{ij}})^{y_i} = \hat{g}_1^{\nu F(\theta) + \nu_2 \sum_{k=1}^{q} y_k F_k(\theta)}$, $g = g_1^{\nu_0}$, $g_2 = \left(\prod_{i=1}^{q} \prod_{j=0,j\neq i}^{q-1} g_{1,j}^{c_{ij}}\right)^{\nu_2} = \hat{g}_1^{\nu_2 \sum_{k=1}^{q} F_k(\theta)}$, $u = \left(\prod_{i=0}^{q} g_{1,i}^{c_i}\right)^{\nu_3} = \hat{g}_1^{\nu_3 F(\theta)}$, $h_1 = \hat{h}_1^{\nu_h}$, $h_\theta = (\hat{h}_1^{\theta})^{\nu_h}$, $w = u^{\eta}$, $d = u^{\xi}$. The initial group public key $gpk_0 = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, g_1, g_2, u, w, d)$ is published. We note that $mok = (\eta, \xi)$ and $mlk = h_1^{\xi}$ are known to $\mathcal{S}$ while $mik = \theta$ is unknown. However, we can show that $\mathcal{S}$ correctly responds to $\mathcal{F}$'s queries.

Next we only focus on how to respond to AddU[3] oracle queries assuming that the adversary adds all users to a group. Other queries, that is, SndToI, RReg, USK, CrptU, and Revoc queries can be easily answered if AddU queries are well presented.

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

- $\mathcal{H}$: On a hash query, return a random value in $\{0,1\}^{\lambda}$. Assume that hash queries are never repeated.
- AddU: On query (i), $\mathcal{S}$ and $\mathcal{F}$ (running UserJoin) proceed as follows.
  (a) Upon receipt of a join-request message (JOIN-REQUEST, ID_i, $\tilde{T}_{ID} = (Z_i = w^{z_i}, \tilde{s}, \tilde{c})$) where $z_i \in \mathbb{Z}_p^*$ is chosen by $\mathcal{F}$, $\mathcal{S}$ checks the validity of this join-request message following the procedure of the protocol. $\mathcal{S}$ then picks $r \xleftarrow{R} \mathbb{Z}_p^*$ and computes $R_i = \hat{g}_1^{\nu_3 F_i(\theta) r \eta} = \hat{g}_1^{\frac{\nu_3 F(\theta)\eta r}{\theta + x_i}} = (w^{\frac{1}{\theta+x_i}})^r$ and sends R_i to $\mathcal{A}$.
  (b) $\mathcal{A}$ computes $\hat{Z}_i = R_i^{z_i}$ and a ZKPK proof, $\hat{T}_{\text{JOIN-EQ},i}$ to prove $\log_w Z_i = \log_{R_i} \hat{Z}_i$, and sends $(\hat{Z}_i, \hat{T}_{\text{JOIN-EQ},i})$ to $\mathcal{S}$.
  (c) Upon receipt of $(\hat{Z}_i, \hat{T}_{\text{JOIN-EQ},i})$, $\mathcal{S}$ checks the validity of $\hat{T}_{\text{JOIN-EQ},i}$ and computes $V = (R_i^{z_i})^{r^{-1}} = w^{\frac{z_i}{\theta+x_i}}$ and $A_i = \hat{g}_1^{\nu F_i(\theta) + \nu_2 \sum_{k=1}^{q} (y_k - y_i) F_{k,i}(\theta)} V^{-1} = (g_1 g_2^{-y_i} w^{-z_i})^{\frac{1}{\theta+x_i}}$, and outputs (i, A_i, x_i, y_i).
  (d) The remaining steps can be finished in an obvious manner.
- SndToI, RReg, USK, CrptU, and Revoc: These queries can be easily answered from AddU queries. We omit the detalis.

Assume that $\mathcal{F}$ finally outputs a forgery σ^* on M^*, and $(i^*, \tau^*) \leftarrow \mathsf{Open}(mok, \mathbf{REG}, M^*, \sigma^*)$. Let g^{y^*} and A^* be values recovered from σ^* under mok. Note that, using an argument which is similar to the forking lemma [3], [25] for the forged signature σ^*, we can obtain $(\alpha^*, x^*, y^*, \gamma^*)$ corresponding to A^* and $z^* = \alpha^* x^* - \gamma^* \bmod p$. Consider two cases in the traceability game, i.e., (1) $i^* = 0$ and (2) $i^* \in \{1, \ldots, q\}$ but $0 = \mathsf{Judge}(M^*, \sigma^*, \mathbf{upk}[i^*] = Z_{i^*} (= w^{z_{i^*}}), \tau^*)$. Next we show that we can solve the q-SDH problem for each case.

• First, assume $i^* = 0$, i.e., i^* does not indicate any user in **REG**.

1) Assume that $A^* \neq A_i$ for any $i \in \{1, \ldots, q\}$.

(a) Assume that $x^* \neq x_i$ for any $i \in \{1, \ldots, q\}$. Let $Q(\theta) = (\nu - z^*\eta\nu_3)F(\theta) + \nu_2 \sum_{k=1}^{q} (y_k - y^*) F_k(\theta) \bmod p$. Note that $A^* = (g_1 g_2^{-y^*} w^{-z^*})^{\frac{1}{\theta+x^*}} = (\hat{g}_1^{(\nu - z^*\eta\nu_3)F(\theta) + \nu_2 \sum_{k=1}^{q} (y_k - y^*) F_k(\theta)})^{\frac{1}{\theta+x^*}} = \hat{g}_1^{\frac{Q(\theta)}{\theta+x^*}}$. Here $\frac{Q(\theta)}{\theta+x^*}$ can be expressed as $\frac{Q(\theta)}{\theta+x^*} = Q'(\theta) + \frac{C'}{\theta+x^*} \bmod p$ 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 $\hat{g}_1^{\frac{1}{\theta+x^*}} = (A^* \hat{g}_1^{-Q'(\theta)})^{C'^{-1}}$ and outputs $(\hat{g}_1^{\frac{1}{\theta+x^*}}, x^*)$ as a solution of the q-SDH problem.

(b) Assume that $x^* = x_i$ for some $i \in \{1, \ldots, q\}$.
(1) Assume that $z^* \neq z_i$. First, compute $T = (A_i^{z^*} A^{*-z_i})^{(z^* - z_i)^{-1}} = ((g_1 g_2^{-y_i} w^{-z_i})^{\frac{z^*}{\theta+x_i}} (g_1 g_2^{-y^*} w^{-z^*})^{\frac{-z_i}{\theta+x_i}})^{(z^*-z_i)^{-1}} = (g_1 g_2^{\frac{-y_i z^* + y^* z_i}{z^* - z_i}})^{\frac{1}{\theta+x_i}} = (\hat{g}_1^{\nu F(\theta) + \nu_2 \sum_{k=1}^{q} y_k F_k(\theta)} \hat{g}_1^{\frac{-y_i z^* + y^* z_i}{z^* - z_i} \nu_2 \sum_{k=1}^{q} F_k(\theta)})^{\frac{1}{\theta+x_i}}$. Let $\rho = \frac{-y_i z^* + y^* z_i}{z^* - z_i} \nu_2 \bmod p$ and $Q(\theta) = \nu F(\theta) + \sum_{j=1}^{q} (\nu_2 y_j + \rho) F_j(\theta) \bmod p$. We have $T = \hat{g}_1^{\frac{Q(\theta)}{\theta+x_i}}$. Here, $Q(\theta)/(\theta + x_i) = \nu F_i(\theta) + (\sum_{j=1, j\neq i}^{q} (\nu_2 y_j + \rho) F_{j,i}(\theta)) + \frac{(\nu_2 y_i + \rho) F_i(\theta)}{\theta + x_i}$ 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. $\mathcal{S}$ computes $\hat{g}_1^{\frac{1}{\theta+x_i}} = (T \hat{g}_1^{-Q'(\theta)})^{C'^{-1}}$ and

[3]For the proof of this theorem, the joining process of PS-OL is slightly modified by adding a zero-knowledge proof of knowledge for the equality of two discrete logarithms. This modification can be easily done with the original joining process.

outputs $(\hat{g}_1^{\frac{1}{\theta+x_i}}, x_i)$ as a solution of the q-SDH problem. (2) Assume that $z^* = z_i$. We have $y^* \neq y_i$ because of $A^* \neq A_i$ and $x^* = x_i$. Compute $T = (A_i A^{*-1})^{(-y_i+y^*)^{-1}} = ((g_2^{-y_i+y^*})^{\frac{1}{\theta+x_i}})^{(-y_i+y^*)^{-1}} = g_2^{\frac{1}{\theta+x_i}} = (\hat{g}_1^{\nu_2 \sum_{k=1}^{q} F_k(\theta)})^{\frac{1}{\theta+x_i}} = \hat{g}_1^{\frac{Q(\theta)}{\theta+x_i}}$ where $Q(\theta) = \nu_2 \sum_{k=1}^{q} F_k(\theta)$. Here, $Q(\theta)/(\theta + x_i) = (\sum_{k=1,k\neq i}^{q} \nu_2 F_{k,i}(\theta)) + \frac{\nu_2 F_i(\theta)}{\theta+x_i}$ 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. $\mathcal{S}$ computes $\hat{g}_1^{\frac{1}{\theta+x_i}} = (T \hat{g}_1^{-Q'(\theta)})^{C'^{-1}}$ and outputs $(\hat{g}_1^{\frac{1}{\theta+x_i}}, x_i)$ as a solution of the q-SDH problem.

2) Assume that $A^* = A_i$ for some $i \in \{1, \ldots, q\}$.

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

(b) Assume that $x^* = x_i$. We have $y^* \neq y_i$ and $z^* \neq z_i$ because $z^* = z_i$ would imply $y^* = y_i$ due to $A^* = A_i$ and violate the assumption $i^* = 0$. $g_2^{-y^*} w^{-z^*} = g_2^{-y_i} w^{-z_i}$ holds because $(g_1 g_2^{-y^*} w^{-z^*})^{\frac{1}{\theta+x^*}} = A^* = A_i = (g_1 g_2^{-y_i} w^{-z_i})^{\frac{1}{\theta+x_i}}$. This result implies $g_2 = w^{(-z_i+z^*)/(-y^*+y_i)}$ and $(-z_i + z^*)/(-y^* + y_i) = \log_w g_2 = \frac{\nu_2 \sum_{k=1}^{q} F_k(\theta)}{\nu_3 \eta F(\theta)} \bmod p$ and so $\frac{-z_i+z^*}{-y^*+y_i}\nu_3 F(\theta) - \nu_2 \sum_{k=1}^{q} F_k(\theta) = 0 \bmod p$. For any i, let $Q(\theta) = \frac{-z_i+z^*}{-y^*+y_i}\nu_3 F(\theta) - \nu_2 \sum_{k=1}^{q} F_k(\theta) \bmod p$ and $\frac{Q(\theta)}{\theta+x_i} = \frac{-z_i+z^*}{-y^*+y_i}\nu_3 F_i(\theta) - (\nu_2 \sum_{k=1,k\neq i}^{q} F_{k,i}(\theta)) + \frac{\nu_2 F_i(\theta)}{\theta+x_i} \bmod p$. Note that $\frac{Q(\theta)}{\theta+x_i}$ 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, $\mathcal{S}$ computes $\hat{g}_1^{\frac{1}{\theta+x_i}} = (\hat{g}_1^{Q'(\theta)})^{-C'^{-1}}$ and outputs $(\hat{g}_1^{\frac{1}{\theta+x_i}}, x_i)$ 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 $0 = \mathsf{Judge}(M^*, \sigma^*, \mathbf{upk}[i] = Z_i (= w^{z_i}), \tau^*)$. Let $\sigma^* = (\hat{\lambda}, D_1^*, D_2^*, D_3^*, c^*, s_\alpha^*, s_x^*, s_y^*, s_\gamma^*)$ and $\tau^* = (i^*, (K_{\mathsf{open}}^*, c_{\mathsf{open}}^*, s_{\mathsf{open}}^*), \mathsf{PI}_i = (Y_i, X_i))$. It is obvious that the first verification of the Judge algorithm always holds because τ^* was generated by the Open algorithm and so is correct. Suppose that the second condition does not hold; that is, $\mathbf{e}(D_2^* K_{\mathsf{open}}^{-1}, X_i h_\theta) \neq \mathbf{e}(g_1 Y_i^{-1} Z_i^{-1}, h_1)$. This implies that $\mathbf{e}(D_2^* K_{\mathsf{open}}^{-1}, h_1^{\theta+x_i}) \neq \mathbf{e}(g_1 g_2^{-y_i} g_3^{-z_i}, h_1)$ and so $D_2^* K_{\mathsf{open}}^{-1} = D_2^* (D_1^{*\eta})^{-1} \neq (g_1 g_2^{-y_i} g_3^{-z_i})^{\frac{1}{\theta+x_i}} = A_i$. Let $D_1 = u^{\alpha^*}$ for some $\alpha^* \in \mathbb{Z}_p^*$. By the proof of Lemma 2, we have $A_i = D_2^* (w^{\alpha^*})^{-1}$. Note that $w^{\alpha^*} = u^{\eta\alpha^*} = D_1^{*\eta}$. Hence we have a contradiction because $A_i = D_2^* (w^{\alpha^*})^{-1} = D_2^* (D_1^{*\eta})^{-1} \neq A_i$. □

Theorem 6: PS-OL is non-frameable under the DL assumption.

Proof: We will construct an efficient algorithm $\mathcal{S}$ which solves the DL problem using $\mathcal{F}$ who breaks the non-frameability of PS-OL as a sub-algorithm. Let $(\mathbb{G}_1, u, B = u^z)$ be a random instance of the DL problem in $\mathbb{G}_1$. The goal of $\mathcal{S}$ is to find $z = \log_u B$.

First, $\mathcal{S}$ picks $\theta, \eta, \xi \xleftarrow{R} \mathbb{Z}_p^*$ and sets up all the group public parameters according to the procedure of SetUp. From the construction, $\mathcal{S}$ is aware of all master secret keys, $mik = \theta, mok = (\eta, \xi)$, and $mlk = h_1^{\xi}$. Let $gpk_0 = (\mathbf{e}, g, h_1, h_\theta, \mathcal{H}, g_1, g_2, u, w, d)$ be the initial group public key. The gpk, mik, mok, and mlk, and also $\{(A_i, x_i, y_i) | i = 1, \ldots, q\}$ are known to the adversary.

Let $q = q_1 + q_2$ be the number of all user keys that are generated during the attack, where q_1 is the number of user signature keys known to the adversary via corruption or user join queries, and q_2 is the number of all (honest) user signature keys that are not exposed to the adversary. $\mathcal{S}$ randomly picks $\zeta \in \{1, \ldots, q\}$ such that the adversary will finally output a forged signature of a user with index ζ. $\mathcal{S}$ computes $\mathbf{upk}[\zeta] = B^\eta = w^z$. Let $\mathbf{USK} = \{(A_i, x_i, y_i, z_i) | i = 1, \ldots, q\}$ denote the ordered set of all user signature keys that are generated and registered during the simulation. Note that $z_\zeta = z$ is unknown to $\mathcal{S}$ during the attack. Without loss of generality, assume that $\{z_{i_1}, \ldots, z_{i_{q_1}}\}$ is known 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 $\{0,1\}^\lambda$. Assume that hash queries are never repeated.
- USK: On query (i), if $i \neq \zeta$, return $\mathbf{usk}[\mathbf{i}] = (\mathbf{A_i}, \mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i})$ and $\mathbf{sk}[i] = z_i$ in **REG**, and otherwise, i.e., if $i = \zeta$, abort.
- GSig: On query (i, m), if $i \neq \zeta$, then generate σ using the secret signing key $\mathbf{usk}[i]$, and return it. Otherwise, i.e., if $i = \zeta$, generate σ using the simulation technique of Lemma 4, and return it.
- WReg and CrptU are trivially treated. In addition, Revoc queries can be obviously treated because $(A_\zeta, x_\zeta, y_\zeta, z_\zeta)$ would be defined as an honest user's key while not being revoked.

Assume that $\mathcal{F}$ finally outputs (M^*, σ^*, i^*) that satisfies the forgery conditions of non-frameability, (1) $1 = \mathsf{GVfy}(M^*, \sigma^*)$, (2) $i^* \in \mathbf{HU}$, $\mathbf{usk}[i^*] \neq \varepsilon$, (3) $(i^*, \tau^*) \leftarrow \mathsf{Open}(mok, \mathbf{REG}, M^*, \sigma^*)$, $1 = \mathsf{Judge}(\mathbf{upk}[i^*], M^*, \sigma^*, \tau^*)$, and (4) Any of $\mathtt{USK}(i)$ and $\mathtt{GSig}(i^*, m^*)$ has not been queried. Let $\sigma^* = (D_1^*, D_2^*, D_3^*, R_1^*, R_2^*, R_3^*, c^*, s_\alpha^*, s_x^*, s_\gamma^*, s_y^*)$. Using a similar argument for the forking lemma technique of [2] and [25] we can obtain another signature $\hat{\sigma}^* = (D_1^*, D_3^*, D_3^*, R_1^*, R_2^*, R_3^*, \hat{c}^*, \hat{s}_\alpha^*, \hat{s}_x^*, \hat{s}_\gamma^*, \hat{s}_y^*)$, where $c^* \neq \hat{c}^* \bmod p$. We can compute $\gamma^* = (c^* - \hat{c}^*)^{-1}(s_\gamma^* - \hat{s}_\gamma^*) \bmod p$, $\alpha^* = (c^* - \hat{c}^*)^{-1}(s_\alpha^* - \hat{s}_\alpha^*) \bmod p$, and $x^* = (c^* - \hat{c}^*)^{-1}(s_x^* - \hat{s}_x^*) \bmod p$, and $z^* = x^*\alpha^* - \gamma^* \bmod p$. Note that the above simulation is perfect unless an abortion occurs. If ζ is correctly guessed then there is no abortion. Hence, at least with probability $1/q$, we succeed in solving the given DL problem. □

VI. 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 three best-known privacy-protecting signature schemes such as [3]

TABLE I
COMPARISON OF SIGNATURE LENGTH

	BBS [3]	DP [25]	HLCCN [35]	Ours
Sig. Len.	$3\ell_{\mathbb{G}_1} + 6\ell_p$	$4\ell_{\mathbb{G}_1} + 5\ell_p$	$4\ell_{\mathbb{G}_1} + 8\ell_p$	$3\ell_{\mathbb{G}_1} + 5\ell_p$

TABLE II
COMPARISON OF COMPUTATION COSTS

	BBS [3]	DP [25]	HLCCN [35]	Ours
GSig	$5t_{1,1}$+$2t_{2,1}$ +$t_{3,T}$	$4t_{1,1}$+$t_{2,1}$ +$t_{3,T}$	$4t_{1,1}$+$3t_{2,1}$ +$2t_{3,1}$+$t_{7,T}$	$3t_{1,1}$+$2t_{2,1}$ +$t_{4,T}$
GVfy	$4t_{2,1}$+$t_{2,2}$ +$t_{3,T}$+t_P	$2t_{2,1}$+$t_{3,1}$+ $t_{2,2}$+$t_{3,T}$+t_P	$4t_{2,1}$+$t_{4,1}$+ $t_{2,2}$+$t_{7,T}$+t_P	$t_{2,1}$+$t_{3,1}$+ $t_{2,2}$+$t_{4,T}$+t_P
Open	$t_{2,1}$	$t_{1,1}$	$t_{1,1}$	$t_{1,1}$
*Open$^{\mathsf{R}}$	N/A	N/A	$2t_{1,1}$+$4t_{2,1}$	$5t_{1,1}$
Judge	N/A	N/A	$2t_{2,1}$+$t_{3,1}$+$2t_P$	$2t_{2,1}$+$2t_P$
Link	N/A	N/A	$2t_P$	$2t_P$

* Open$^{\mathsf{R}}$ denotes the opening algorithm when key update by revocation is considered. N/A means 'non-available'.

TABLE III
THE D174-163 CURVE WITH THE EMBEDDING DEGREE $k = 6$

q	5cae4587bb49e6a9de84b7524b73dcd45e278fa8001
n	5cae4587bb49e6a9de8532e4778e7d3884047be9604
r	68100635978611268a4080e591fc762a25dce0393
a	-3
b	3469278224570b7470f0c0d017fa1de5e16599c5049
nk	9392bba4a8b57ad531cbcfdd6ebc5744fb08aa77c81c2af2ea1 4dde2a3d202559b7f40b6563c885f0048350baeeb3e911a0ef0f 34f89f8f0db6525393d78aff22631d9312df837e8d42f5990d4 67d99b9f5f90c3207c7830b6d4fdc913c98a05644a19969a85b cc627d221155d1fadabdbb7920a54ff8402ff7850be1b710400
hk	37f7f251e3314ff4223a881674a92bbcb3648ff7e9a77a6af1f69 6c6f9d004485f738eee279a0cf5bbdb5c3255f3f7bd9a6620d54 7417b8f16a6f3787fe0e95fbdda8d1f858ec443a840490aba125 aa08b717fc14809600
coef0	17004054aa2b0ada96696fbd9fbbffc59bfe4b02189
coef1	3c3794474d5ac33f9dd2b557a678a0bb09d735e2354
coef2	21fc3dc490253b90b42fbb1cd519a5cb127ecc07088
nqr	4041cc3f31f249792155ad90f40635ad9c900a54609

referred to as BBS, [25] referred to as DP, [35] referred to as HLCCN, and ours.[4]

A. 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. Table I summarizes the comparison result among BBS, DP, HLCCN, and ours in terms of signature length. Let $\mathbb{G}_1$ be a bilinear group of prime order p on an MNT curve where ℓ_p and $\ell_{\mathbb{G}_1}$ are almost the same. Our signature length is the shortest one. When $\ell_p = 170$ and $\ell_{\mathbb{G}_1} = 171$ (which are equivalent in strength to 80-bit symmetric keys) are considered, our signature length is only about 1363 bits or 171 bytes.

B. Computation Cost

Table II summarizes comparison results of (theoretical) computation costs for signing, verifying, opening, judging, and linking, concretely. In the table, we use the following notations: $t_{x,\omega}$ denotes simultaneous multi-exponentiation using x elements of group $\mathbb{G}_{\omega\in\{1,2,T\}}$. t_P denotes a pairing computation. In our scheme, all the pairings $\mathbf{e}(\hat{w}, h_\theta)$, $\mathbf{e}(\hat{w}, h_1)$, $\mathbf{e}(\hat{g}_1, h_1)$, and $\mathbf{e}(\hat{g}_1, h_1)$ can be precomputed and included in the group public key or cached by users.

- Signing and Verifying: A signer can cache $\mathbf{e}(\hat{A}, h_1)$ instead of evaluating a pairing for each signature. Since $\mathbf{e}(D_2, h_1)$ can be computed by $\mathbf{e}(\hat{A}, h_1)\mathbf{e}(\hat{w}, h_1)^{\alpha}$, signing requires no pairing computation, and three exponentiations in $\mathbb{G}_1$, two multi-exponentiations with two bases in $\mathbb{G}_1$, and one multi-exponentiation with four bases in $\mathbb{G}_T$. Since all bases for multi-exponentiation in signing are fixed, further speedup by precomputation is possible. A verifier can derive R_2 by merging $\mathbf{e}(D_2, h_1)^{s_x}$ and $\mathbf{e}(D_2, h_\theta)^{c}$ into $\mathbf{e}(D_2, h_1^{s_x} h_\theta^{c})$ and evaluating one pairing. Thus, verifying requires one multi-exponentiations with two bases in $\mathbb{G}_1$, one multi-exponentiations with three bases in $\mathbb{G}_1$, one multi-exponentiation with two bases in $\mathbb{G}_2$, one multi-exponentiation with four bases in $\mathbb{G}_T$, and one pairing computation.
- Opening: Assume that a key-update by revocation is considered.[5] In our scheme, the value g^y does not change regardless of revocation and join. To open a signature, our scheme first extract some linking index, g^y from a signature and then directly decide the corresponding signer. Hence our approach minimizes the opening computation.
- Linking: As the scheme of [35], our linking algorithm computes two pairing computations for two given signatures $\sigma = (\lambda, D_1, D_2, D_3, \ldots)$ and $\sigma' = (\lambda', D_1', D_2', D_3', \ldots)$ by testing $\mathbf{e}(D_3/D_3', h_1) = \mathbf{e}(D_1/D_1', U)$.
- Key-Updating: Whenever a revocation occurs, all but revoked users must update their secret signature keys. Let r be the number of revoked users. This updating requires $k \cdot r$ multi-exponentiations (in $\mathbb{G}_1$) for a fixed constant k.

C. Experimental Results

The test for these results was performed on an Intel Core2 Quad Q9550 model CPU clocked at 2.83GHz. This test makes use of four *d-type* curves from the PBC library [46] running on top of Gnu GMP [32] on Linux 2.6.25-14 (Fedora Core 9). Due to the space limitation, we only show the parameters for D174-163 Curve in Table III through we have also used the D224-224 curve.[6] Every test result is the average of 1,000 tests.

Table IV shows the running time of GSign, GVrfy, Open, Judge, and Link of our scheme. Open$^{0\mathsf{R}}$ and Open$^{10^4\mathsf{R}}$ represent the opening procedure without generating a

[4]To the best of our knowledge, we believe that BBS and DP are well-known group signatures to provide a revocation method based on key-update and proven to be secure in the random oracle model as ours. HLCCN offers the same properties as ours.

[5]When a key-update by revocation is considered, BBS [3] and DP [25] provides explicit opening algorithms.

[6]d174-163 and d224-224 provide 978-bits and 1344-bits of security which are comparable to that provided by 978-bit and 1344-bit RSA cryptosystems, respectively. Currently, 1024-bits of security may be considered adequate.

TABLE IV
EXPERIMENTAL RESULT I

	d174-163	d224-224
GSig/GVfy	23.76 / 47.68	38.11 / 80.36
$\mathsf{Open}^{0R}/\mathsf{Open}^{10^4R}$	1.54 / 1.54	2.76 / 2.82
Open/ Judge	8.34 / 30.32	15.15 / 52.70
Link	26.08	46.54

(time:msec)

TABLE V
EXPERIMENTAL RESULT II (ON iPHONE4)

	d174-163	d224-224
GSig	296.33	391.84
GVfy	359.86	542.99

(time:msec)

convincing proof[7] after key-updates with 0 and 10^4 revoked users, respectively.

Table V shows the test results of our scheme which are performed on iPhone4 platform. This platform is known to use A4 processor clocked at maximum 1Ghz. In Table V, the running time of GSig on iPhone4 is $8.2 \sim 12.5$ times slower than that on Intel Core2 Quad and GVfy on iPhone4 is $6.6 \sim 7.5$ times slower.

VII. VARIATIONS

In this section we show that our scheme can be modified to gain various properties.

First, the proposed scheme provides global linkability via a linking key in the sense that signatures generated by a signer, regardless of the change of membership, can be linked by a linking key. In certain situations this property will not be desirable for privacy. To remove this property, we can easily modify our scheme so that a signer generates y whenever he/she re-joins a group.

Second, in some situations, a signer may want to give its linking information flexibly according to some policies or benefits. To enable this *signer-controlled linkability*, we can modify the proposed scheme as follows. To generate a signature, a signer computes $\mathsf{g} = H_0(\omega)$ and then $D_3 = \mathsf{g}^y \hat{d}^\alpha$, instead of $g^y \hat{d}^\alpha$, where $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$ is a map-to-point hash function[8] and ω is a link base. Signatures generated with ω as a link base can only be linked by a linking key. Note that, even if the same ω is used, signatures generated by different signers cannot be linked because $\mathsf{g}^y \neq \mathsf{g}^{y'}$. In addition, the use of the same ω does not guarantee that signatures are linked (without a linking key).

Finally, the revocation of the proposed scheme relies on a key-update based mechanism. That is, whenever a revocation occurs, the list of (partial information of) revoked keys are published so that each signature key and the group public key are updated. However we can consider other approaches to support various revocations [4]. For example, in the above modified scheme with the signer-controlled linkability, we can use a link information, g^y to revoke a signature associated with it. To revoke a signature, a verifier checks if $(\hat{u}, \hat{d}, D_1 = \hat{u}^\alpha, \hat{d}^\alpha = D_3(\mathsf{g}^y)^{-1})$ is a correct DDH tuple.

[7]Unlike the proposed scheme, this opening procedure assumes only *trusted opener*.

[8]It maps arbitrary messages onto points of pairing-friendly elliptic curves [46].

VIII. CONCLUSION

We have constructed a dynamic PS-OL scheme which yields a short signature. We also formally proved that the constructed scheme achieves anonymity, traceability, non-frameability, and also three security requirements for (controllable) linkability. We showed that our scheme outperforms the best-known anonymous signature schemes. Our scheme will be very versatile and useful in many privacy-enhancing applications with limited resources.

REFERENCES

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 1880. Berlin, Germany: Springer-Verlag, 2000, pp. 255–270.

[2] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," *J. Cryptol.*, vol. 21, no. 2, pp. 149–177, 2008.

[3] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 3152. Berlin, Germany: Springer-Verlag, 2004, pp. 41–55.

[4] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proc. ACM CCS*, 2004, pp. 132–145.

[5] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup, "Anonymous credentials on a standard java card," in *Proc. ACM CCS*, 2009, pp. 600–610.

[6] E. Brickell, L. Chen, and J. Li, "Simplified security notions of direct anonymous attestation and a concrete scheme from pairings," *Int. J. Inf. Security*, vol. 8, no. 5, pp. 315–330, 2009.

[7] P. Bichsel, J. Camenisch, G. Neven, B. Warinschi, and N. P. Smart, "Get shorty via group signatures without encryption," in *Security and Cryptography for Networks*. Berlin, Germany: Springer-Verlag, 2010, pp. 381–398.

[8] X. Boyen and C. Delerablee, "Expressive subgroup signatures," in *Security and Cryptography for Networks* (Lecture Notes in Computer Science), vol. 5229. Berlin, Germany: Springer-Verlag, 2008, pp. 185–200.

[9] E. Brickell and J. Li, "A pairing-based DAA scheme further reducing TPM resources," in *Proc. 3rd TRUST*, 2010, pp. 181–195.

[10] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 2656. Berlin, Germany: Springer-Verlag, 2003, pp. 614–629.

[11] J.-M. Bohli and A. Pashalidis, "Relations among privacy notions," *ACM Trans. Inf. Syst. Security*, vol. 14, no. 1, 2011, Art. ID 4.

[12] M. Bellare, H. Shi, and C. Zang, "Foundations of group signatures: The case of dynamic groups," in *Topics in Cryptology* (Lecture Notes in Computer Science), vol. 3376. Berlin, Germany: Springer-Verlag, 2004, pp. 136–153.

[13] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proc. ACM CCS*, 2004, pp. 168–177.

[14] X. Boyen and B. Waters, "Compact group signatures without random oracles," in *Advances in Cryptology*, vol. 4004. Berlin, Germany: Springer-Verlag, 2006, pp. 427–444.

[15] X. Boyen and B. Waters, "Full-domain subgroup hiding and constant-size group signatures," in *Public Key Cryptography*, vol. 4450. Berlin, Germany: Springer-Verlag, 2007, pp. 1–15.

[16] J. Camenisch and T. Groß, "Efficient attributes for anonymous credentials," in *Proc. ACM CCS*, 2004, pp. 345–356.

[17] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact E-cash," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 3494. Berlin, Germany: Springer-Verlag, 2005, pp. 302–321.

[18] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 2442. Berlin, Germany: Springer-Verlag, 2002, pp. 61–76.

[19] J. Camenisch, D. Sommer, and R. Zimmermann, "A general certification framework with applications to privacy-enhancing certificate infrastructures," in *Security and Privacy in Dynamic Environments*. Berlin, Germany: Springer-Verlag, 2006.
[20] S. Canard, B. Schoenmakers, M. Stam, and J. Traoré, "List signature schemes," *Discrete Appl. Math.*, vol. 154, no. 2, pp. 189–201, Feb. 2006.
[21] D. Chaum and E. van Heyst, "Group Signatures," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 547. Berlin, Germany: Springer-Verlag, 1991, pp. 257–265.
[22] L. Chen and J. Li, "Flexible and scalable digital signatures in TPM 2.0," in *Proc. ACM CCS*, 2013, pp. 37–48.
[23] S. G. Choi, K. Park, and M. Yung, "Short traceable signatures based on bilinear pairings," in *Advances in Information and Computer Security* (Lecture Notes in Computer Science), vol. 4266. Berlin, Germany: Springer-Verlag, 2006, pp. 88–103.
[24] S. S. M. Chow, W. Susilo, and T. H. Yuen, "Escrowed linkability of ring signatures and its applications," in *Progress in Cryptology* (Lecture Notes in Computer Science), vol. 4341. Berlin, Germany: Springer-Verlag, 2006, pp. 175–192.
[25] C. Delerablée and D. Pointcheval, "Dynamic fully anonymous short group signatures," in *Progress in Cryptology* (Lecture Notes in Computer Science), vol. 4341. Berlin, Germany: Springer-Verlag, 2006, pp. 193–210.
[26] M. Franz, B. Meyer, and A. Pashalidis, "Attacking unlinkability: The importance of context," in *Privacy Enhancing Technologies* (Lecture Notes in Computer Science), vol. 4776. Berlin, Germany: Springer-Verlag, 2007, pp. 1–16.
[27] P.-A. Fouque and D. Pointcheval, "Threshold cryptosystems secure against chosen-ciphertext attacks," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 2248. Berlin, Germany: Springer-Verlag, 2001, pp. 351–368.
[28] I. F. Blake, G. Seroussi, and N. P. Smart, *Advances in Elliptic Curve Cryptography*, vol. 317. Cambridge, U.K.: Cambridge Univ. Press, 2005, ch. 9, pp. 183–213.
[29] S. D. Gordon, J. Katz, and V. Vaikuntanathan, "A group signature scheme from lattice assumptions," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 6477. Berlin, Germany: Springer-Verlag, 2010, pp. 395–412.
[30] J. Groth, "Fully anonymous group signatures without random oracles," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 4833. Berlin, Germany: Springer-Verlag, 2007, pp. 164–180.
[31] J. Guo, J. P. Baugh, and S. Wang, "A group signature based secure and privacy-preserving vehicular communication framework," in *Proc. Mobile Netw. Veh. Environ.*, May 2007, pp. 103–108.
[32] *The GNU Multiple Precision Arithmetic Library*. [Online]. Available: http://gmplib.org, accessed Jun. 2014.
[33] D. Hughes and V. Shmatikov, "Information hiding, anonymity and privacy: A modular approach," *J. Comput. Secur.*, vol. 12, no. 1, pp. 3–36, 2004.
[34] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang, "Short group signatures with controllable linkability," in *Proc. LightSec*, Mar. 2011, pp. 44–52.
[35] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang, "Group signatures with controllable linkability for dynamic membership," *Inf. Sci.*, vol. 222, pp. 761–778, Feb. 2013.
[36] E. Hufschmitt and J. Traoré, "Fair blind signatures revisited," in *Pairing-Based Cryptography* (Lecture Notes in Computer Science), vol. 4575. Berlin, Germany: Springer-Verlag, 2007, pp. 268–292.
[37] IETF. *RFC5636 Traceable Anonymous Certificate*. [Online]. Available: https://tools.ietf.org/html/rfc5636, accessed Aug. 2009
[38] A. Kiayias and M. Yung, "Group signatures with efficient concurrent join," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 3494. Berlin, Germany: Springer-Verlag, 2005, pp. 198–214.
[39] A. Kiayias, Y. Tsiounis, and M. Yung, "Traceable signatures," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 3027. Berlin, Germany: Springer-Verlag, 2004, pp. 571–589.
[40] A. Langlois, S. Ling, K. Nguyen, and H. Wang, "Lattice-based group signature scheme with verifier-local revocation," in *Public-Key Cryptography* (Lecture Notes in Computer Science), vol. 4833. Berlin, Germany: Springer-Verlag, 2007, pp. 164–180.
[41] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," in *Information Security and Privacy* (Lecture Notes in Computer Science), vol. 3108. Berlin, Germany: Springer-Verlag, 2004, pp. 325–335.
[42] B. Libert and M. Yung, "Efficient traceable signatures in the standard model," *Theoretical Comput. Sci.*, vol. 412, nos. 12–14, pp. 1220–1242, Mar. 2011.
[43] M. Manulis, "Democratic group signatures: On an example of joint ventures," in *Proc. ASIACCS*, 2006, pp. 365–201.
[44] M. Manulis, A.-R. Sadeghi, and J. Schwenk, "Linkable democratic group signatures," in *Information Security Practice and Experience* (Lecture Notes in Computer Science), vol. 3903. Berlin, Germany: Springer-Verlag, 2006, pp. 187–201.
[45] K. G. Paterson, "Cryptography from pairings," in *Advances in Elliptic Curve Cryptography*, vol. 317. Cambridge, U.K.: Cambridge Univ. Press, 2005, ch. 10, pp. 215–251.
[46] *PBC Library*. [Online]. Available: http://crypto.stanford.edu/pbc, accessed Jun. 2014.
[47] C.-M. Park and H.-S. Lee, "Pairing-friendly curves with minimal security loss by Cheon's algorithm," *ETRI J.*, vol. 33, no. 4, pp. 656–659, 2011.
[48] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–369, 2000.
[49] Y. Sun, R. Lu, X. Lin, X. Shen, and J. Su, "An efficient pseudonymous authentication scheme with strong privacy preservation for vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 59, no. 7, pp. 3589–3603, Sep. 2010.
[50] J. Sun, C. Zhang, and Y. Fang, "An ID-based framework achieving privacy and non-repudiation in vehicular ad hoc networks," in *Proc. IEEE Military Commun. Conf.*, Oct. 2007, pp. 1–7.
[51] P. P. Tsang, V. K. Wei, T. K. Chan, M. H. Au, J. K. Liu, and D. S. Wong, "Separable linkable threshold ring signatures," in *Progress in Cryptology* (Lecture Notes in Computer Science), vol. 3348. Berlin, Germany: Springer-Verlag, 2004, pp. 337–376.
[52] C. Zhang, X. Lin, R. Lu, P.-H. Ho, and X. Shen, "An efficient message authentication scheme for vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 57, no. 6, pp. 3357–3368, Nov. 2008.
[53] *TCG TPM V1.2 Specification*. [Online]. Available: http://www.trustedcomputinggroup.org/resources/tpm_main_specification, accessed Jun. 2014.
[54] G. Tsudik and S. Xu, "Accumulating composites and improved group signing," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 2894. Berlin, Germany: Springer-Verlag, 2003, pp. 269–286.
[55] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery: An overview," in *Advances in Knowledge Discovery and Data Mining*. Menlo Park, CA, USA: AAAI, 1996.
[56] V. K. Wei, "Tracing-by-linking group signatures," in *Information Security* (Lecture Notes in Computer Science), vol. 3650. Berlin, Germany: Springer-Verlag, 2005, pp. 149–163.
[57] Q. Wu, J. Domingo-Ferrer, and U. Gonzalez-Nicolas, "Balanced trustworthiness, safety, and privacy in vehicle-to-vehicle communications," *IEEE Trans. Veh. Technol.*, vol. 59, no. 2, pp. 559–573, Feb. 2010.
[58] Z. Wan, K. Ren, B. Zhu, B. Preneel, and M. Gu, "Anonymous user communication for privacy protection in wireless metropolitan mesh networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 2, pp. 519–532, Feb. 2010.

Jung Yeon Hwang received the B.S. degree in mathematics and the M.S. and Ph.D. degrees in information security from Korea University, Seoul, Korea, in 1999, 2003, and 2006, respectively. He has been a Post-Doctoral Researcher with Korea University from 2006 to 2009. Since 2009, he has been a Senior Member of the Engineering Staff with the Electronics and Telecommunications Research Institute, Daejeon, Korea. His research interests include theory and applications of cryptography. His main contribution is in the area of broadcast encryption and privacy-enhancing authentication.

Liqun Chen is currently a Researcher with Hewlett-Packard Labs (HP Labs), Bristol, U.K. She has developed several cryptographic schemes adopted by the International Standards and some of them have been implemented in Trusted Platform Modules. She has an extensive publication record and holds a large number of granted patents in cryptography and information security. She received the Ph.D. degree in engineering from Southeast University, Lakeland, FL, USA. Prior to joining HP Labs, she was with Southeast University, the University of Oxford and Royal Holloway, Oxford, U.K., and the University of London, London, U.K.

Hyun Sook Cho is currently the Vice President and a Principal Researcher with the Electronics and Telecommunication Research Institute, Daejeon, Korea. She has worked for the development of information security technologies, in particular, key management mechanism and cryptographic algorithm. She has authored over 100 paper journal for cyber security and safety. She is a coauthor of *Next-Generation Network Security Technology* and an author of a special issue paper, "Cyber cold war: Present situation and future task." She received the Ph.D. degree in computer science from Chungbuk National University, Cheongju, Korea.

DaeHun Nyang received the B.Eng. degree in electronics engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1994, and the M.S. and Ph.D. degrees in computer science from Yonsei University, Seoul, Korea, in 1996 and 2000, respectively. He was a Senior Member of Engineering Staff with the Electronics and Telecommunications Research Institute, Daejeon, from 2000 to 2003. Since 2003, he has been a Full Professor with the Department of Computer and Information Engineering, Inha University, Incheon, Korea, where he is currently the Founding Director of the Information Security Research Laboratory. He is a member of the Board of Directors and a member on the Editorial Board of the Korean Institute of Information Security and Cryptology, Seoul, Korea. His research interests include cryptography and network security, privacy, usable security, biometrics, and their applications to authentication and public key cryptography.