Limited-linkable Group Signatures with Distributed-Trust Traceability[⋆]

Clemens Hlauschek[1], John Black[3], Giovanni Vigna[2], and Christopher Kruegel[2]

[1]Vienna University of Technology, Austria
[2] University of California, Santa Barbara
[3] University of Colorado, Boulder
haku@iseclab.org,jrblack@cs.colorado.edu,{vigna,chris}@cs.ucsb.edu

Abstract. Group signatures allow a group member to sign anonymously on behalf of a group. In the dynamic case, a group manager can add and revoke group members. An opening manager can revoke the anonymity of a signature and trace it back to the original group member. We introduce *limited-linkable* group signatures: two signatures on identical messages by the same group member can be efficiently linked. Furthermore, we show how to distribute the opening manager, so that *no trusted third party* is required to guarantee anonymity. Our system generates short and efficient signatures, and is provably secure in the random oracle model.

Keywords: Group Signatures, Limited-linkability, Anonymity, Distributed-trust

1 Introduction

Group signatures have been introduced by Chaum and van Heyst [8] and allow a group member to sign anonymously on behalf of a group. Bellare, Micciancio, and Warinshi [1] have provided formal definitions for the static case, and Bellare, Shi, and Zang [3] extended the definitions to the dynamic case, where a group manager $\mathcal{GM}$ can add and revoke group members. Boneh, Boyen, and Shacham (BBS) [6] provided an efficient, short construction for the static case by using bilinear pairings, although with relaxed anonymity requirement. Based on BBS, Delerablée and Pointcheval [9] have constructed an even shorter scheme, which is secure according to the formal definitions for the dynamic case.

Based on the definition of dynamic group signatures [3], we introduce and formalize the notion of *limited-linkable group signatures* ($\mathcal{LGS}$): we add a *limited-linkability* property, which ensures that two signatures on identical messages by the same group member can be efficiently linked by anyone. Signatures on different messages by the same group member remain unlinkable. This property is useful for example in anonymous voting systems, where a user has to collect

[⋆] This is a revision of an earlier version of the paper titled "Limited-linkable group signatures based on bilinear pairings."

signatures from a number of distinct group members: as long as a group member does not sign identical messages twice, all his signatures remain unlinkable. A user presenting a number of signatures on a message can easily prove to anyone that all signatures on the message are indeed from distinct group members.

In dynamic group signatures [3], an opening manager $\mathcal{OM}$ can revoke the anonymity of a signature and trace it back to the original group member. Furthermore, to guarantee anonymity, a *trusted third party* has to be involved during the setup phase. We show how to eliminate the dependence on the trusted third party for the anonymity property by devising a *distributed-trust* based scheme: we distribute the opening manger $\mathcal{OM}$ to all parties and compute setup parameters with a distributed key generation (DKG) [12] protocol. That way, all parties involved have to cooperate to trace a signature and de-anonymize the signer.

We base our construction on the efficient systems of Boneh, Boyen, and Shacham [6] and Delerablée and Pointcheval [9]. We achieve the *limited-linkability* property by integrating a tag, constructed from the verifiable random function of Dodis and Yampolskiy [10], into the zero-knowledge proof of knowledge underlying the scheme. We present security proofs of our system in the random oracle model.

Related Work. Ring signatures, first introduced by Rivest [17], are similar to group signatures, but lack a group manager who is responsible for adding and revoking members. Ring signatures with similar linkability properties as our scheme exist [13, 20, 21].

2 Overview

Our *Limited-linkable dynamic group signature* ($\mathcal{LGS}$) scheme is based on the work of Boneh, Boyen, and Shacham (BBS) [6], a very efficient scheme generating short, constant-sized signatures using bilinear pairings, and its improvement by Delerablée and Pointcheval (DP) [9], which is secure according to the formal definitions of dynamic group signatures ($\mathcal{GS}$), as given by Bellare, Shi and Zhang [3]. We introduce and formalize the notion of *limited-linkable group signatures* by extending and modifying the formal definitions of $\mathcal{GS}$.

BBS and DP signatures are basically *honest-verifier zero-knowledge proofs of knowledge* turned into *signatures of knowledge* (SK) by the Fiat-Shamir heuristic [11]. They are provably secure in the random oracle model [2].

Limited-linkability. Limited-linkability guarantees that two signatures on an identical message by the same group member can be efficiently linked by anyone seeing those two signatures. To achieve the *limited linkability* property, we extend the underlying SK by integrating a tag that is verifiably and deterministically computed by a pseudorandom function, due to DY [10], seeded by the message m and the group member's private key y. Obviously, this tag is the same for every equal pair (m, y). Otherwise, it is indistinguishable from random.

Distributed Key Generation. A dynamic group signature $\mathcal{GS}$ [3] scheme assumes a trusted third party that runs the *setup* algorithm GKg, which generates the group manager $\mathcal{GM}$'s secret key ik, the opening manager $\mathcal{OM}$'s secret key ok, and the group public key gpk. We get rid of this trusted third party in the anonymity definition. In our scheme, all parties need to cooperate to trace a signature and de-anonymize the signer of a message. No single entity can silently put any possible trapdoor information into the group public key gpk during the setup phase.

We distribute the opening manger $\mathcal{OM}$ to all parties involved and compute parameters with a distributed key generation (DKG) [12] protocol during the setup phase of our scheme. This disables tracing signers and ensures anonymity as long as a single party refuses to cooperate to open a signature. By formally keeping the *traceability* property, we retain all properties (*unforgeability* and *coalition resistance*) that follow from the *traceability* and *non-frameability* properties [3].

We instantiate the DKG protocol of Gennaro, Jarecki, Krawczyk and Rabin (GJKR) [12]. GJKR is an improvement over the well known Pedersen's DKG [16] protocol. The GJKR protocol allows us to securely compute a public key $y \leftarrow g^x \pmod{p}$ of a discrete-log based cryptosystem by a set of n peers, so that no information on x can be learned by the adversary (except for what is implied by the value $y = g^x \pmod{p}$) as long as no more than t peers collaborate. The value y obtained from the distributed computation has the property that it is taken from a uniform random distribution. The computation is secure and robust as long as an adversary controls at most $t < n/2$ of the n parties involved in the computation.

3 Preliminaries

Bilinear Groups. We recall the algebraic setting of BBS [6] and DP [9]. We follow the notation of BBS: G_1, G_2, G_T are cyclic groups of prime order p. g_1 (g_2) is a generator of G_1 (G_2). We assume an efficiently computable bilinear map $e : G_1 \times G_2 \mapsto G_T$, s.t. $\forall\ u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$: $e(u^a, v^b) = e(u, v)^{ab}$ (*Bilinearity*), and $e(g_1, g_2) \neq 1$ (*Non-degeneracy*). We assume an easy computable, but hard to invert isomorphism ψ from G_2 onto G_1.

***q*-Strong Diffie-Hellman (SDH) Assumption.** Given a $(q+2)$-tuple $(g_1 = \psi(g_2), g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \cdots, g_2^{(\gamma^q)})$ in the bilinear group pair (G_1, G_2), it is hard to output a pair $(g_1^{1/(\gamma+x)}, x) \in (G_1, \mathbb{Z}_p)$, with a freely chosen $\gamma \in \mathbb{Z}_p$.[1]

[1] SDH was introduced and analyzed in the generic group model [19] by Boneh [5].

eXternal Diffie-Hellman (XDH) Assumption. XDH states that the Decisional Diffie-Hellman (DDH) problem is hard in G_1, implying that the isomorphism ψ is one-way.[2]

***q*-Decisional Diffie-Hellman Inversion (DBDHI) Assumption.** DBDHI asserts that it is hard to distinguish $e(g_1, g_2)^{1/x}$ from random even when $(g_2, g_2^x, g_2^{(x^2)}, \cdots, g_2^{(x^q)})$ is known.[3]

4 Algorithms

We review the algorithms and protocols of a dynamic group signature ($\mathcal{GS}$) [3] scheme: a dynamic group signature scheme is a tuple $\mathcal{GS} = (\mathsf{GKg}, \mathsf{UKg}, \mathsf{Join}, \mathsf{Iss}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge})$ of algorithms.

GKg– The *setup* algorithm GKg, on input 1^k, outputs the group manager's secret key ik, the opening manager's secret key ok, and the group public key gpk.

Join– During the Join protocol, $\mathcal{GM}$ makes an entry $\mathsf{reg}[i]$ in the registration table reg, and the new group member $\mathcal{U}_i$ obtains his private *group signing key* $\mathsf{gsk}[i]$.

GSig– Group member $\mathcal{U}_i$ invokes the *signing* algorithm GSig and obtains a group signature σ on message m, using his group signing key $\mathsf{gsk}[i]$.

GVf– The *verification* algorithm GVf takes as input a message m, a candidate signature σ, and a group public key gpk, and outputs 1 if σ is valid; 0 otherwise.

$\mathsf{Open}, \mathsf{Judge}$– The *opening* algorithm Open, given m, σ, ok as input, revokes the anonymity of the signer, while Judge verifies the correctness of the results of Open.

Our Scheme. Our group signature scheme $\mathcal{LGS}$ is specified by the tuple (GKg, Join, GSig, GVf, Link). We do not implement the algorithms Open and Judge, but require them for the oracles in the security proofs. We introduce the additional efficient *linking* algorithm Link.

Link– The *linking* algorithm Link takes as input a tuple of group signatures $(\sigma_1, \sigma_2, \cdots, \sigma_n)$ and outputs 1 if it contains a pair of signatures (σ_i, σ_j) that was generated by one and the same group member $\mathcal{U}_i$ using his group signing key $\mathsf{gsk}[i]$ on the same message m.

In order to formalize the definition of the *anonymity* property of our $\mathcal{LGS}$ scheme, we split the GKg algorithm into the Setup algorithm and the GKg' protocol. The algorithm GKg is thus specified by the tuple $\mathsf{GKg} = (\mathsf{Setup}, \mathsf{GKg}')$.

[2] XDH was also previously used in the full version of Compact E-Cash [7]. Note that our scheme can be easily adapted to be secure under a weaker assumption, which would result in slightly longer signatures.

[3] DBDHI was first introduced in a slightly different, symmetric bilinear group setting (with $G_1 = G_2$) [4] and analyzed in the generic group model [19] by DY [10].

$\mathsf{Setup}-$, The algorithm Setup, on input of a security parameter 1^k, outputs params. The output params contains the general settings of our scheme: The groups G_1, G_2, G_T, the bilinear map $e : G_1 \times G_2 \mapsto G_T$, and the isomorphism $\psi : G_2 \mapsto G_1$. In practice, these parameters will be hardcoded.

$\mathsf{GKg}'-$ The protocol GKg' takes as input params. GKg' is run between all users $\mathcal{U}_i$ and the group manager $\mathcal{GM}$. On input params, the GKg' protocol outputs gpk to all parties. $\mathcal{GM}$ obtains the group manager $\mathcal{GM}$'s secret key ik. Shares of the opening manager $\mathcal{OM}$'s secret key ok are distributed to all parties.

The group public key gpk consists of the tuple $\mathsf{gpk} = (\mathsf{params}, g_1, g_2, \tilde{g}_1, \mathsf{ompk}, \mathsf{gmpk})$. The opening manager $\mathcal{OM}$'s public key ompk is associated with $\mathcal{OM}$'s secret key ok; gmpk denotes the public key associated with the group manager $\mathcal{GM}$'s secret key ik; and $g_1, \tilde{g}_1$, resp. g_2 are generators of G_1, resp. G_2.

5 Security Notions

Unless explicitly stated otherwise, the security definitions of our $\mathcal{LGS}$ scheme are identical to those of $\mathcal{GS}$ [3]. Next, we provide an informal overview of the security properties of $\mathcal{LGS}$.

Correctness. Signatures generated by honest group members are valid. The opening algorithm Open correctly identifies the signer (and Judge accepts the results of Open).

Traceability. It is impossible to produce a signature that cannot be correctly opened by $\mathcal{OM}$.

Non-frameability. No one is able to accuse someone wrongly for having signed a message.

Anonymity. The anonymity property ensures that signatures do not reveal their signer's identity. We change the anonymity definition of $\mathcal{GS}$ due to our new limited-linkability property, and because we remove the dependence on a trusted third party during the setup phase.

Limited-linkability. We introduce the *limited-linkability* property: Two signatures σ_1, σ_2, both generated with the signing key $\mathsf{gsk}[i]$ on the message m, can be efficiently linked.

5.1 Formal Security Definitions

The *correctness*, *traceability* and *non-frameability* properties are identical to $\mathcal{GS}$ [3]. Here, we define the *anonymity* and *limited-linkability* properties.

We first recall the oracles of a dynamic groups signature $\mathcal{GS}$ scheme, as defined by Bellare et al. [3], which are used in the formal security definitions to model the attackers capabilities. Each user $\mathcal{U}_i$ has an associated personal public and private key pair $(\mathsf{upk}[i], \mathsf{usk}[i])$, the table of public keys upk are made available to all users $\mathcal{U}$. We will introduce the new $\mathsf{DKG}(\cdot)$ oracle to model the distributed key generation, and modify the challenge oracle.

Oracles from $\mathcal{GS}$ [3]:

$\mathsf{AddU}(\cdot)$– Add a new user $\mathcal{U}_i$ as a member to the group by running the Join protocol between the group manager $\mathcal{GM}$ and $\mathcal{U}_i$. Add user $\mathcal{U}_i$ to the list of honest users HU.

$\mathsf{CrptU}(\cdot, \cdot)$– Corrupt a user $\mathcal{U}_i$ by setting his public key $\mathsf{upk}[i]$ to a value chosen by the adversary. Add user $\mathcal{U}_i$ to the list of corrupted users CU.

$\mathsf{SndToI}(\cdot, \cdot)$– Send a previously corrupted user $\mathcal{U}_i$ to the (honest) group manager $\mathcal{G}$ to engage in the Join protocol.

$\mathsf{SendToU}(\cdot, \cdot)$– An adversary, acting as a not necessarily honest group manager $\mathcal{GM}$, uses the *send to user* oracle to engage with an honest user $\mathcal{U}_i$ in the Join protocol.

$\mathsf{USK}(\cdot)$– Extract the group signing key $\mathsf{gsk}[i]$ and the private key $\mathsf{usk}[i]$ of user $\mathcal{U}_i$.

$\mathsf{RReg}(\cdot)$– Read the entry $\mathsf{reg}[i]$ of the registration table reg.

$\mathsf{WReg}(\cdot, \cdot)$– Write the entry $\mathsf{reg}[i]$ of the registration table reg.

$\mathsf{GSig}(\cdot, \cdot)$– Obtain a group signature σ from user $\mathcal{U}_i$ on message m.

$\mathsf{Open}(\cdot, \cdot)$– Obtain the output of the *opening* algorithm Open.

DKG Oracle. The DKG oracle $\mathsf{DKG}(\cdot)$, on input params, outputs the generators $g_1 = \psi(g_2), g_2, \tilde{g}_1$ and the group manager public key gmpk. It does so by running the GJKR protocol. Note that the GJRK protocol can be simulated under the assumption that more than $(n/2)$ of the n parties involved are not under the control of the adversary and honest. By the security of GJKR, a simulator takes $g_2 \overset{R}{\longleftarrow} G_2$, $\tilde{g}_1 \overset{R}{\longleftarrow} G_1$ and $\mathsf{ompk} \overset{R}{\longleftarrow} G_1$ uniformly at random.

```
DKG(params)
    $(G_1, G_2, G_T, e, \psi) \leftarrow \mathsf{params}$
    if $\mathrm{GPK} \neq \epsilon$ then
        return $\bot$
    $g_2 \leftarrow \mathrm{GJKR}$ ; $g_1 \leftarrow \psi(g_2)$ ; $\tilde{g}_1 \leftarrow \mathrm{GJKR}$ ; $\mathsf{ompk} \leftarrow \mathrm{GJKR}$
    $\mathrm{GPK} \leftarrow (\mathsf{params}, g_1, g_2, \tilde{g}_1, \mathsf{ompk}, \cdot)$
    return $(g_1, g_2, \tilde{g}_1, \mathsf{ompk})$
```

Challenge Oracle. We redefine the *challenge* oracle $\mathsf{Ch_b}(\cdot, \cdot, \cdot)$ of dynamic group signatures $\mathcal{GS}$ [3]. The challenge oracle of our $\mathcal{LGS}$ scheme does not give the adversary control over the message m. Instead, the oracle $\mathsf{Ch}_{b,m}(\cdot, \cdot, \cdot)$ takes a message m as input, set by the overlying experiment. Note that this means our $\mathcal{LGS}$ scheme does not guarantee anonymity if an attacker can give a group member arbitrary messages to sign. This is a trivial consequence of the *limited-linkability* property.

Additionally, the challenge oracle $\mathsf{Ch}_{b,m}(\cdot,\cdot,\cdot)$ takes as input the group public key gpk. The oracle fails if the group public key gpk is not consistent with the values in GPK, which are set during the $\mathsf{DKG}(\cdot)$ oracle.

The adversary invokes the *challenge* oracle with the identities of users $\mathcal{U}_{i_1}$, $\mathcal{U}_{i_2}$, and gpk. If the users $\mathcal{U}_{i_1}, \mathcal{U}_{i_2}$ are in the set of honest users HU, the adversary obtains as output a message m, and a group signature σ on message m signed with $\mathsf{gsk}[i_b]$, the group signing key of $\mathcal{U}_{i_b}$.

```
$\mathsf{Ch}_{b,m}(i_0, i_1, \mathsf{gpk})$
    **if** GPK $= \epsilon$ **then**
        return $\perp$
    $(\mathsf{params}', g_1', g_2', \tilde{g}_1', \mathsf{ompk}') \leftarrow \mathsf{gpk}$ ; $(\mathsf{params}, g_1, g_2, \tilde{g}_1, \mathsf{ompk}) \leftarrow \mathrm{GPK}$
    **if** $(\mathsf{params}, g_1, g_2, \tilde{g}_1, \mathsf{ompk}) \neq (\mathsf{params}', g_1', g_2', \tilde{g}_1', \mathsf{ompk}')$ **then**
        return $\perp$
    **if** $\mathcal{U}_{i_0}, \mathcal{U}_{i_1} \notin HU$ **then**
        return $\perp$
    **if** $\mathsf{gsk}[i_0] = \epsilon$ *or* $\mathsf{gsk}[i_1] = \epsilon$ **then**
        return $\perp$
    $\sigma \leftarrow \mathsf{GSig}(\mathsf{gpk}, \mathsf{gsk}[i_b], m)$
    return $\sigma, m$
```

Anonymity. An attacker A of the *anonymity* of our $\mathcal{LGS}$ group signature scheme is modeled by the following experiment. The challenge message m is chosen uniformly at random and is of length l, where l is security parameter, such that the probability that the same message gets chosen twice is negligible. $\mathcal{M}$.

```
Experiment $\mathbf{Exp}^{\mathrm{anon}-b}_{\mathcal{LGS},A}(k)$    // $b \in \{0,1\}$
    $\mathsf{params} \leftarrow \mathsf{Setup}(1^k)$ ; CU $\leftarrow \emptyset$ ; HU $\leftarrow \emptyset$ ; GPK $\leftarrow \epsilon$
    $m \overset{R}{\leftarrow} \{0,1\}^l$
    $d \leftarrow A(\mathsf{params} : \mathsf{DKG}(\cdot), \mathsf{Ch}^{\mathsf{m}}_{\mathsf{b}}(\cdot,\cdot,\cdot), \mathsf{SndToU}(\cdot,\cdot), \mathsf{GSig}', \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$
    return $d$
```

For our group signature scheme $\mathcal{LGS}$, any adversary A, and any $k \in \mathbb{N}$, the advantage of adversary A breaking the *anonymity* property is defined by the function

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{LGS},A}(k) = \Pr\left[\mathbf{Exp}^{\mathrm{anon}-1}_{\mathcal{LGS},A}(k) = 1\right] - \left[\mathbf{Exp}^{\mathrm{anon}-0}_{\mathcal{LGS},A}(k) = 0\right] \quad . \qquad (1)$$

The scheme $\mathcal{LGS}$ is *anonymous* if the function $\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{LGS},A}$ is negligible for any probabilistic polynomial time adversary A.

Limited-linkability. We define an attacker A against the *limited-linkability* property of our $\mathcal{LGS}$ scheme by the following experiment.

```
Experiment $\mathbf{Exp}^{\text{link}}_{\mathcal{LGS},A}(k)$
    $(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok}) \leftarrow \mathsf{GKg}(1^k)$ ; $\text{CU} \leftarrow \emptyset$ ; $\text{HU} \leftarrow \emptyset$
    $(m, \sigma, \sigma') \leftarrow A(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok} : \mathsf{AddU}(\cdot), \mathsf{GSig}(\cdot), \mathsf{CrptU}(\cdot), \mathsf{SndToI}(\cdot,\cdot), \mathsf{Open}(\cdot))$
    if $\mathsf{GVf}(\mathsf{gpk}, m, \sigma) = 0$ or $\mathsf{GVf}(\mathsf{gpk}, m, \sigma) = 0$ then
        return 0
    $(i, \tau) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \mathsf{reg}, m, \sigma)$
    $(j, \tau') \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \mathsf{reg}, m, \sigma')$
    if $\mathsf{Judge}(\mathsf{gpk}, i, \mathsf{upk}[i], m, \sigma, \tau = 0)$ then
        return 0
    if $\mathsf{Judge}(\mathsf{gpk}, j, \mathsf{upk}[j], m, \sigma', \tau' = 0)$ then
        return 0
    $l = \mathsf{Link}((\sigma, \sigma'))$
    if $i = j$ and $i \neq 0$ and $l = 0$ then
        return 1
    else return 0
```

For our group signature scheme $\mathcal{LGS}$, any adversary A, and any $k \in \mathbb{N}$, the advantage of adversary A breaking the *limited linkability* property is defined by the function

$$\mathbf{Adv}^{\text{link}}_{\mathcal{LGS},A}(k) = \Pr\left[\mathbf{Exp}^{\text{link}}_{\mathcal{LGS},A}(k) = 1\right] \quad . \tag{2}$$

The scheme LGS is *limited-linkable* if the function $\mathbf{Adv}^{\text{link}}_{\mathcal{LGS},A}(k)$ is negligible for any probabilistic polynomial time adversary A.

6 Construction of our $\mathcal{LGS}$ Scheme

The Join algorithm as well as the revocation of group membership in our $\mathcal{LGS}$ scheme is identical to DP [9]. As a result of the Join protocol, the new group member $\mathcal{U}_i$ obtains an extended SDH-tuple (A, x, y) with $A \in G_1, x, y \in \mathbb{Z}_p$, s.t. $A^{x+\gamma} = g_1 \cdot h^y$ holds, where γ is $\mathcal{GM}$'s private key ik, and y is private to $\mathcal{U}_i$.

The group signature σ is a non-interactive zero-knowledge proof of knowledge of the tuple (A, x, y). The proof includes a verifiable encryption of A, which can be recovered by $\mathcal{OM}$ using his private key $\mathsf{ok} = \xi$ (the corresponding public component of ok is $h = \tilde{g_1}^{\xi}$).

To achieve the *limited-linkability* property, we include a tag, computed by a verifiable random function (VRF) [14], into the signature σ. From DY [10], we get the following efficient VRF: $F_s(x) = e(g_1, g_2)^{1/(x+s)}$, where $s, x \in \mathbb{Z}_p$. Given a public key $pk = g_1^s$ and the proof $\pi_s(x) = g_1^{1/(x+s)}$, the output $y_{DY} = F_s(x)$ can be verified by checking whether both $e(g_1^x \cdot pk, \pi)$ and $y_{DY} = e(g_1, \pi)$ hold.

The setup algorithm GKg. First, a trusted third party chooses the bilinear groups G_1, G_2, G_T with a computable isomorphism $\psi : G_2 \mapsto G_1$, and a bilinear map $e : G_1 \times G_2 \mapsto G_T$. Note that this part of the setup phase will be hardcoded in the implementation. For the rest of the setup phase, we do not require a trusted third party.

Next, the GJKR protocol is run between all parties three times to obtain a generator $g_2 \overset{R}{\longleftarrow} G_2$ of G_2 and generators $\tilde{g}_1, h \overset{R}{\longleftarrow} G_1$ of G_1, uniformly at random.

The group manager $\mathcal{GM}$ takes $\gamma \overset{R}{\longleftarrow} \mathbb{Z}_p$ at random, and calculates the group manager public key $\mathsf{gmpk} = w \leftarrow g_2^{\gamma}$.

The group manager $\mathcal{GM}$ then initializes the registration table reg.

Parameters of the $\mathcal{LGS}$ scheme:
– group public key: $\mathsf{gpk} = (\mathsf{params}, g_2, g_1 \leftarrow \psi(g_2), \tilde{g}_1, \mathsf{ompk}, \mathsf{gmpk})$
with $\mathsf{params} = (G_1, G_2, G_T, e, \psi)$, $\mathsf{ompk} = h \leftarrow \tilde{g}_1^{\xi}$, $\mathsf{gmpk} = w \leftarrow g_2^{\gamma}$
– group public key: $\mathsf{gpk} = (G_1, G_2, G_T, e, \psi, g_2, g_1 \leftarrow \psi(g_2), \tilde{g}_1, h \leftarrow \tilde{g}_1^{\xi}, w \leftarrow g_2^{\gamma})$
– OM's secret key $\mathsf{ok} = \xi$
– GM's secret key $\mathsf{ik} = \gamma$

The Join protocol. The Join protocol is identical with the Join protocol of DP [9]. It is run between the group manager $\mathcal{GM}$ and a user $\mathcal{U}_i$. The user $\mathcal{U}_i$ is in possession of a personal key pair $(\mathsf{upk}[i], \mathsf{usk}[i])$. The public key $\mathsf{upk}[i]$ must be certified through a PKI.

The user $\mathcal{U}_i$ chooses $y \overset{R}{\longleftarrow} \mathbb{Z}_p$ at random, and obtains an extended SDH-tuple (A, x, y), such that $A^{x+\gamma} = g_1 \cdot h^y$.

The group manager $\mathcal{GM}$ sets the registration table entry $\mathsf{reg}[i] \leftarrow (\mathsf{upk}[i], A, x, S)$, where S is a signature of $\mathcal{A}$, signed with the users personal private key $usk[i]$.

The GSig algorithm. To generate a group signature σ on a message m, a group member $\mathcal{U}_i$ invokes the following algorithm GSig.

Algorithm GSig of the $\mathcal{LGS}$ scheme:
Input: $(A, x, y), \mathsf{gpk}$, and a hash of $m : m' = \mathcal{H}(m)$. Take $\alpha, \rho \overset{R}{\longleftarrow} \mathbb{Z}_p$ at random.
1. Compute:

$$T_1 \leftarrow \tilde{g}_1^{\alpha}, \quad T_2 \leftarrow A \cdot h^{\alpha} \tag{3}$$

$$T_3 \leftarrow y_{DY} = e(g_1, g_2)^{\frac{1}{m'+y}} \tag{4}$$

$$T_4 \leftarrow y_{DY}^{\rho} \;, \quad T_5 \leftarrow \pi^{\rho} = g_2^{\frac{\rho}{m'+y}} \tag{5}$$

$$T_6 \leftarrow g_2^{\frac{y}{\rho}} \;, \quad T_7 \leftarrow g_1^{\frac{m'}{\rho}} \tag{6}$$

2. Next, take $r_\alpha, r_\rho, r_x, r_{\delta_1}, r_{\delta_2} \overset{R}{\leftarrow} \mathbb{Z}_p$, and compute the commitments:

$$R_1 \leftarrow \tilde{g}_1^{r_\alpha} \tag{7}$$

$$R_2 \leftarrow e(T_2, g_2)^{r_x} \cdot e(h, w)^{-r_\alpha} \cdot e(h, g_2)^{-r_{\delta_1}} \tag{8}$$

$$R_3 \leftarrow T_3^{r_\rho} \quad , \quad R_4 \leftarrow T_7^{r_\rho} \tag{9}$$

$$R_5 \leftarrow e(T_2, g_2)^{r_x} \cdot e(h, T_6)^{-r_\rho} \cdot e(h, g_2)^{-r_{\delta_2}} \cdot e(h, w)^{-r_\alpha} \tag{10}$$

3. Next, compute the challenge c with the hash function $\mathcal{H} : \{0,1\}^* \mapsto \mathbb{Z}_p$:

$$c \leftarrow \mathcal{H}(m, T_1, T_2, T_3, T_4, T_5, T_6, T_7, R_1, R_2, R_3, R_4, R_5) \tag{11}$$

4. Finally, compute the response values, with $\delta_1 = x\alpha + y, \delta_2 = x\alpha$:

$$s_\alpha \leftarrow r_\alpha + c\alpha \bmod p \quad , \quad s_\rho \leftarrow r_\rho + c\rho \bmod p$$
$$s_x \leftarrow r_x + cx \bmod p \quad , \quad s_{\delta_1} \leftarrow r_{\delta_1} + c\delta_1 \bmod p \quad , \quad s_{\delta_2} \leftarrow r_{\delta_2} + c\delta_2 \bmod p \tag{12}$$

5. Output the signature σ:

$$\sigma = (T_1, T_2, T_3, T_4, T_5, T_6, T_7, c, s_\alpha, s_\rho, s_x, s_{\delta_1}, s_{\delta_2})$$

The values T_1, T_2 are the ElGamal encryption of the value A of the users extended SDH-tuple (A, x, y). Different from DP, where double ElGamal encryption is employed to achieve IND-CCA2 security by the Naor-Yung methodology [15], we use classical ElGamal encryption to save space and reduce complexity. We can do so because the adversary in our anonymity experiment has no access to the Open oracle: Since we heavily restrict disable the opening manager $\mathcal{OM}$ by distributing it to all parties, this is reasonable.

The ElGamal encryption is semantically secure under the XDH assumption (which requires that the Decisional Diffie-Hellman problem is hard in G_1) against chosen plaintext attacks (CPA). Note that we could also easily replace the ElGamal encryption with e.g., the Linear Encryption scheme, as used in BBS. The anonymity property of our scheme would then depend on the weaker Decisional Linear Diffie-Hellman assumption.

The value T_3 is the output of the verifiable random function y_{DY}. It is the tag for the *limited-linkability* property, and is used by the *Link* algorithm. The values T_4, T_5, T_6, T_7 constitute a randomized proof of the correct computation of T_3.

The GVf algorithm. Any user can run the following verification protocol GVf to check the validity of a group signature σ of a message m.

Algorithm GVf of the $\mathcal{LGS}$ scheme:
Input: gpk, σ and a hash of $m : m' = \mathcal{H}(m)$.

1. Verify that the following equations hold:

$$e(T_7 \cdot \psi(T_6), T_5) \stackrel{?}{=} e(g_1, g_2) \tag{13}$$

$$T_4 \stackrel{?}{=} e(g_1, T_5) \tag{14}$$

2. Compute:

$$\tilde{R}_1 \leftarrow \tilde{g}_1^{s_\alpha} \cdot T_1^{-c} \tag{15}$$

$$\tilde{R}_2 \leftarrow e(T_2, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha} \cdot e(h, g_2)^{-s_{\delta_1}} \cdot \left(\frac{e(g_1, g_2)}{e(T_2, w)}\right)^{-c} \tag{16}$$

$$\tilde{R}_3 \leftarrow T_3^{s_\rho} \cdot T_4^{-c} \tag{17}$$

$$\tilde{R}_4 \leftarrow T_7^{s_\rho} \cdot (g_1^{m'})^{-c} \tag{18}$$

$$\tilde{R}_5 \leftarrow e(T_2, g_2)^{s_x} \cdot e(h, T_6)^{-s_\rho} \cdot e(h, g_2)^{-s_{\delta_2}} \cdot e(h, w)^{-s_\alpha} \cdot \left(\frac{e(g_1, g_2)}{e(T_2, w)}\right)^{-c} \tag{19}$$

3. Check that these, along with the 1. round values in σ, give c:

$$c \stackrel{?}{=} \mathcal{H}(m, T_1, T_2, T_3, T_4, T_5, T_6, T_7, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3, \tilde{R}_4, \tilde{R}_5) \tag{20}$$

3. If all checks succeed, output 1; output 0 otherwise.

The Open and Judge algorithm. The values (T_1, T_2) of a group signature σ are the ElGamal encryption of A, belonging to the signer's extended SDH-tuple (A, x, y). A is encrypted with $\mathcal{OM}$'s public key $\mathsf{ompk} = h = \tilde{g}^\xi$, so the opening manager $\mathcal{OM}$ can decrypt (T_1, T_2) and obtain A. Having access to the registration table reg, he can look up the identity of the group member who generated σ.

Furthermore, $\mathcal{OM}$ can generate a proof τ of the correct decryption of (T_1, T_2): $\mathcal{OM}$ generates a zero knowledge proof of knowledge of equality of the logarithms T_1^ξ and $\tilde{g}^\xi$. Given T_1^ξ, anyone can reproduce the decryption of (T_1, T_2) and see that it is really an encryption of A. The opening manager reveals the signature S of the entry $\mathsf{reg}[i] = (\mathsf{upk}[i], A, x, S)$ in the registration table reg, which verifies that the decrypted value A indeed belongs to user $\mathcal{U}_i$.

The Link algorithm. The Link algorithm, which takes as input a tuple of signatures $(\sigma_1, \sigma_2, \cdots, \sigma_n)$, compares all pairs of signatures and outputs 1 if it finds a pair (σ_i, σ_j), such that the value T_3 in both σ_i and σ_j is equal. Otherwise Link outputs 0.

7 Security Proof

Our group signature $\mathcal{LGS}$ scheme is secure in the random oracle model. We first show that the interactive version of the signature protocol, implicitly defined by the algorithm GSig and GVf, is an *honest-verifier zero-knowledge proof of knowledge*. Note that the zero-knowledge protocol is basically an adaption of the Schnorr [18] identification protocol.

Theorem 1. *The interactive version of the signature protocol is an honest-verifier zero-knowledge proof of knowledge of an extended SDH-tuple (A, x, y) and a honest-verifier zero-knowledge proof of the correct encryption of A and correct computation of the VRF y_{DY}, seeded by the secret value y of the tuple (A, x, y) and a given message m, under the XDH assumption.*

The proof follows directly from the *completeness*, *zero-knowledge* and *soundness* properties of the protocol. We prove these properties succeedingly.

Note that an extended SDH-tuple (A, x, y) satisfying $A^{x+\gamma} = h^y + g_1$ is valid if and only if the following equation holds (apply $e(\cdot, g_2)$ on both sides to see).

$$e(A, g_2)^x \cdot e(A, w) = e(h, g_2)^y \cdot e(g_1, g_2) \ . \tag{21}$$

Equation (21), expressed differently, gives

$$\frac{e(A, g_2)^x \cdot e(A, w)}{e(h, g_2)^y} = e(g_1, g_2) \ . \tag{22}$$

Lemma 1. *The interactive protocol is complete.*

Proof. We show that a proofer $\mathcal{P}$ who is in possession of a valid extended SDH tuple (A, x, y) will always convince an honest verifier $\mathcal{V}$ that $\mathcal{P}$ knows (A, x, y), the ephemeral key α, and the randomizing factor ρ for the proof of the VRF.

First, if the tag y_{DY} of the VRF was computed according to (4), and the values T_4, T_5, T_6, T_7 according to (5), (6), then

$$e(T_7 \cdot \psi(T_6), T_5) = e(g_1^{\frac{m'}{\rho}} g_1^{\frac{y}{\rho}}, g_2^{\frac{\rho}{m'+y}}) = e(g_1^{\frac{m'+y}{\rho}}, g_2^{\frac{\rho}{m'+y}}) = e(g_1, g_2) \ ,$$

so (13) holds, and

$$T_4 = y_{DY}^{\rho} = e(g_1, g_2)^{\frac{\rho}{m'+y}} = e(g_1, g_2^{\frac{\rho}{m'+y}}) = e(g_1, T_5) \ ,$$

so (14) holds.

Next, if s_α was computed according to (12), and T_1 according to (3), then, applying (15) and (7) gives

$$\tilde{R}_1 = \tilde{g}_1^{s_\alpha} \cdot T_1^{-c} = \tilde{g}_1^{r_\alpha} \tilde{g}_1^{(c\alpha)} \cdot \tilde{g}_1^{(-c\alpha)} = \tilde{g}_1^{r_\alpha} = R_1 \ ,$$

so $\tilde{R}_1 = R_1$. For analogous reasons, expanding $\tilde{R}_3$ according to (17) gives equality to R_3, according to (9):

$$\tilde{R}_3 = T_3^{s_\rho} \cdot T_4^{-c} = T_3^{r_\rho} T_3^{c\rho} \cdot T_4^{-c} = T_3^{r_\rho} y_{DY}^{c\rho} \cdot y_{DY}^{-c\rho} = T_3^{r_\rho} = R_3 \ ,$$

so $\tilde{R}_3 = R_3$. For analogous reasons, expanding (18) gives R_4 of (9):

$$\tilde{R}_4 = T_7^{s_\rho} \cdot (g_1^{m'})^{-c} = T_7^{r_\rho} T_7^{c\rho} \cdot (g_1^{m'})^{-c} = T_7^{r_\rho} (g_1^{\frac{m'}{\rho}})^{c\rho} \cdot (g_1^{m'})^{-c} = T_7^{r_\rho} = R_4 \quad ,$$

so $\tilde{R}_4 = R_4$. It remains to show that $\tilde{R}_2 = R_2$ and $\tilde{R}_5 = R_5$. First,

$$\begin{aligned}
\tilde{R}_2 &= e(T_2, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha} \cdot e(h, g_2)^{-s_{\delta_1}} \cdot \left(\frac{e(g_1, g_2)}{e(T_2, w)} \right)^{-c} \\
&= R_2 \cdot e(T_2, g_2)^{cx} \cdot e(h, w)^{-c\alpha} \cdot e(h, g_2)^{-c\delta_1} \cdot \left(\frac{e(g_1, g_2)}{e(T_2, w)} \right)^{-c} \\
&= R_2 \cdot \left(\frac{e(Ah^\alpha, g_2)^x}{e(h, w)^\alpha \cdot e(h, g_2)^{x\alpha} \cdot e(h, g_2)^y} \cdot \right)^c \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} \\
&= R_2 \cdot \left(\frac{e(A, g_2)^x \cdot \cancel{e(h, g_2)^{x\alpha}}}{e(h, w)^\alpha \cdot \cancel{e(h, g_2)^{x\alpha}} \cdot e(h, g_2)^y} \cdot \right)^c \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} \\
&= R_2 \cdot \left(\frac{e(A, w)}{e(A, w)} \frac{e(A, g_2)^x}{e(h^\alpha, w) \cdot e(h, g_2)^y} \cdot \right)^c \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} \\
&= R_2 \cdot \left(\frac{1}{e(Ah^\alpha, w)} \frac{e(A, g_2)^x \cdot e(A, w)}{e(h, g_2)^y} \cdot \right)^c \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} \\
&\overset{(22)}{=} R_2 \cdot \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^c \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} = R_2 \quad .
\end{aligned}$$

Secondly,

$$\begin{aligned}
\tilde{R}_5 &= e(T_2, g_2)^{s_x} \cdot e(h, T_6)^{-s_\rho} \cdot e(h, g_2)^{-s_{\delta_2}} \cdot e(h, w)^{-s_\alpha} \cdot \left(\frac{e(g_1, g_2)}{e(T_2, w)} \right)^{-c} \\
&= R_5 \cdot \left(\frac{e(T_2, g_2)^x}{e(h, T_6)^\rho \cdot e(h, g_2)^{\delta_2} \cdot e(h, w)^\alpha} \right)^c \cdot \left(\frac{e(g_1, g_2)}{e(T_2, w)} \right)^{-c} \\
&= R_5 \cdot \left(\frac{e(Ah^\alpha, g_2)^x}{e(h, g_2^{\frac{y}{\cancel{\rho}}})^{\cancel{\rho}} \cdot e(h, g_2)^{x\alpha} \cdot e(h, w)^\alpha} \right)^c \cdot \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} \\
&= R_5 \cdot \left(\frac{e(A, g_2)^x \cdot \cancel{e(h, g_2)^{x\alpha}}}{e(h, g_2)^y \cdot \cancel{e(h, g_2)^{x\alpha}} \cdot e(h, w)^\alpha} \right)^c \cdot \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} \\
&= R_5 \cdot \left(\frac{e(A, w)}{e(A, w)} \frac{e(A, g_2)^x}{e(h, g_2)^y \cdot e(h^\alpha, w)} \right)^c \cdot \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} \\
&= R_5 \cdot \left(\frac{1}{e(Ah^\alpha, w)} \frac{e(A, g_2)^x \cdot e(A, w)}{e(h, g_2)^y} \right)^c \cdot \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} \\
&\overset{(22)}{=} R_5 \cdot \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^c \cdot \left(\frac{e(g_1, g_2)}{e(Ah^\alpha, w)} \right)^{-c} = R_5 \quad .
\end{aligned}$$

$\square$

Lemma 2. *The interactive protocol is auxiliary-input zero-knowledge under the XDH assumption.*

Proof. We have to show that a simulator can output transcripts of the protocol drawn from the same distribution as real interactions. The simulator $\mathcal{S}$ takes as input a message m, the identity i of a user $\mathcal{U}_i$, and the group public key gpk.

First, the simulator $\mathcal{S}$ picks $A \overset{R}{\leftarrow} G_1$ and $\alpha \overset{R}{\leftarrow} \mathbb{Z}_p$ uniformly at random. The simulator $\mathcal{S}$ sets $T_1 \leftarrow \tilde{g}^{\alpha}$ and $T_2 \leftarrow A \cdot h^{\alpha}$. Note that T_1, T_2 is the ElGamal encryption of A. Under the XDH assumption, the Decisional Diffie Hellman DDH problem is hard in G_1, therefore, by the semantic security of the ElGamal encryption [22], the tuple (T_1, T_2) is drawn from a distribution that is indistinguishable from the output distribution of a particular prover. The remainder of the simulation does not assume knowledge of A, x, y or α.

Second, the simulator $\mathcal{S}$ uses a deterministic hash function $\mathcal{H} : \{0,1\}^* \mapsto \mathbb{Z}_p$, and computes $m' \leftarrow \mathcal{H}(m)$ and $\bar{y} \leftarrow \mathcal{H}(i)$. Simulator $\mathcal{S}$ then computes

$$T_3 \leftarrow e(g_1, g_2)^{\frac{1}{m'+\bar{y}}} \quad , \quad T_4 \leftarrow T_3^{\rho} \quad ,$$
$$T_5 \leftarrow g_2^{\frac{\rho}{m'+\bar{y}}} \quad , \quad T_6 \leftarrow g_2^{\frac{\bar{y}}{\rho}} \quad , \quad T_7 \leftarrow g_1^{\frac{m'}{\rho}} \quad .$$

Note that y is chosen at random during the Join protocol, and although A depends on the value y, this is of no concern to us, due to the semantic security of the ElGamal encryption. We can thus use the hash function $\mathcal{H}$ to simulate the random choice of y of each user $\mathcal{U}_i$.

Next, the simulator $\mathcal{S}$ chooses the challenge $c \overset{R}{\leftarrow} \mathbb{Z}_p$ and the response values $s_\alpha, s_\rho, s_x, s_{\delta_1}, s_{\delta_2} \overset{R}{\leftarrow} \mathbb{Z}_p$ at random. The simulator $\mathcal{S}$ then computes R_1, R_2, R_3, R_4, R_5 as follows:

$$R_1 \leftarrow \tilde{g}_1^{s_\alpha} \cdot T_1^{-c} \quad , \quad R_2 \leftarrow e(T_2, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha} \cdot e(h, g_2)^{-s_{\delta_1}} \cdot \left(\frac{e(g_1, g_2)}{e(T_2, w)}\right)^{-c} \quad ,$$
$$R_3 \leftarrow T_3^{s_\rho} \cdot T_4^{-c} \quad , \quad R_4 \leftarrow T_7^{s_\rho} \cdot (g_1^{m'})^{-c} \quad ,$$
$$R_5 \leftarrow e(T_2, g_2)^{s_x} \cdot e(h, T_6)^{-s_\rho} \cdot e(h, g_2)^{-s_{\delta_2}} \cdot e(h, w)^{-s_\alpha} \cdot \left(\frac{e(g_1, g_2)}{e(T_2, w)}\right)^{-c} \quad .$$

Hence, the simulated transcript (T_1, T_2, T_3, T_4, T_5, T_6, T_7, R_1, R_2, R_3, R_4, R_5, c, s_α, s_ρ, s_x, s_{δ_1}, s_{δ_2}) produced by the simulator $\mathcal{S}$ is indistinguishable from a transcript between a real prover $\mathcal{P}$ and verifier $\mathcal{V}$. $\square$

Lemma 3. *The interactive protocol is sound.*

Proof. We show that an extractor $\mathcal{X}$, with blackbox access in the protocol, can gain knowledge of a valid extended SDH tuple (A, x, y) and the value ρ that randomizes the proof for the output of the VRF y_{DY}.

First, the prover $\mathcal{P}$ sends $T_1, T_2, T_3, T_4, T_5, T_6, T_7$ and R_1, R_2, R_3, R_4, R_5. To the challenge value c, the prover $\mathcal{P}$ responds with $s_\alpha, s_\rho, s_x, s_{\delta_1}, s_{\delta_2}$. The extractor

$\mathcal{X}$ then rewinds the protocol just above the point before the challenge value c was given to prover $\mathcal{P}$. The prover then responds to a new challenge value $c' \neq c$ with $s'_\alpha, s'_\rho, s'_x, s'_{\delta_1}, s'_{\delta_2}$

We denote $\Delta c = c - c'$, $\Delta s_\alpha = s_\alpha - s'_\alpha$, and analogously $\Delta s_\rho, \Delta s_x, \Delta s_{\delta_1}, \Delta s_{\delta_2}$.

If the check (15) is successful, then $R_1 = \tilde{g}_1^{s_\alpha} \cdot T_1^{-c} = \tilde{g}_1^{s'_\alpha} \cdot T_1^{-c'}$. Massaging this equation, we obtain $\tilde{g}_1^{\Delta s_\alpha} = T_1^{\Delta c}$, and further $\tilde{g}_1^{(\Delta s_\alpha / \Delta c)} = T_1$. In order to get an $\hat{\alpha}$ s.t. $T_1 = \tilde{g}_1^{\hat{\alpha}}$ according to (3), the extractor $\mathcal{X}$ sets $\hat{\alpha} = \Delta s_\alpha / \Delta c$.

Similarly, from (17) we have $R_3 = T_3^{s_\rho} \cdot T_4^{-c} = T_3^{s'_\rho} \cdot T_4^{-c'}$, so $T_3^{\Delta s_\rho / \Delta c} = T_4 = T_3^{\hat{\rho}}$, so the extractor $\mathcal{X}$ obtains $\hat{\rho} = \Delta s_\rho / \Delta c$. The same $\hat{\rho}$ can also be obtained form (18).

Next, from (16), we obtain

$$e(T_2, g_2)^{\Delta s_x} \cdot e(h, w)^{-\Delta s_\alpha} \cdot e(h, g_2)^{-\Delta s_{\delta_1}} = \left(\frac{e(g_1, g_2)}{e(T_2, w)} \right)^{\Delta c} \quad .$$

Taking Δc-th roots, and letting $\hat{x} = \Delta s_x / \Delta c$ and $\hat{x}\hat{\alpha} + \hat{y} = \Delta s_{\delta_1} / \Delta c$, we obtain

$$\begin{aligned} e(T_2, g_2)^{\hat{x}} \cdot e(h, w)^{-\hat{\alpha}} \cdot e(h, g_2)^{-\hat{x}\hat{\alpha} - \hat{y}} &= \frac{e(g_1, g_2)}{e(T_2, w)} \quad , \\ e(T_2, g_2)^{\hat{x}} \cdot e(h^{-\hat{\alpha}}, w) \cdot e(h^{-\hat{\alpha}}, g_2)^{\hat{x}} &= \frac{e(h, g_2)^{\hat{y}} \cdot e(g_1, g_2)}{e(T_2, w)} \quad , \\ e(T_2 h^{-\hat{\alpha}}, g_2)^{\hat{x}} \cdot e(T_2 h^{-\hat{\alpha}}, w) &= e(h, g_2)^{\hat{y}} \cdot e(g_1, g_2) \quad . \end{aligned}$$

Thus, by setting $\hat{A} = T_2 h^{\hat{\alpha}}$ the extractor $\mathcal{X}$ obtains a tuple $(\hat{A}, \hat{x}, \hat{y})$ that satisfies (21) and is thus a valid extended SDH-tuple. Perforce, T_1, T_2 is the encryption of $\hat{A}$.

Furthermore, from (19), we obtain in a similar manner, by letting $\hat{\alpha}\hat{x} = \Delta s_{\delta_2}$,

$$\begin{aligned} e(T_2, g_2)^{\Delta s_x} \cdot e(h, T_6)^{-\Delta s_\rho} \cdot e(h, g_2)^{-\Delta s_{\delta_2}} \cdot e(g, w)^{\Delta s_\alpha} &= \left(\frac{e(g_1, g_2)}{e(T_2, w)} \right)^{\Delta c} \quad , \\ e(\hat{A}, g_2)^{\hat{x}} \cdot e(\hat{A}, w) &= e(g_1, g_2) \cdot e(h, T_6)^{\hat{\rho}} \quad . \end{aligned}$$

Thus, T_6 must have necessarily been computed as $T_6 = g_2^{\hat{y}/\hat{\rho}}$. Because (13) and (14) hold, it is now straightforward to see that the value T_3 of the VRF y_{DY} (4) was computed correctly. □

Theorem 2 (Correctness). *The group signature scheme $\mathcal{LGS}$ is correct.*

Proof. In the random oracle mode, the correctness of the group signature scheme $\mathcal{LGS}$ follows directly from Theorem 1, where we have shown that the underlying interactive protocol is a honest-verifier zero-knowledge proof of knowledge.

Because the group signature σ is a proof of knowledge of an extended SDH-tuple (A, x, y), by Lemma 1, the signature σ will always be accepted by the verifier.

By the soundness of the proof of knowledge, shown in Lemma 3, (T_1, T_2) is a correct ElGamal encryption of A with the public key $\mathsf{ompk} = h = \tilde{g}^{\xi}$ corresponding to the opening manager $\mathcal{OM}$'s secret key $\mathsf{ok} = \xi$. Thus, the opening manager $\mathcal{OM}$ can always open a group signature σ. Furthermore, the opening manager $\mathcal{OM}$ can always create a proof τ of the correct decryption of (T_1, T_2) by generating a zero knowledge proof of knowledge of equality of the logarithms T_1^{ξ} and $\tilde{g}^{\xi}$. □

Theorem 3 (Limited-linkability). *The group signature scheme $\mathcal{LGS}$ is limited-linkable.*

Proof. The *limited-linkability* property follows directly from Theorem 1 and from the deterministic nature of the VRF $y_{DY} = e(g_1, g_2)^{1/(m'+y)}$.

Since a group signature σ is a proof of knowledge of an extended SDH-tuple (A, x, y), and by the definition of the correctness of the $\mathcal{LGS}$ signature, proved in Theorem 2, any signature that verifies correctly can always be opened, such that the correct user $\mathcal{U}_i$, as registered in the registration table reg, is revealed.

From the soundness property of the underlying honest-verifier zero-knowledge proof of knowledge, proved in Lemma 3, we can deduct that if a different $m' \neq m$ or a different $y' \neq y$ was chosen for the computation of T_3 than the y in the extended SDH-tuple (A, x, y), then the signature σ would not verify correctly. However, due to the deterministic nature of the VRF y_{DY}, the tag T_3 will always be the same when we have the same m and y, and thus, the Link algorithm will detect a pair of signatures on the same message m that can be opened to the same signer identity i.

Thus it is easy to see that an attacker has no chance to win the *limited linkability* experiment $\mathbf{Exp}^{\mathrm{link}}_{\mathcal{LGS},A}(k)$. □

Theorem 4 (Anonymity). *The group signature scheme $\mathcal{LGS}$ is anonymous if the ElGamal encryption is semantically secure against chosen plaintext attacks, and if at least $n/2$ of the n parties involved in the GJKR protocol are honest, and if the DBDHI assumption holds.*

Proof. Suppose adversary $\mathcal{A}$ breaks the anonymity of the $\mathcal{LGS}$ scheme. We construct a polynomial time algorithm $\mathcal{B}$ that breaks the semantic security of the ElGamal encryption.

Algorithm $\mathcal{B}$ is given a group $\bar{G}_1$, a generator $\bar{g}_1$ of $\bar{G}_1$, and a public key $\bar{h} = \bar{g}_1^{\bar{x}}$.

First, algorithm $\mathcal{B}$ simulates the Setup algorithm and gives $\mathsf{params} = (\bar{G}_1, G_2, G_T, e, \psi)$ to the adversary.

Adversary $\mathcal{A}$ has to query the DKG oracle, otherwise the challenge oracle $\mathsf{Ch}_{b,m}$ will fail. Algorithm $\mathcal{B}$ answers the first query to the DKG oracle by simulating the GJKR protocol under the assumption that $n/2$ of the n parties involved in the GJKR protocol are honest. So algorithm $\mathcal{B}$ takes $g_2 \overset{R}{\leftarrow} G_2$ at random and computes $g_1 \leftarrow \psi(g_2)$. Algorithm $\mathcal{B}$ returns the tuple $(g_1, g_2, \bar{g_1}, \mathsf{ompk} = \bar{h})$.

The adversary $\mathcal{A}$ is responsible for adding honest users via the SndToU oracle. During the SndToU oracle, Algorithm $\mathcal{B}$ plays the users' part by creating new user identities i and personal key pairs $(\mathsf{upk}[i], \mathsf{usk}[i])$, and engaging in the Join protocol with adversary $\mathcal{A}$, who plays the group manager $\mathcal{GM}$.

Requests to the GSig and USK oracles can be easily answered by algorithm $\mathcal{B}$ because $\mathcal{B}$ holds the group member signing keys of all honest users.

If adversary $\mathcal{A}$ sends a query to the challenge oracle, algorithm $\mathcal{B}$ gives A_{i_0}, A_{i_1} of the distinct honest users $\mathcal{U}_{i_0}$,$\mathcal{U}_{i_1}$'s extended SDH-tuple $(A_{i_0}, x_{i_0}, y_{i_0})$, $(A_{i_1}, x_{i_1}, y_{i_1})$ to the semantic security challenger of the ElGamal encryption. The challenger takes a bit $\hat{b} \xleftarrow{R} \{0,1\}$ at random and returns the ElGamal encryption T_1, T_2 of $A_{i_{\hat{b}}}$, using the secret key $\bar{x}$, which corresponds to the public key $\bar{h} = \bar{g}_1^{\bar{x}}$. Algorithm $\mathcal{B}$ takes a message $m \leftarrow \{0,1\}^l$ uniformly at random, chooses any arbitrary value $j \leftarrow \mathbb{Z}$, and uses the simulator $\mathcal{S}$ from Lemma 2 with input m, j to produce the remaining values of a simulated transcript of a group signature $\sigma = (T_1, T_2, T_3, T_4, T_5, T_6, T_7, c, s_\alpha, s_\rho, s_x, s_{\delta_1}, s_{\delta_2})$. Algorithm $\mathcal{B}$ then gives the group signature σ and the message m to the adversary $\mathcal{B}$.

The reason why algorithm $\mathcal{B}$ can choose any aribtrary value j is (1) because the tag T_3, computed by the VRF $y_{DY} = e(g_1, g_2)^{\frac{1}{m'+y}}$, which computes T_3 with $y = \mathcal{H}(j)$ and $m' = \mathcal{H}(m)$, is indistinguishable from random under the DBDHI assumption for every distinct value m. We guarantee the distinctness of m by taking m uniformly at random from $\{0,1\}^l$, where l is large enough so that the probability to chose the same message twice is negligible. And (2) because the randomizing value ρ perfectly hides the user's actual value y in T_5 and T_6.

The adversary $\mathcal{A}$ finally outputs a bit d, which algorithm $\mathcal{B}$ returns as a answer to his own challenge. Whenever adversary $\mathcal{A}$ makes the right guess during his challenge, algorithm $\mathcal{B}$ makes the right guess concerning the encrypted value $A_{i_{\bar{B}}}$, against the semantic security property of the ElGamal encryption. $\square$

Because we have shown that Theorem 1 holds, and we use identical definitions of the *traceability* and the *non-frameability* properties, the following two theorems can be shown by analogously following the corresponding proofs in DP [9].

Theorem 5 (Traceability). *The group signature scheme $\mathcal{LGS}$ is traceable.*

Theorem 6 (Non-Frameability). *The group signature scheme $\mathcal{LGS}$ is non-frameable.*

Acknowledgements

This work was partly supported by the *Marshall Plan Foundation*. Thanks to *Adam Doupé* for proofreading and commenting on a much earlier draft of this paper.

References

1. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assump-

tions. In: Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques. pp. 614–629. EUROCRYPT'03, Springer-Verlag, Berlin, Heidelberg (2003)
2. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on Computer and communications security. pp. 62–73. CCS '93, ACM, New York, NY, USA (1993)
3. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) Topics in Cryptology CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer Berlin / Heidelberg (2005)
4. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology - EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer Berlin / Heidelberg (2004)
5. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Advances in Cryptology—EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Berlin: Springer-Verlag (2004), available at `http://www.cs.stanford.edu/~xb/eurocrypt04a/`
6. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Advances in Cryptology – CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Berlin: Springer-Verlag (2004)
7. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Cramer, R. (ed.) Advances in Cryptology EUROCRYPT 2005, LNCS, vol. 3494, pp. 566–566. Springer Berlin / Heidelberg (2005)
8. Chaum, D., van Heyst, E.: Group signatures. In: Proceedings of Eurocrypt 1991. pp. 257–265. LNCS, Springer-Verlag (1991)
9. Delerablée, C., Pointcheval, D.: Dynamic fully anonymous short group signatures. In: VIETCRYPT. pp. 193–210 (2006)
10. Dodis, Y., Yampolskiy, R.: A verifiable random function with short proofs and keys. In: In Public Key Cryptography, volume 3386 of LNCS. pp. 416–431. Springer-Verlag (2005)
11. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Proceedings on Advances in cryptology – CRYPTO '86. pp. 186–194. Springer-Verlag, London, UK (1987)
12. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. 20, 51–83 (January 2007)
13. Liu, J., Wei, V., Wong, D.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) Information Security and Privacy. Lecture Notes in Computer Science, vol. 3108, pp. 325–335. Springer Berlin / Heidelberg (2004)
14. Micali, S., Vadhan, S., Rabin, M.: Verifiable random functions. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science. pp. 120–. FOCS '99, IEEE Computer Society, Washington, DC, USA (1999)
15. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing. pp. 427–437. STOC '90, ACM, New York, NY, USA (1990)
16. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology. pp. 129–140. CRYPTO '91, Springer-Verlag, London, UK (1992)
17. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology. pp. 552–565. ASIACRYPT '01, Springer-Verlag, London, UK (2001)

18. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology. pp. 239–252. CRYPTO '89, Springer-Verlag, London, UK, UK (1990)
19. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques. pp. 256–266. EUROCRYPT'97, Springer-Verlag, Berlin, Heidelberg (1997)
20. Tsang, P., Wei, V.: Short linkable ring signatures for e-voting, e-cash and attestation. In: Deng, R., Bao, F., Pang, H., Zhou, J. (eds.) Information Security Practice and Experience. Lecture Notes in Computer Science, vol. 3439, pp. 48–60. Springer Berlin / Heidelberg (2005)
21. Tsang, P., Wei, V., Chan, T., Au, M., Liu, J., Wong, D.: Separable linkable threshold ring signatures. In: Canteaut, A., Viswanathan, K. (eds.) Progress in Cryptology - INDOCRYPT 2004. Lecture Notes in Computer Science, vol. 3348, pp. 337–376. Springer Berlin / Heidelberg (2005)
22. Tsiounis, Y., Yung, M.: On the security of elgamal based encryption. In: Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography. pp. 117–134. PKC '98, Springer-Verlag, London, UK (1998)