Fair Traceable Multi-Group Signatures

Vicente Benjumea[1], Seung Geol Choi[2], Javier Lopez[1], and Moti Yung[3]

[1] Computer Science Dept. University of Malaga, Spain
{benjumea, jlm}@lcc.uma.es
[2] Computer Science Dept. Columbia University, USA
sgchoi@cs.columbia.edu
[3] Google Inc. & Computer Science Dept. Columbia University, USA
moti@cs.columbia.edu

Abstract. This paper presents fair traceable multi-group signatures (FTMGS), which have enhanced capabilities, compared to group and traceable signatures, that are important in real world scenarios combining accountability and anonymity. The main goal of the primitive is to allow multiple groups that are managed separately (managers are not even aware of the other ones), yet allowing users (in the spirit of the Identity 2.0 initiative) to manage what they reveal about their identity with respect to these groups by themselves. This new primitive incorporates the following additional features.

- While considering multiple groups it discourages users from sharing their private membership keys through two orthogonal and complementary approaches. In fact, it merges functionality similar to credential systems with anonymous type of signing with revocation.
- The group manager now mainly manages joining procedures, and new entities (called fairness authorities and consisting of various representatives, possibly) are involved in opening and revealing procedures. In many systems scenario assuring fairness in anonymity revocation is required.

We specify the notion and implement it in the random oracle model.

1 Introduction

Group signatures. Group signatures, introduced by Chaum and Van Heyst [11], and later studied and improved [9, 2, 23, 25], were a major step in designing cryptographic primitives supporting anonymity. In these schemes, users join groups and issue signatures on behalf of the group. When these signatures are verified, we learn that some member of the group generated them, but not which one. It is also impossible to link two signatures generated by the same member of the group. However, the *group manager* has the capability of opening a signature and trace its signer among the members of the group (in [23] managing join of users and tracing by separate authorities was suggested).

Multi-group signatures and sharing private keys. Ateniese et al. [3] extended group signatures to deal with the case where a single anonymous user has to prove that she is simultaneously a member of several groups. In this scenario, a multi-group membership is proved by zero-knowledge proof of equality on some discrete logarithms in the signatures from different groups. Though multi-group signatures are based on the ability of linking some designated group signatures (via equality proofs), such a fact does not affect the main properties of group signatures such as anonymity and unlinkability for other signatures. Note that the multi-group feature is very interesting in anonymous authorization scenarios, since in these environments it is quite common for a user to prove simultaneous possession of some properties in order to be authorized to carry out some transaction. However, the scheme due to [3], as mentioned in the paper, presents some problems when linking signatures from groups that are managed separately with unrelated group keys.

Also, if a user is able to share some of the private keys with other users in a multi-group anonymous environment, it is a severe handicap in a system where privileges depend upon membership to some groups, since this sharing of private keys would undermine the whole system assumptions.

Embedding some valuable information into sensitive data is a commonly used method to dissuade users from sharing their private key. Dwork et al. [12] embedded some user's valuable information, such as the credit card number, into a key in order to protect digital content from illegal redistribution. Also, Goldreich et al. [18] presented several schemes to deter propagation of secondary secret keys in the field of self-delegation of personalized rights. Moreover, Lysyanskaya et al. [24] embedded a user's master secret key into the secret that allows a user to prove possession of a credential on a pseudonym.

Traceable signatures. In group signatures, under critical circumstances such as dishonest behavior, the group manager is able to open a signature and identify the dishonest user. If a user is under suspicion, a judge may decide to identify which transactions were performed by such user. In this latter case, the group manager has to open all selected signatures to identify which ones were issued by the user under suspicion. This approach has two main disadvantages: (i) it discloses the identity of the issuers of the signatures, violating their privacy, even for honest members; and (ii) the group manager has to be involved in this heavy task, being a potential bottleneck for scalability.

Taking into account the above scenario, Kiayias et al. [22] introduced *traceable signatures* as a group signature scheme with further refinement of tracing under anonymity. This primitive incorporates a feature that enables the group manager to reveal a trapdoor for a given member of the group. The trapdoor allows the tracing agents to identify which signatures were issued by the member under suspicion without revealing any further information. This approach benefits us by removing the aforementioned disadvantages: (i) the privacy of non-involved members remains unaltered; and (ii) the group manager is relieved from this task, which can be performed by several tracing agents. Additionally, traceable signatures also incorporate a claiming facility, that allows a member to claim that a given signature was issued by herself.

Splitting roles of the group manager. It is usually accepted that the group manager is a trusted party with respect to joining new members to the group. However, in many scenarios the group manager is a party in interest, and therefore it can not be trusted with respect to user's privacy. For example, a company can manage a group for employees and a group for clients, and can be trusted with respect to joining users that are actually employees or clients respectively. However, the company does not offer any guarantee with respect to keep users' privacy, specially when they carry out anonymous transactions with the company itself.

We note that in group signatures, there have been several proposals to divide the duties of the group manager into two entities [23, 25], one responsible for joining new members and the other one responsible for opening signatures. However, in the context of traceable signatures such splitting of duties has not been considered yet.

Our contribution. This paper presents Fair Traceable Multi-Group Signatures (FTMGS, pronounced: fat-mugs), a new primitive that supports anonymity with extended concerns that rise in realistic scenarios. It can be regarded as a primitive that has the flavor of anonymous signatures with various revocations but with a refined notion of access control (via multiple groups) and thus supporting anonymous activities in a fashion similar to anonymous credential systems [24, 7]. The main issues that make this primitive suitable to various trust relationships are:

- It provides anonymous and unlinkable signatures in the way group and traceable signatures do.

– It includes multi-group features to guarantee that several signatures have been issued by the same anonymous user with no detriment of user's anonymity. This allows limited local linkability most useful in many cases (linking are user controlled).
– It includes a mechanism to dissuade the group members from sharing their private membership keys. This is very useful in increasing the incentive for better "access control" to anonymous credentials.
– It further splits the duties of the group manager into several authorities, allowing better control over opening and tracing operations. Now the group manager manages only joining. Newly introduced parties, whom we call *fairness authorities*, by cooperating with each other, manage opening signatures and revealing tracing trapdoors. A single fairness authority alone cannot do the opening or revealing. In this way, a user's sensitive information can be guaranteed only to be disclosed when there exist enough reasons.

Let us next further elaborate on some of the above characteristics of the primitive. With respect to multi-group features, as opposed to the scheme introduced in [3], group management is separate and groups are formed where group managers are not necessarily aware of each other. There is no coordination and group keys are solely under the control of its group manager (based on some accepted security parameters). At the user level, however, management of identity is up to herself; linking signatures and claiming identity are executed according to her desire. This latter approach is in concordance with the identity 2.0 [19] effort.

General scenario for this new primitive. The group manager creates a group with the collaboration of designated fairness authorities[4]. A user, that has been authorized by some external procedure, is able to join the group by engaging in an interactive protocol with the group manager. The external user's authentication can be based on her identity[5] or even an anonymous authentication supported by this new primitive. At the end of the procedure, the group manager gets some sensitive data regarding the new member (i.e. join transcript with authentication information), and the user gets a membership private key that enables her to issue signatures on behalf of the group.

When a user wants to carry out a transaction with a server, she sometimes has to generate a proof to show she has the required privilege. This proof usually implies that she belongs to several groups. In this case, she issues suitable signatures for the involved groups, and establishes a link among them to guarantee that they have been issued by the same single anonymous user.

Under critical circumstances, fairness authorities and the judge open a signature to identify a malicious user. If necessary, they may also reveal her tracing trapdoor so that tracing agents, using the trapdoor, trace all the transactions she issued.

2 A Model for Fair Traceable Multi-Group Signatures

In this section, we present our model for fair traceable multi-group signatures. We describe the types of entities and operations in the system. See Figure 1 for the notations used in the model.

Participating Entities. There are five types of participating entities: users, group managers, fairness authorities, tracing agents, and the judge.

[4] Roughly Speaking, in order to split the roles, the group manager creates a part of group key related to joining procedure, and the fairness authorities create the other part related to opening and tracing procedures.
[5] A certified public key via PKI based on discrete logarithm (e.g., DSA, El-Gamal signatures, Schnorr signatures).

1. parameters
   ν: security parameter ζ: the number of fairness authorities
2. G^v: the group (i.e., service provider) with index v
3. participating entities
   GM^v: group manager of the group G^v $\mathrm{U_i}$: user i J: judge
   $\mathrm{FA_j}^v$: j-th fairness authority of G^v $\mathrm{TA_j}^v$: j-th tracing agent of G^v
4. various keys and data
   gpk^v: group public key of G^v gsk^v: private key of GM^v
   fgsk^v: fairness private key of G^v $\mathsf{fgsk}_j{}^v$: $\mathrm{FA_j}^v$'s share for fgsk^v
   umk_i: master secret key of $\mathrm{U_i}$. usk_i^v: signing key of $\mathrm{U_i}$ w.r.t. G^v
   jlog_i^v: join transcript generated while $\mathrm{U_i}$ joins the group G^v
   σ: signature auth_i: authentication string issued by $\mathrm{U_i}$
   $\tau_i{}^v$: tracing key for $\mathrm{U_i}$ in G^v
   ω_σ: member reference (locator used to search for the corresponding join transcript)
5. predicates w.r.t jlog_i^v, usk_i^v and auth_i
   $\mathtt{mkey}(\mathsf{usk}_i^v$ or $\mathsf{auth}_i)$: master secret key of usk_i^v or auth_i
   $\mathsf{mref}(\mathsf{jlog}_i^v$ or $\mathsf{usk}_i^v)$: member reference of jlog_i^v or usk_i^v
   $\mathsf{tkey}(\mathsf{jlog}_i^v$ or $\mathsf{usk}_i^v)$: tracing key of jlog_i^v or usk_i^v
6. etc
   $\mathtt{acc}$: accept $\perp$: error m: message γ: challenge string

Fig. 1. Legends of our model for fair traceable multi-group signatures

- Users join groups and generate signatures, claims, link-claims, etc. Usually, users join groups if the group manager authorizes it.
- Each group has one group manager, which manages joining and the corresponding database.
- Each group has multiple fairness authorities. They cooperate together, under the judge's supervision, to either open a signature or to reveal a tracing key of a user under suspicion.
- Each group has multiple tracing agents. They trace the transaction databases and find out signatures related to the revealed tracing key.
- The judge manages opening and tracing operations with the help of the group manager, fairness authorities and tracing agents.
- It is assumed that an external PKI provides legal binding between users and public keys, such that users do not want to lend their corresponding private keys to any other user, since actions performed under these public keys entail legal responsibilities.

Operations. A fair traceable multi-group signature scheme consists of the following operations. Two operations are newly added compared with original traceable signatures: ClaimLink and VerifyLink. The rest of the operations are slightly changed so that fairness authorities may be involved.

1. Setup$(1^\nu, \zeta)$. This interactive procedure generates the group public key gpk^v, the secret key gsk^v for the group manager and the secret keys $\{\mathsf{fgsk}_j{}^v\}_{j=1}^{\zeta}$ for the fairness authorities.
2. Join$(\mathsf{gpk}^v, [\mathsf{gsk}^v], [\mathsf{umk}_i])$. This interactive procedure is used when a user joins a group, where the group public key gpk^v is common input, and the group secret key gsk^v and user master key umk_i are private inputs of the group manager and the user respectively. As result, the group manager gets a join transcript jlog_i^v and the user gets a membership private key usk_i^v.

3. JoinOnAuth($\mathsf{gpk}^v, \mathsf{auth}_i, [\mathsf{gsk}^v], [\mathsf{umk}_i]$). This interactive procedure is used when an authenticated user joins a group, where the group public key gpk^v and user authentication string auth_i are common input, and the group secret key gsk^v and user master key umk_i are private inputs of the group manager and the user respectively. Depending on the situation, the authentication string auth_i can be a public key, a digital signature, a traceable signature of another group or combination of them. We require that the key used to generate auth_i should have umk_i as its part. As result, the group manager gets a join transcript jlog_i^v and the user gets a membership private key usk_i^v.
4. Sign($\mathsf{gpk}^v, \mathsf{usk}_i^v, m$). With this algorithm, a member generates a group signature σ on message m.
5. Verify($\mathsf{gpk}^v, m, \sigma$). Any entity can verify a signature σ on a message m.
6. Open($\mathsf{gpk}^v, \sigma, [\{\mathsf{fgsk}_j{}^v\}_{j=1}^{\zeta}]$). This interactive procedure opens a signature σ, generating a reference ω_σ to the member that issued it. It requires the private inputs of the fairness authorities' secret keys $\{\mathsf{fgsk}_j{}^v\}_{j=1}^{\zeta}$.
7. Reveal($\mathsf{gpk}^v, \mathsf{jlog}_i^v, [\{\mathsf{fgsk}_j{}^v\}_{j=1}^{\zeta}]$). This interactive procedure reveals the member tracing key ${\tau_i}^v$ from the join transcript jlog_i^v. It requires the private inputs of the fairness authorities' secret keys $\{\mathsf{fgsk}_j{}^v\}_{j=1}^{\zeta}$.
8. Trace($\mathsf{gpk}^v, \sigma, {\tau_i}^v$). This tracing algorithm allows the tracing agents to check if the signature σ is associated with the tracing key ${\tau_i}^v$.
9. Claim($\mathsf{gpk}^v, \mathsf{usk}_i^v, \sigma, \gamma$). This algorithm generates an authorship proof π for a signature σ on the challenge γ.
10. VerifyClaim($\mathsf{gpk}^v, \sigma, \gamma, \pi$). Any entity can verify an authorship proof π of a signature σ on a challenge γ.
11. ClaimLink($\mathsf{gpk}^{v_1}, \mathsf{usk}_i^{v_1}, \sigma^1, \mathsf{gpk}^{v_2}, \mathsf{usk}_i^{v_2}, \sigma^2, \gamma$). This algorithm generates a link proof λ between two signatures σ^1, σ^2 on the challenge γ, if the two signatures have been issued with the same master key.
12. VerifyLink($\mathsf{gpk}^{v_1}, \sigma^1, \mathsf{gpk}^{v_2}, \sigma^2, \gamma, \lambda$). Any entity can verify a link proof λ between two signatures σ^1, σ^2 on a challenge γ.

3 Preliminaries

Notation. We denote $\{0, \dots, \ell - 1\}$ by $[\ell]$. Throughout the paper we work mostly in the group of quadratic residues modulo n, denoted by $QR(n)$, with $n = pq$, for safe primes p and q ($p = 2p' + 1$ and $q = 2q' + 1$). Let the security parameter $\nu := \lceil \log p'q' \rceil$. We define the following sets:

$$\Lambda = \{1, \dots, 2^{\nu/4} - 1\}, \quad \mathrm{M} = \{1, \dots, 2^{\nu/2} - 1\},$$
$$\Gamma = \{2^{3\nu/4-1} + 1, \dots, 2^{3\nu/4-1} + 2^{\nu/2} - 1\},$$
$$\Lambda_\epsilon^k = \{1 + \Delta_{\nu/4}, \dots, 2^{\nu/4} - 1 - \Delta_{\nu/4}\}, \quad \mathrm{M}_\epsilon^k = \{1 + \Delta_{\nu/2}, \dots, 2^{\nu/2} - 1 - \Delta_{\nu/2}\},$$
$$\Gamma_\epsilon^k = \{2^{3\nu/4-1} + 1 + \Delta_{\nu/2}, \dots, 2^{3\nu/4-1} + 2^{\nu/2} - 1 - \Delta_{\nu/2}\},$$

where $\Delta_\mu = 2^{\mu-1} - 2^{\frac{\mu-2}{\epsilon} - k}$ for $\epsilon > 1$ and $k > 128$. Sometimes we will call the sets $\Lambda, \mathrm{M}, \Gamma$ spheres, and $\Lambda_\epsilon^k, \mathrm{M}_\epsilon^k, \Gamma_\epsilon^k$ inner spheres.

Assumptions. Below are listed the assumptions we use in the paper.

Definition 1. (Strong-RSA [4]). *Given $n = pq$, where p and q are both safe primes, and $z \in QR(n)$, it is hard to find $u \in \mathbb{Z}_n$ and $e > 1$ such that $u^e = z \pmod{n}$.*

Definition 2. (Decision Composite Residuosity [26]) *n is as above. Consider the group $\mathbb{Z}_{n^2}$ and the subgroup $\mathbf{P}$ of $\mathbb{Z}_{n^2}^*$ consisting of all n-th powers of elements in $\mathbb{Z}_{n^2}^*$, it is hard to distinguish random elements of $\mathbb{Z}_{n^2}^*$ from random elements of $\mathbf{P}$.*

Definition 3. (Discrete-Logarithm) *Given two values a, b of a multiplicative group Z_n^* or $\mathbb{Z}_{n^2}^*$ it is hard to find x such that $a^x = b$ even if the factorization of n is known.*

Definition 4. (Decisional Diffie-Hellman [22]) *Given a generator g of a cyclic group $QR(n)$ where n is as above, define $\mathcal{D} := \{(g, g^x, g^y, g^{xy}) : x \in \mathcal{B}_1, y \in \mathcal{B}_2\}$ and $\mathcal{R} := \{(g, g^x, g^y, g^z) : x \in \mathcal{B}_1, y \in \mathcal{B}_2, z \in \mathcal{B}_3\}$, where $\mathcal{B}_i$ $(1 \le i \le 3)$ is $\varLambda, \mathrm{M}, \varGamma$, or $[p'q']$. Define the DDH advantage of $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH}}(\nu) = \Big| \mathrm{Pr}_{v \in \mathcal{D}}[\mathcal{A}(1^\nu, v) = 1] - \mathrm{Pr}_{v \in \mathcal{R}}[\mathcal{A}(1^\nu, v) = 1] \Big|.$$

Then for any PPT algorithm $\mathcal{A}$, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH}}(\nu) = neg(\nu)$.
Kiayias et al. [23] showed that DDH over $QR(n)$ does not depend on the hardness of factoring, that is, if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH}}(\nu) = neg(\nu)$ for a cyclic group modulo a safe prime, then $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH\text{-}KF}}(\nu) = neg(\nu)$ for the cyclic group of quadratic residues modulo a safe composite with known factorization.

Definition 5. (Cross Group DDH [20]) *Given generators g_1, g_2 of $QR(n_1)$ and $QR(n_2)$ where n_1 and n_2 are as above, $n_1 \neq n_2$, and $\nu_1 = \nu_2$, we define $\mathcal{D} := \{(g_1, g_1^x, g_2, g_2^x) : x \in \mathcal{B}_1 \cap \mathcal{B}_2\}$ and $\mathcal{R} := \{(g_1, g_1^x, g_2, g_2^y) : x, y \in \mathcal{B}_1 \cap \mathcal{B}_2\}$, where $\mathcal{B}_i$ $(1 \le i \le 2)$ is $\varLambda_i, \mathrm{M}_i, \varGamma_i$, or $[p_i'q_i']$. Define the advantage of $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CG\text{-}DDH}}(\nu_1) = \Big| \mathrm{Pr}_{v \in \mathcal{D}}[\mathcal{A}(1^{\nu_1}, v) = 1] - \mathrm{Pr}_{v \in \mathcal{R}}[\mathcal{A}(1^{\nu_1}, v) = 1] \Big|.$$

Then for any PPT algorithm $\mathcal{A}$, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CG\text{-}DDH}}(\nu_1) = neg(\nu_1)$.
We also assume that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CG\text{-}DDH\text{-}KF}}(\nu_1) = neg(\nu_1)$ for the cyclic group of quadratic residues modulo a safe composite with known factorization.

In other words, the CG-DDH assumption states that it is infeasible to test equality of discrete logs across groups.

4 Building Blocks

Signature of Knowledge of Discrete Logarithm. We use the notation due to Camenisch and Stadler [9] for signatures of knowledge of discrete logarithms. For example, $\mathrm{SK}\{(a, b) :\ y = g^a;\ z = h^a f^b\}(m)$ denotes a *signature of knowledge of integers a and b on m such that $y = g^a$ and $z = h^a f^b$ holds.*

Kiayias et al. [22] presented a scheme for signatures of knowledge in discrete-log relation sets and proved its security. Here we only briefly describe how it works by taking an example[6]. Consider the following signature of knowledge of a discrete logarithm: SK$\{\ (x) : y = g^x \pmod n\ ;\ \gamma = \beta^x \pmod \rho\ \}(m)$. It can be represented as the following triangular discrete-log relation set:

$$\left[\begin{array}{c|c} \text{Objects}: \ y \ \ g \ \ \gamma \ \ \beta & \\ \hline y = g^x : -1\ x \ \ 0 \ \ 0 & (\bmod\ n) \\ \gamma = \beta^x :\ \ 0 \ \ 0\ -1\ x & (\bmod\ \rho) \end{array}\right].$$

The signature of knowledge for this relation is $\langle c, s_x \rangle$, where $B_1 = g^{t_x} \pmod n$ (t_x is chosen randomly), $B_2 = \beta^{t_x} \pmod \rho$, $c = \mathrm{Hash}(B_1, B_2, g, n, y, \beta, \rho, \gamma, \mathit{env\text{–}data}, m)$ and $s_x = t_x - cx$. Verification is done by computing $B_1' = g^{s_x} y^c \pmod n$, $B_2' = \beta^{s_x} \gamma^c \pmod \rho$ and checking if $c \overset{?}{=} \mathrm{Hash}(B_1', B_2', g, n, y, \beta, \rho, \gamma, \mathit{env\text{–}data}, m)$.

[6] For simplicity, we ignored details on range checking of discrete log variable. See [22] for more technical detail.

DL-Representations. In the exposition below we use some fixed values $a_0, a, b \in QR(n)$.

Definition 6. (DL-Representation [22]) *A discrete-log representation is a tuple $\langle A, e, x, x' \rangle$ such that $A^e = a^x b^{x'} a_0$ holds where $x \in \mathrm{M}$, $x' \in \Lambda$ and $e \in \Gamma$.*

Note that we changed the range of x to M (originally the range was Λ) in order to ensure the hardness of the following problem in the adaptive setting.

Definition 7. (Adaptive One-More Representation Problem) *Let Q_{rep} be an oracle that, on input $x'_i \in \Lambda$, ouputs A_i, e_i, x_i such that $A_i^{e_i} = a^{x_i} b^{x'_i} a_0$ holds with $x_i \in \mathrm{M}$, e_i is a prime number in Γ (i.e., $\langle A_i, e_i, x_i, x'_i \rangle$ is a DL-representation.) The "adaptive one-more representation problem" is to find another DL- representation where it is allowed to query to the Q_{rep} oracle K times adaptively.*

Lemma 1. *Under the Strong RSA assumption, the adaptive one-more representation problem is hard.*

Non-adaptive Drawing of Random Powers. Kiayias et al. [22] showed an efficient two-party protocol for the non-adaptive drawing of random powers, where n and $a \in QR(n)$ are the common input parameters to the protocol. As result one party gets a random secret x in a certain sphere, and the other party gets $a^x \in QR(n)$ with the guarantee that the unknown x was non-adaptively selected at random. See [22] for more details.

Threshold Cryptosystems. It is often dangerous for only one person to have the power of decryption. By distributing the decryption ability, threshold cryptosystems [28, 10, 15, 1] avoid the risk. Following the notation due to [15], a (t, ζ)-threshold cryptosystem consists of the following components:

- A key generation algorithm $\langle pk, \{sk_j\}_{j=1}^{\zeta}, \{vk_j\}_{j=1}^{\zeta} \rangle \leftarrow K(1^\nu, t, \zeta)$, where 1^ν is a security parameter, ζ is the number of decryption servers, t is the threshold parameter, pk is the public key, and sk_j (resp. vk_j) is the secret key share (resp. the verification key) of the j-th decryption server.
- An encryption algorithm $c \leftarrow E(pk, m)$, where m is cleartext, and c is ciphertext.
- Partial decryption algorithms $\sigma_j \leftarrow D_j(sk_j, c)$, for $j \in [1, \zeta]$. Here, σ_j is called a decryption share, and it may include a verification part to achieve robustness.
- A recovery algorithm $m \leftarrow R(c, \{\sigma_j\}_{j=1}^{\zeta}, \{vk_j\}_{j=1}^{\zeta})$, which recovers the plaintext m from the ciphertext c.

The security of threshold cryptography must satisfy two properties: security of the underlying encryption (IND-CPA or IND-CCA2) and robustness. Robustness means that corrupted players should not be able to prevent uncorrupted servers from decrypting ciphertexts. In our scheme, we use two simplified $(\zeta - 1, \zeta)$ threshold IND-CPA cryptosystems by assuming all decryption servers do not abort.

ElGamal Cryptosystem. Consider the following ElGamal encryption scheme [13]:

Let $g \in QR(n)$ be a generator. Let $y = g^x$ be the public key for the secret key x. The encryption of a message m is $(g^r, m \cdot y^r)$ for $r \in_R [1, p'q']$. The decryption of a ciphertext (α, β) is β/α^x.

The threshold version is as follows [27, 16, 17, 6]:

Let $g \in QR(n)$ be a generator. Let $y_j = g^{x_j}$ and x_j be the verification key and the secret key respectively for the j-th decryption server. Let $y = \prod_{j=1}^{\zeta} y_j$ be the encryption public key. Encryption of a message m is as ElGamal above. To decrypt a ciphertext (α, β), each decryption server computes a decryption share $\alpha_j = \alpha^{x_j}$, and proves that $\log_g y_j = \log_\alpha \alpha_j$. The combiner gets the shares and computes $\alpha_\pi = \prod_{j=1}^{\zeta} \alpha_j$. Finally, the combiner recovers the message by computing β/α_π.

Under the DDH assumption, this threshold ElGamal cryptosystem is semantically secure and robust.

Simplified Camenisch-Shoup (sCS) Cryptosystem. The Camenisch-Shoup encryption scheme [8] can be simplified into a semantically secure encryption scheme by removing the CCA checking tag.[7]

Let n be as above: $n = pq$, where p, q are safe primes. In this encryption scheme, multiplications and exponentiations are done in $\mathbb{Z}_{n^2}$. Let $g = g'^{2n}$ where $g' \in_R \mathbb{Z}_{n^2}$. Denote $h = 1 + n$. Then public key is $y = g^x$ and secret key is $x \in_R [1, n^2/4]$. The encryption of a message $m \in \mathbb{Z}_n$ is $(g^r, h^m y^r)$ for $r \in_R [1, n/4]$. To decrypt a ciphertext (α, β), compute $\hat{m} = (\beta/\alpha^x)^{2t}$ for $t = 2^{-1} \pmod n$; if $m = \frac{\hat{m}-1}{n}$ is an integer in $\mathbb{Z}_n$, then output m, otherwise output $\perp$.

The threshold version can be constructed by using the similar technique for the ElGamal encryption, but generating the modulus n, with unknown factorization is done by means of a suitable distributed key generation protocol [14] to guarantee that $QR(n)$ is cyclic and has large prime factors. Under the Decision Composite Residuosity assumption, this threshold sCS cryptosystem is semantically secure and robust.

5 Design of a FTMGS Scheme

Our scheme is based on the original traceable signature scheme from [22][8]; the main differences lie in the setup and join procedures. Here, in our scheme, the user owns a single master key (i.e., x'_i), and this key is embedded in every membership private key of hers. Because this master key is actually the private key corresponding to her public key (e.g., published via the PKI), she is dissuaded from sharing her membership private keys. Moreover, this binding also guarantees that different users have different master keys.

This master key provides a common nexus among all membership private keys that belong to each user, so that she can link any two signatures of hers by proving that the signatures have been issued by membership private keys into which the same master key is embedded. This capability of linking helps our scheme to enjoy multi-group features. Note that even when the user joins the group by means of an anonymous authentication, the join procedure forces her to use the same master key, so that the relationship between her master key and public key still holds.

The group is created by the collaboration among the group manager and the fairness authorities. The GM knows the factorization of n, and therefore is able to join new members. Fairness authorities are also involved in the setup process, in such a way that the keys related opening (o_j) and revealing $(\hat{o}_j)$ are distributed among the fairness authorities. Therefore, opening a signature or revealing a member tracing key requires the participation of fariness authorities.

[7] Jarecki and Shmatikov [21] showed that this holds even when the length of the secret key is shortened. However, in the paper, we use a version with only the CCA checking tag removed.

[8] Which is in turn based on the state of the art in group signatures [2].

Opening a signature is a matter of the distributed decryption, by the fairness authorities (without GM), of part of the signature (i.e., encrypted A_i). Likewise, revealing a member tracing key is also a matter of the distributed decryption, by the fairness authorities (without GM), of the encrypted member tracing key (x_i). This key, however, has to be generated when a user joins the group and cannot be generated randomly without GM. Therefore, we employ a little more complicated mechanism: verifiable encryption. The user encrypts the member tracing key using the public key ($\hat{y} \in QR(\hat{n})$) of verifiable encryption scheme, where the corresponding private key is shared among fairness authorities ($\hat{o}_j$). Still, GM can verify the validity of the encryption without decryption. Later, if the user becomes under suspicion, then the fairness authorities collaborate to decrypt the encrypted form of her member's tracing key.

Finally, the join transcript (jlog_i) also holds some non-repudiable proofs that allow to verify the integrity of the record, making the scheme robust against some kind of database manipulation.

• **System Parameters**. $\epsilon \in \mathbb{R}$ such that $\epsilon > 1$, $k \in \mathbb{N}$, three spheres $\Lambda, \mathrm{M}, \Gamma$ as specified in Section 3, the inner spheres $\Lambda_\epsilon^k, \mathrm{M}_\epsilon^k, \Gamma_\epsilon^k$ and the security parameter ν.

• **FA$_0$–Setup**. Played by the fairness authorities to seed the computation of the public key. It generates a public modulus $\hat{n}$ with unknown factorization by using a suitable distributed key generation protocol [14] that guarantees that $QR(\hat{n})$ is cyclic and its order has no small prime factors. It also selects $\hat{g}' \in_R \mathbb{Z}_{\hat{n}^2}$ and sets $\hat{g} = \hat{g}'^{2\hat{n}}$. Denote the output of this procedure by $\mathsf{fpk}_0 = \langle \hat{n}, \hat{g} \rangle$.

• **FA$_j$–Setup**. Played by j-th fairness authority $\forall\ j \in \{1, \cdots, \zeta\}$ to compute the private and public key pair to manage membership tracing keys. Given $\mathsf{fpk}_0 = \langle \hat{n}, \hat{g} \rangle$, the j-th authority selects a random prime $\hat{o}_j \in \mathbb{Z}_{\hat{n}^2/4}$ and computes $\hat{y}_j = \hat{g}^{\hat{o}_j} \pmod{\hat{n}^2}$. Denote the private and public output by $\mathsf{fsk}_j = \langle \hat{o}_j \rangle$ and $\mathsf{fpk}_j = \langle \hat{y}_j \rangle$ respectively.

• **Group–Setup**. It is an interactive procedure composed of the following procedures: *GM–Init–Setup*, *FA$_j$–Group–Setup*, and *GM–Group–Setup*.

• **GM–Init–Setup**. Played by GM to seed the creation of the group. It generates the prime numbers p, p', q, q' such that $p = 2p' + 1, q = 2q' + 1$, and sets $n = pq$. It also selects $a_0, a, b, g \in_R QR(n)$. Let $\mathsf{gsk} = \langle p, q \rangle$ and $\mathsf{gdef} = \langle n, a_0, a, b, g \rangle$ be the private and public output respectively.

• **FA$_j$–Group–Setup**. Played by j-th fairness authority $\forall\ j \in \{1, \cdots, \zeta\}$ to compute the opening private and public key pair for the group. Given $\mathsf{gdef} = \langle n, a_0, a, b, g \rangle$, it selects $h_j \in_R QR(n)$ and a random prime $o_j \in \mathbb{Z}_{\nu/2}$, then computes $y_j = g^{o_j} \pmod{n}$. Let $\mathsf{fgsk}_j = \langle o_j \rangle$ and $\langle h_j, \mathsf{fgpk}_j \rangle$ be the private and public output respectively, where $\mathsf{fgpk}_j = \langle y_j \rangle$.

• **GM–Group–Setup**. Played by GM to compute the group public key from previously computed public values. Given $\langle \mathsf{gdef}, \mathsf{fpk}_0, \{h_j, \mathsf{fgpk}_j, \mathsf{fpk}_j\}_{j=1}^{\zeta} \rangle$, it computes $h = \prod_{j=1}^{\zeta} h_j \pmod{n}$, $y = \prod_{j=1}^{\zeta} y_j \pmod{n}$ and $\hat{y} = \prod_{j=1}^{\zeta} \hat{y}_j \pmod{\hat{n}^2}$. Let $\mathsf{gpk} = \langle n, a, a_0, b, g, h, y, \hat{n}, \hat{g}, \hat{y} \rangle$ be the public output of the procedure.

• **JoinOnAuth**.[9] Interactive procedure played between a user and the GM when the user joins the group as a new member. Let $\langle \mathsf{gpk}, \mathsf{auth}_u \rangle$ be the common input of the procedure, and let gsk and umk_u be the GM's and the user's private inputs respectively, where auth_u may be $\langle \rho, \beta, \gamma \rangle$ (then $\gamma = \beta^{\mathsf{umk}_u} \pmod{\rho}$) or empty (then umk_u is empty). First, the user sets $x_i' = \mathsf{umk}_u$ (if umk_u is empty then chooses a random $x_i' \in \Lambda_\epsilon^k$). She computes $C_i = b^{x_i'} \pmod{n}$ and send it to the GM. Second, the user and the GM engage in a protocol for non-adaptive drawing a random power, and as a result the user gets $x_i \in_R \mathrm{M}_\epsilon^k$ and GM gets $X_i = a^{x_i} \pmod{n}$. The user encrypts x_i using sCS encryption scheme (see Section 4), i.e., $E_i = \langle U_i = \hat{g}^{\hat{r}}\ ,\ V_i = \hat{y}^{\hat{r}} \cdot \hat{h}^{x_i} \rangle \pmod{\hat{n}^2}$, where $\hat{h} = 1 + \hat{n}$

[9] The design of *Join* and *JoinOnAuth* have been merged due to space limitations.

and $\hat{r} \in_R \mathbb{Z}_{\hat{n}/4}$. Now, the user computes the following signatures of knowledge that guarantee that C_i and E_i are well formed[10].

$$E_i^{\wp} = \mathrm{SK}\{(x', r, x) : C_i = b^{x'} \pmod{n};\ \gamma = \beta^{x'} \pmod{\rho};\ X_i = a^x \pmod{n};$$
$$U_i = \hat{g}^r \pmod{\hat{n}^2};\ V_i = \hat{y}^r \hat{h}^x \pmod{\hat{n}^2}\}(\mathsf{auth}_u, C_i, X_i, U_i, V_i).$$

The GM, having received $E_i^{\wp}$ from the user, verifies $E_i^{\wp}$. Then GM selects a random prime $e_i \in \Gamma_{\epsilon}^{k}$, computes $A_i = (C_i X_i a_0)^{e_i^{-1}} \pmod{n}$, sends $\langle A_i, e_i \rangle$ to the user. Let $\mathsf{jlog}_i = \langle A_i, e_i, C_i, X_i, U_i, V_i, E_i^{\wp}, \mathsf{auth}_u \rangle$ and $\mathsf{usk}_i = \langle A_i, e_i, x_i, x_i' \rangle$ be the GM's and User's private outputs respectively.

• **Sign**. Played by a member of the group to issue signatures. Let $\langle m, \mathsf{gpk}, \mathsf{usk}_i \rangle$ be the input of the procedure, then it computes

$$T_1 = A_i y^r,\ T_2 = g^r,\ T_3 = g^{e_i} h^r,\ T_4 = g^{x_i k},\ T_5 = g^k,\ T_6 = g^{x_i' k'},\ T_7 = g^{k'}\ .$$

where $r, k, k' \in_R$ M, and then computes the following signature of knowledge:

$$\sigma^{\wp} = \mathrm{SK}\{(x, x', e, r, h') :\ T_2 = g^r\ ;\ T_3 = g^e h^r\ ;\ T_2^e = g^{h'}\ ;\ T_5^x = T_4\ ;$$
$$T_7^{x'} = T_6\ ;\ a_0 a^x b^{x'} y^{h'} = T_1^e\ \}(m)\ .$$

Let $\sigma = \langle T_1, \cdots, T_7, \sigma^{\wp} \rangle$ be the public output of the procedure.

• **Verify**. Played by any entity that wants to verify a signature. Let $\langle m, \mathsf{gpk}, \sigma \rangle$ be the input of the procedure, then it verifies if $\sigma^{\wp}$ specified in the *Sign* procedure holds.

• **Open**. It is an interactive procedure composed of the following procedures: *OpenSigDShare*, *OpenSignature*, *OpenRefCheck*.

• **OpenSigDShare**. Played by j-th fairness authority $\forall\ j \in \{1, \cdots, \zeta\}$ to decrypt a share of the member reference from the signature. Let $\langle \sigma, \mathsf{gpk}, \mathsf{fgpk}_j, \mathsf{fgsk}_j \rangle$ be the input of the procedure, then computes $\hat{\omega}_{j\sigma} = T_2^{o_j} \pmod{n}$ and a signature of knowledge that the share is correct:

$$\hat{\omega}_{j\sigma}^{\wp} = \mathrm{SK}\{(o) :\ y_j = g^o \pmod{n}\ ;\ \hat{\omega}_{j\sigma} = T_2^o \pmod{n}\}(\sigma)\ .$$

Let $\langle \hat{\omega}_{j\sigma}, \hat{\omega}_{j\sigma}^{\wp} \rangle$ be the public output of the procedure.

• **OpenSignature**. Played by Judge, combines the shares to compute a member reference. Let $\langle \sigma, \mathsf{gpk}, \{\mathsf{fgpk}_j, \hat{\omega}_{j\sigma}, \hat{\omega}_{j\sigma}^{\wp}\}_{j=1}^{\zeta} \rangle$ be the input of the procedure, then it verifies if $\{\hat{\omega}_{j\sigma}^{\wp}\}_{j=1}^{\zeta}$ specified in the *OpenSigDShare* procedure holds. and computes $\omega_\sigma = T_1 / (\prod_{j=1}^{\zeta} \hat{\omega}_{j\sigma}) \pmod{n}$. Let ω_σ be the public output of the procedure.

• **OpenRefCheck**. Played by Judge or the GM to check the matching of the member reference with a given join transcript. Let $\langle \omega_\sigma, \mathsf{jlog}_i \rangle$ be the input of the procedure, then it verifies the jlog_i integrity, by means of the *VerifyJoinLog* procedure (described later), and checks if ω_σ equals A_i from jlog_i.

• **Reveal**. It is an interactive procedure composed of the following procedures: *RevealDShare* and *RevealTKey*.

• **RevealDShare**. Played by j-th fairness authority $\forall\ j \in \{1, \cdots, \zeta\}$ to decrypt a share of the member tracing key from the join transcript. Let $\langle \mathsf{jlog}_i, \mathsf{gpk}, \mathsf{fpk}_j, \mathsf{fsk}_j \rangle$ be the input of the procedure, then it verifies if $E_i^{\wp}$ specified in the *JoinOnAuth* procedure holds, and computes $\hat{\tau}_{ji} = U_i^{\hat{o}_j} \pmod{\hat{n}^2}$ and a signature of knowledge that the share is correct:

$$\hat{\tau}_{ji}^{\wp} = \mathrm{SK}\{(o) :\ \hat{y}_j = \hat{g}^o \pmod{\hat{n}^2}\ ;\ \hat{\tau}_{ji} = U_i^o \pmod{\hat{n}^2}\}(\mathsf{jlog}_i)\ .$$

Let $\langle \hat{\tau}_{ji}, \hat{\tau}_{ji}^{\wp} \rangle$ be the public output of the procedure.

• **RevealTKey**. Played by Judge, combines the shares to compute a member tracing key. Let $\langle \mathsf{jlog}_i, \mathsf{gpk}, \{\mathsf{fpk}_j, \hat{\tau}_{ji}, \hat{\tau}_{ji}^{\wp}\}_{j=1}^{\zeta} \rangle$ be the input of the procedure, then it verifies the jlog_i integrity by means of the *VerifyJoinLog* procedure, and if $\{\hat{\tau}_{ji}^{\wp}\}_{j=1}^{\zeta}$ specified in the *RevealDShare* procedure

[10] If auth_u is empty, the part $\gamma = \beta^{x'} \pmod{\rho}$ is ignored.

hold, then computes $\hat{x}_i = (V_i / (\prod_{j=1}^{\zeta} \hat{\tau}_{ji}))^{2t} \pmod{\hat{n}^2}$ with $t = 2^{-1} \pmod{\hat{n}}$, and $\tau_i = (\hat{x}_i - 1)/\hat{n}$. Let τ_i be the public output of the procedure.

• **Trace**. Played by the Tracing Agents to identify if the member tracing key matches a signature. Let $\langle \mathsf{gpk}, \tau_i, \sigma \rangle$ be the input of the procedure, then checks if T_4 equals $T_5^{\tau_i} \pmod{n}$.

• **Claim**. Played by a member of the group to prove that issued the signature. Let $\langle \mathsf{gpk}, \sigma, \gamma, \mathsf{usk} \rangle$ be the input of the procedure, where γ is a challenge string, then it computes a signature of knowledge:

$$\pi^{\wp} = \mathrm{SK}\{(x') : \; T_6 = T_7^{x'} \pmod{n}\}(\sigma, \gamma) \; .$$

Let $\pi^{\wp}$ be the public output of the procedure.

• **VerifyClaim**. Played by any entity that wants to verify a claim. Let $\langle \mathsf{gpk}, \sigma, \gamma, \pi^{\wp} \rangle$ be the input of the procedure, then it verifies if $\pi^{\wp}$ specified in the *Claim* procedure holds.

• **ClaimLink**. Played by a member of both groups to create a link between two signatures. Let $\langle \mathsf{gpk}_1, \sigma_1, \mathsf{gpk}_2, \sigma_2, \gamma, \mathsf{usk}_1, \mathsf{usk}_2 \rangle$ be the input of the procedure, such that $\mathtt{mkey}(\mathsf{usk}_1) = \mathtt{mkey}(\mathsf{usk}_2)$ and γ is a challenge string, then it computes a signature of knowledge:

$$\lambda^{\wp} = \mathrm{SK}\{(x') : \; T_{6\sigma_1} = T_{7\sigma_1}^{x'} \pmod{n_{\sigma_1}} \; ; T_{6\sigma_2} = T_{7\sigma_2}^{x'} \pmod{n_{\sigma_2}}\}(\sigma_1, \sigma_2, \gamma) \; .$$

Let $\lambda^{\wp}$ be the public output of the procedure.

• **VerifyLink**. Played by any entity that wants to verify a link between two signatures. Let $\langle \mathsf{gpk}_1, \sigma_1, \mathsf{gpk}_2, \sigma_2, \gamma, \lambda^{\wp} \rangle$ be the input of the procedure, then it verifies if $\lambda^{\wp}$ specified in the *Claim-Link* procedure holds.

• **VerifyJoinLog**. It checks that the integrity of the join transcript holds. Let $\langle \mathsf{gpk}, \mathsf{jlog}_i \rangle$ be the input of the procedure then it verifies if $A_i^{e_i}$ equals $a_0 X_i C_i \pmod{n}$ and if $E_i^{\wp}$ holds. Note that $E_i^{\wp}$ is a user's non-repudiable proof that binds $\langle \mathsf{auth}_u, C_i, X_i, E_i \rangle$.

Note 1. Note that the order of $QR(\hat{n})$ must be unknown because the security of the verifiable encryption scheme is based on the Decision Composite Residuosity assumption, which does not hold if the factorization of $\hat{n}$ is known.

Note also that h is computed by the fairness authorities because if $\mathsf{dlog}_g h$ is known by any party, then such party would be able to open and trace the signatures for this group.

Note 2. The *JoinOnAuth* procedure accepts both: (i) a string that identifies the user, in this case auth_u relates the user's public key, which in case of a DSA public key would be $\langle \rho, \beta, \gamma \rangle$, such that $\gamma = \beta^{\alpha} \pmod{\rho}$; and (ii) a string that anonymously authenticates the user, such as a FTMG–signature, and then auth_u takes the values $\langle n, T_7, T_6 \rangle$ from the signature.

In any case, the user master key is the private key (α) that corresponds with the user's public key ($\alpha = \mathrm{dlog}_{\beta}\gamma$ and $\alpha = \mathrm{dlog}_{T_7} T_6$ respectively), and remains unaltered even if a user joins a group, and then uses this group for being authenticated to join another group, an so on successively. Note that if a signature is opened or traced, the non-repudiable binding with the user holds even through multiple nested anonymous joins.

If the authentication string in *JoinOnAuth* is used in the aforementioned way, then different users have different master keys, and therefore it is not possible to link signatures issued by different users.

6 Performance Analysis

This section analyzes the performance of the proposed scheme and compares it with related works, considering the features provided by each one.

Table 1. Performance Analysis

	ACJT00	CL01	FTMGS
Member-Size	1280	608	1488
Sign-Size	656	1728	1312
Sign-Exp	12	28	21
Vrfy-Exp	11	30	21

Table 2. Summary of Features

	ACJT00	CL01	FTMGS
Anonymous	$+$	$+$	$+$
Unlinkable	$+$	$-^{(\star)}$	$+$
Reversible	$+$	$+$	$+$
Traceable	$-$	$-$	$+$
Revocable	$-$	$-^{(\ddagger)}$	$+$
MultiGroup	$-$	$+^{(\star)}$	$+$
DeterSharing	$-$	$+$	$+$
Fairness	$-$	$+$	$+$
Non-Repudiation	$+^{(\dagger)}$	$+$	$+$

Table 1 shows the performance for the proposed scheme (FTMGS) and compares it with the state of the art in group signatures (ACJT00 [2]), and a anonymous credential systems (CL01 [7]). In this analysis, joining to a group and sign/verify[11] in both ACJT00 and FTMGS are compared with credential issuance and showing a credential under a pseudonym with revocation in CL01 respectively.

In this table, the *member-size* row refers to the size[12] of data (in bytes) the group manager (organization) has to keep for each member of the group (credential issued). The *sign-size* row shows the length (in bytes) of a signature (credential show). Moreover, the *sign-exp* and *vrfy-exp* rows show the number of exponentiations required to generate and verify a signature (credential show).

Additionally, Table 2 shows a summary of the main features that the proposed scheme (FT-MGS) exhibits, and compares it with the above schemes. In this case, ACJT00$^{(\dagger)}$ assumes that during the join phase, the user signs some binding term. Also, CL01$^{(\ddagger)}$ calls revocation to what we call reversibility, and by revocability we means the ability to remove a member from the group, or in the CL01 case, the ability to make sure that a given user can not succeed in showing a credential if the given credential has been revoked (without breaking the anonymity of non-revoked users). Additionally, when a user shows several credentials to an organization in CL01$^{(\star)}$, she guarantees that the credentials belong to the same person by exposing the pseudonym under which the organization knows that user. In this case the scheme exhibits multi-group features, but then protocols showing credentials are linkable. Otherwise, if the pseudonym is not exposed, then the protocols showing credentials are unlinkable, but then they do not enjoy the multi-group feature.

Finally, both ACJT00 and FTMGS can be incorporated into standard frameworks [5] to provide support, with very interesting features, for anonymous authentication and authorization inside standard infrastructures.

7 Conclusion

We have presented a new cryptographic primitive in the scope of group signatures. This primitive combines features from traceable signatures and multi-group signatures. It also incorporates a mechanism to dissuade users from sharing their private keys, and introduces a threshold scheme to

[11] In FTMGS, the overhead of linking signatures is included in the signature analysis.

[12] In the measures, the elements of $QR(n)$, the free variable witnesses, and the hashed challenges are 1024, 512 and 128 bits long respectively.

guarantee fairness in the process of opening or tracing signatures. All this combination of features makes the scheme very suited to support anonymity in real world scenarios.

References

1. R. Aditya, K. Peng, C. Boyd, E. Dawson, and B. Lee. Batch verification for equality of discrete logarithms and threshold decryptions. In *ACNS*, pages 494–508, 2004.
2. G. Ateniese, J. Camenish, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, pages 255–270, 2000.
3. G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Financial Cryptography*, pages 196–211, 1999.
4. N. Bari and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT*, pages 480–494, 1997.
5. V. Benjumea, S. G. Choi, J. Lopez, and M. Yung. Anonymity 2.0: X.509 extensions supporting privacy-friendly authentication. In *CANS*, pages 265–281, 2007.
6. F. Brandt. Efficient cryptographic protocol design based on distributed ElGamal encryption. In *ICISC*, pages 32–47, 2005.
7. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001.
8. J. Camenish and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, pages 126–144, 2003.
9. J. Camenish and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO*, pages 410–424, 1997.
10. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT*, pages 90–106, 1999.
11. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.
12. C. Dwork, J. B. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information (preliminary version). In *STOC*, pages 489–498, 1996.
13. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1985.
14. P. Fouque and J. Stern. Fully distributed threshold RSA under standard assumptions. In *ASIACRYPT*, 2001.
15. P.-A. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT*, pages 351–368, 2001.
16. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT*, pages 295–310, 1999.
17. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure applications of pedersen's distributed key generation protocol. In *CT-RSA*, pages 373–390, 2003.
18. O. Goldreich, B. Pfitsmann, and R. L. Rivest. Self-delegation with controlled propagation - or - what if you lose your laptop. In *CRYPTO*, pages 153–168, 1998.
19. Identity 2.0. `http://www.identity20.com/`.
20. M. Jakobsson, A. Juels, and P. Q. Nguyen. Proprietary certificates. In *CT-RSA*, pages 164–181, 2002.
21. S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. In *EUROCRYPT*, pages 97–114, 2007.
22. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT*, pages 571–589, 2004. Full Version: `http://eprint.iacr.org/2004/007`.
23. A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/`.
24. A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199, 1999.
25. L. Nguyen and R. Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *ASIACRYPT*, pages 372–386, 2004.
26. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
27. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
28. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *EUROCRYPT*, pages 1–16, 1998.

A Security of Our Scheme

A.1 Correctness of Fair Traceable Multi-Group Signature Schemes.

Definition 8. *A fair traceable multi-group signature scheme with security parameter ν is* **correct** *if the following six conditions are satisfied (with overwhelming probability in ν). See Fig 2 for abbreviations.*

1. (Join-Correctness) *For any v, $\mathrm{U_i} \notin G^v$, and $\langle \mathsf{usk}_i^v, \mathsf{jlog}_i^v \rangle \leftarrow \textsc{Join}^v/\textsc{JoinOnAuth}^v\ \ (\mathsf{auth}_i)$, it holds that $\mathtt{mkey}(\mathsf{usk}_i^v) = \mathtt{mkey}(\mathsf{auth}_i)$, $\mathsf{tkey}(\mathsf{usk}_i^v) = \mathsf{tkey}(\mathsf{jlog}_i^v)$ and $\mathsf{mref}(\mathsf{usk}_i^v) = \mathsf{mref}(\mathsf{jlog}_i^v)$.*
2. (Sign-Correctness) *For any v, $\mathrm{U_i} \in G^v$, and m, it holds that* $\textsc{Verify}^v(m, \textsc{Sign}_i^v(m)) = \mathtt{acc}$.
3. (Open-Correctness) *For any v, $\mathrm{U_i} \in G^v$, and m, it holds that* $\textsc{Open}^v(\textsc{Sign}_i^v(m)) = \mathsf{mref}(\mathsf{jlog}_i^v)$.
4. (Trace-Correctness) *For any v, $\mathrm{U_i} \in G^v$, and m, it holds that* $\textsc{Trace}^v(\textsc{Sign}_i^v(m), \textsc{Reveal}^v(\mathsf{jlog}_i^v)) = \mathtt{acc}$*; on the other hand, for any v, $\mathrm{U}_j \neq \mathrm{U_i}$, and m, it holds that* $\textsc{Trace}^v(\textsc{Sign}_j^v(m), \textsc{Reveal}^v(\mathsf{jlog}_i^v)) = \perp$.
5. (Claim-Correctness) *For any v, $\mathrm{U_i} \in G^v$, m, and γ, it holds that* $\textsc{VerifyClaim}^v(\textsc{Claim}_i^v(\textsc{Sign}_i^v(m), \gamma)) = \mathtt{acc}$.
6. (ClaimLink-Correctness) *For any v_1, v_2, $\mathrm{U_i} \in G^{v_1} \cap G^{v_2}$, m_1, m_2, and γ where $\mathtt{mkey}(\mathsf{usk}_i^{v_1}) = \mathtt{mkey}(\mathsf{usk}_i^{v_2})$, it holds that* $\textsc{VerifyLink}^{v_1,v_2}(\textsc{ClaimLink}_i^{v_1,v_2}(\textsc{Sign}_i^{v_1}(m_1), \textsc{Sign}_i^{v_2}(m_2), \gamma)) = \mathtt{acc}$.

A.2 Security Model for Fair Traceable Multi-Group Signature Schemes

Now we define the security of fair traceable multi-group signature schemes. The security definitions will be formulated via experiments in which an adversary's attack capabilities are modelled by providing it with access to certain oracles. Below is the list of oracles provided in our model. See Figure 2 for the bookkeeping variables shared by the oracles.

- $\mathsf{Q}_{\mathsf{p-gengroup}}()$. This oracle runs $\textsc{Setup}$ algorithm, sets $\mathsf{N}_g \leftarrow \mathsf{N}_g + 1$ and $\mathsf{ginfo}^{\mathsf{N}_g} \leftarrow \langle \mathsf{gpk}^{\mathsf{N}_g}, \mathsf{gsk}^{\mathsf{N}_g}, \{\mathsf{fgsk}_j^{\mathsf{N}_g}\}_{j=1}^{\zeta} \rangle$, adds N_g into G_p and returns $\langle \mathsf{N}_g, \mathsf{gpk}^{\mathsf{N}_g} \rangle$.
- $\mathsf{Q}_{\mathsf{b-gengroup}}()$. The group generated by this oracle allows the adversary to control its group manager. This oracle runs $\textsc{Setup}$ algorithm, sets $\mathsf{N}_g \leftarrow \mathsf{N}_g + 1$ and $\mathsf{ginfo}^{\mathsf{N}_g} \leftarrow \langle \mathsf{gpk}^{\mathsf{N}_g}, \mathsf{gsk}^{\mathsf{N}_g}, \{\mathsf{fgsk}_j^{\mathsf{N}_g}\}_{j=1}^{\zeta} \rangle$, adds N_g into G_b and returns $\langle \mathsf{N}_g, \mathsf{gpk}^{\mathsf{N}_g}, \mathsf{gsk}^{\mathsf{N}_g} \rangle$.
- $\mathsf{Q}_{\mathsf{stat}}()$. It returns N_g and $(\mathsf{N}_u^1, \ldots, \mathsf{N}_u^{\mathsf{N}_g})$.
- $\mathsf{Q}_{\mathsf{fa-share}}(v, j)$. This oracle leaks a fairness authority's secret key. Specifically, if $|\mathsf{F}^v| < \zeta - 1$ and $j \leq \zeta$ then it adds j into F^v and returns $\langle \mathsf{fgsk}_j^v \rangle$ from ginfo^v; otherwise it returns $\perp$.
- $\mathsf{Q}_{\mathsf{p-join}}(v)$. This oracle simulates the $\textsc{Join}$ protocol in private. Let $i = \mathsf{N}_u^v + 1$. If $v \in \mathsf{G}_p$ then it sets $\mathsf{N}_u^v \leftarrow i$, adds $\langle i, \mathsf{jlog}_i^v, \mathsf{usk}_i^v \rangle$ into uinfo^v, adds i into U_p^v, and returns i; otherwise it returns $\perp$.
- $\mathsf{Q}_{\mathsf{p-authjoin}}(v_1, v_2)$. If $v_1 \notin \mathsf{G}_p$ or $v_2 \notin \mathsf{G}_p$ then it returns $\perp$ and terminates. If $\mathsf{U}_p^{v_1} = \emptyset$, then it executes $\mathsf{Q}_{\mathsf{p-join}}(v_1)$. Now, it firstly chooses $i_1 \in_R \mathsf{U}_p^{v_1}$, and computes $\mathsf{auth}_{i_1} \leftarrow \textsc{Sign}_{i_1}^{v_1}(m)$ for a random message m,[13] and simulates the $\textsc{JoinOnAuth}$ protocol on G^{v_2} in private with $\mathtt{mkey}(\mathsf{usk}_{i_1}^{v_1})$ on the user's side. The rest is similar to $\mathsf{Q}_{\mathsf{p-join}}$'s bookkeeping of variables (here, bookkeeping is done for G^{v_2}). It returns i_2.
- $\mathsf{Q}_{\mathsf{a-join}}(v)$. This oracle allows the adversary to introduce an adversarially controlled user to the system. If $v \notin \mathsf{G}_p$ then returns $\perp$ and terminates. The oracle, taking the role of GM^v, executes $\textsc{Join}$ with the adversary who takes the role of a joining user. Let i be the value of $\mathsf{N}_u^v + 1$. When the protocol terminates successfully, it sets $\mathsf{N}_u^v \leftarrow i$, $\mathsf{uinfo}_i^v \leftarrow \langle i, \mathsf{jlog}_i^v, \perp \rangle$, and adds i into U_a^v. The adversary will obtain $\langle i, \mathsf{usk}_i^v \rangle$.

[13] The authentication string can also be a digital signature by generalizing that G^0 is PKI. The combination of traceable signatures and digital signatures with link or proof of same master secret key can also be used. However, in the modeling we choose a single traceable signature for the purpose of clear exposition.

Abbreviations.

$\textsc{Sign}_i^v, \textsc{Claim}_i^v, \textsc{ClaimLink}^{v,v'}$: signing, claiming, and link-claim algorithm using usk_i^v w.r.t. G^v and $G^{v'}$
$\textsc{Open}^v, \textsc{Reveal}^v$: opening and revealing procedure using gsk_i^v and fsk_i^v w.r.t. G^v
$\textsc{Join}^v, \textsc{JoinOnAuth}^v$: joining procedures w.r.t. G^v
$\textsc{Verify}^v, \textsc{VerifyClaim}^v, \textsc{VerifyLink}^{v,v'}$: verification algorithms for signatures, claims, and link-claims w.r.t. G^v and $G^{v'}$

Bookkeeping variables.

$\mathsf{N}_g, \mathsf{N}_u^v$: the number of existing groups and users in G^v respectively.
$\mathsf{ginfo}^v, \mathsf{uinfo}^v$: $\langle v, \mathsf{gpk}^v, \mathsf{gsk}^v, \{\mathsf{fgsk}_j^v\}_{j=1}^{\zeta}\rangle$, and $\{\langle i, \mathsf{usk}_i^v, \mathsf{jlog}_i^v\rangle\}_{\mathrm{U_i} \in G^v}$ respectively.
sigs^v: signatures by $\mathsf{Q}_{\mathsf{sig}}$
$\mathsf{G}_p, \mathsf{G}_b$: groups created by $\mathsf{Q}_{\mathsf{p-gengroup}}$ and $\mathsf{Q}_{\mathsf{b-gengroup}}$ respectively.
F^v: fsk_j^v's leaked by $\mathsf{Q}_{\mathsf{fa-share}}$
U_r^v: users revealed by $\mathsf{Q}_{\mathsf{reveal}}$
U_p^v: users in G^v from $\mathsf{Q}_{\mathsf{p-join}}, \mathsf{Q}_{\mathsf{p-authjoin}}$
U_a^v: users in G^v from $\mathsf{Q}_{\mathsf{a-join}}, \mathsf{Q}_{\mathsf{a-authjoin}}$
U_b^v: users in G^v from $\mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}$

Fig. 2. Abbreviations and Bookkeeping variables

– $\mathsf{Q}_{\mathsf{a-authjoin}}(v_1, v_2, i_1, \mathsf{auth}_{i_1})$. If $v_1 \notin \mathsf{G}_p$ or $v_2 \notin \mathsf{G}_p$ or $i_1 \notin \mathsf{U}_a^{v'}$ then returns $\perp$ and terminates. The oracle, taking the role of GM^{v_2}, executes $\textsc{JoinOnAuth}$ with the adversary playing as a joining user with $\mathtt{mkey}(\mathsf{usk}_{i1}^{v_1})$. The rest is similar to $\mathsf{Q}_{\mathsf{a-join}}$'s bookkeeping of variables (here, bookkeeping is done for G^{v_2}). The adversary will obtain $\langle i_2, \mathsf{usk}_{i_2}^{v_2}\rangle$.
– $\mathsf{Q}_{\mathsf{t-join}}(v, x'), \mathsf{Q}_{\mathsf{t-authjoin}}(v_1, v_2, i_1, x')$. As in [22], we introduce these oracles for technical reasons. The oracles which are identical to $\mathsf{Q}_{\mathsf{p-join}}$ and $\mathsf{Q}_{\mathsf{p-authjoin}}$ except that x' is chosen by the adversary, i is added to U_a^v, and they return $\langle i, \mathsf{usk}_i^v\rangle$ at the end.
– $\mathsf{Q}_{\mathsf{b-join}}(v)$. If $v \notin \mathsf{G}_b$ then returns $\perp$ and terminates. The oracle, taking the role of a joining user, executes $\textsc{Join}$ with the adversary playing as GM^v. Let i be the value of $\mathsf{N}_u^v + 1$. When the protocol terminates successfully, it sets $\mathsf{N}_u^v \leftarrow i$, $\mathsf{uinfo}_i^v \leftarrow \langle i, \perp, \mathsf{usk}_i^v\rangle$, and adds i into U_b^v. The adversary will obtain (i, jlog_i^v).
– $\mathsf{Q}_{\mathsf{b-authjoin}}(v_1, v_2)$. If $v_1 \notin \mathsf{G}_b$ or $v_2 \notin \mathsf{G}_b$ or $G^{v_1} = \emptyset$ then it returns $\perp$ and terminates. Now, it firstly chooses $i_1 \in_R \mathsf{U}_b^{v_1}$, and computes $\mathsf{auth}_{i_1} \leftarrow \textsc{Sign}_{i_1}^{v_1}(m)$ for a random message m, and executes $\textsc{JoinOnAuth}$, using $\mathtt{mkey}(\mathsf{usk}_{i_1}^{v_1})$ as a joining user, with the adversary playing as GM^{v_2}. The rest is similar to $\mathsf{Q}_{\mathsf{b-join}}$'s bookkeeping of variables (here, bookkeeping is done for G^{v_2}). The adversary will obtain $(i_2, \mathsf{jlog}_{i_2}^{v_2})$.
– $\mathsf{Q}_{\mathsf{sig}}(v, i, m)$. If $i \notin \mathsf{U}_p^v \cup \mathsf{U}_b^v$, then it returns $\perp$ and terminates. Otherwise, it generates a signature σ using usk_i^v, adds $\langle i, \sigma\rangle$ into sigs^v, and returns σ.
– $\mathsf{Q}_{\mathsf{reveal}}(v, i)$. This oracle reveals the tracing key τ for member $i \in G_v$. If $i \in \mathsf{U}_p^v \cup \mathsf{U}_b^v$, then it adds i into U_r^v, and returns $\mathsf{tkey}(\mathsf{jlog}_i^v)$; otherwise it returns $\perp$.

Now we define various experiments to capture the security we need. See Figure 3 and 4 for the description of the experiments. In misidentification attack ($\mathbf{Exp}_{\mathcal{A}}^{\mathsf{mis}}$), the adversary tries to produce a signature that does not open or trace to any of the adversarially controlled users. In framing attack ($\mathbf{Exp}_{\mathcal{A}}^{\mathsf{fra}}$), the adversary is allowed to act as a group manager, and tries to generate a signature, a claim, or a link-claim that traces to an honest user. When an authentication string is used during the joined procedure, we require stronger non-framability; the adversary is allowed to control the join database. In anonymity attack ($\mathbf{Exp}_{\mathcal{A}}^{\mathsf{anon}}$), the adversary is allowed to act as a group manager, and tries to break the anonymity of signatures. Note that *anonymity here is stronger than that in the original traceable signature* in the sense that even the group manager does

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{mis}}(1^\nu)$
  $\langle \mathsf{st}_1, v\rangle \leftarrow \mathcal{A}(\,:\, \mathsf{Q}_{\mathsf{p-gengroup}});$
  If $v \notin \mathsf{G}_p$, then return 0;
  $\langle m, \sigma\rangle \leftarrow \mathcal{A}(\mathsf{st}_1 : \mathsf{Q}_{\mathsf{a-join}}, \mathsf{Q}_{\mathsf{a-authjoin}}, \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{p-join}}, \mathsf{Q}_{\mathsf{p-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}});$
  If $\sigma \in \mathsf{sigs}^v$ or $\textsc{Verify}^v(m,\sigma) = \perp$, then return 0;
  If $\forall\, i \in \mathsf{U}_a^v$, $\textsc{Open}^v(\sigma) \neq \mathsf{mref}(\mathsf{jlog}_i^v)$, then return 1;
  If $\forall\, i \in \mathsf{U}_a^v$, $\textsc{Trace}^v(\sigma, \mathsf{tkey}(\mathsf{jlog}_i^v)) = \perp$, then return 1;
  return 0;
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{fra}}(1^\nu)$
  $\langle \mathsf{st}_1, v, v'\rangle \leftarrow \mathcal{A}(\,:\, \mathsf{Q}_{\mathsf{b-gengroup}});$
  $\langle m, \sigma, \gamma, \pi, m'\sigma', \lambda\rangle \leftarrow \mathcal{A}(\mathsf{st}_1 : \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}});$
  If $\textsc{Verify}^v(m,\sigma) = \perp$, then return 0;
  If $\sigma \notin \mathsf{sigs}^v$ and $\exists i \in \mathsf{U}_b^v : \textsc{Open}^v(\sigma) = \mathsf{mref}(\mathsf{usk}_i^v)$, then return 1;
  If $\sigma \notin \mathsf{sigs}^v$ and $\exists i \in \mathsf{U}_b^v \setminus \mathsf{U}_r^v$: $\textsc{Trace}^v(\textsc{Reveal}^v(i), \sigma) = \mathtt{acc}$, then return 1;
  If $\sigma \in \mathsf{sigs}^v$ and $\textsc{VerifyClaim}^v(\sigma, \gamma, \pi) = \mathtt{acc}$, then return 1;
  If $\sigma \in \mathsf{sigs}^v$ and $\textsc{Verify}^{v'}(m', \sigma') = \mathtt{acc}$ and $\textsc{VerifyLink}^{v,v'}(\sigma, \sigma', \gamma, \lambda) = \mathtt{acc}$,
     then return 1;
  return 0;
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{anon}}(1^\nu)$
  $\langle \mathsf{st}_1, v, i_0, i_1, m\rangle \leftarrow \mathcal{A}(\,:\, \mathsf{Q}_{\mathsf{b-gengroup}}, \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}});$
  If $i_0 \notin \mathsf{U}_b^v$ or $i_1 \notin \mathsf{U}_b^v$, then return $e$ where $e \xleftarrow{R} \{0,1\}$;
  $\delta \xleftarrow{R} \{0,1\}$; $\sigma \leftarrow \textsc{Sign}_{i_b}^v(m)$;
  $\delta^* \leftarrow \mathcal{A}(\mathsf{st}_1, \sigma : \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}});$
  If $\{i_0, i_1\} \cap \mathsf{U}_r^v \neq \emptyset$, then return $\xi$ where $\xi \xleftarrow{R} \{0,1\}$;
  If $\delta = \delta^*$ then return 1;
  return 0;
```

Fig. 3. Experiments in the original traceable signatures

not know which member signed. In link-forgery attack ($\mathbf{Exp}_{\mathcal{A}}^{\mathsf{link}}$), the adversary tries to forge a false link. In join-anonymity attack ($\mathbf{Exp}_{\mathcal{A}}^{\mathsf{j-anon}}$), the adversary is allowed to act as a group manager, and tries to track a member's joining situation. In addition to experiments, we define another security aspect: trusted-fairness, which means fairness authorities function as a trusted party collectively.

Definition 9. *A fair traceable multi-group signature scheme is said to be* **secure** *if the following conditions are satisfied:*

1. *For any PPT algorithm $\mathcal{A}$, $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{mis}}(1^\nu) = 1] = neg(\nu)$.*
2. *For any PPT algorithm $\mathcal{A}$, $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{fra}}(1^\nu) = 1] = neg(\nu)$.*
3. *For honest users who joined with an authentication string, the scheme remains secure against framing attack even the GM maliciously manages the* jlog *database.*
4. *For any PPT algorithm $\mathcal{A}$, $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{anon}}(1^\nu) = 1] - 1/2| = neg(\nu)$.*
5. *For any PPT algorithm $\mathcal{A}$, $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{j-anon}}(1^\nu) = 1] - 1/2| = neg(\nu)$.*[14]
6. *For any PPT algorithm $\mathcal{A}$, $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{link}}(1^\nu) = 1] = neg(\nu)$.*
7. *The above statements remain true even when $Q_{fa-share}$ is allowed to $\mathcal{A}$.*

[14] Our scheme guarantees that joining protocols are unlinkable among them and reveals nothing about the joining user's master secret key. However, a collusion of GMs may be able to establish an external connection among them if the authentication string is not anonymous. This fact does not affect at all to the anonymity of the scheme.

```
Experiment $\mathbf{Exp}^{\mathsf{j-anon}}_{\mathcal{A}}(1^\nu)$
  $\langle \mathsf{st}_1, v, v', i_0, i_1\rangle \leftarrow \mathcal{A}(\,:\mathsf{Q}_{\mathsf{b-gengroup}}, \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
  If $i_0$ or $i_1$ is not in $\mathsf{U}^v_b$ or has already joined $G^{v'}$, then return $\xi$ where $\xi \xleftarrow{R} \{0,1\}$;
  $\delta \xleftarrow{R} \{0,1\}$;
  Execute $\textsc{JoinOnAuth}^{v'}$ with $\mathcal{A}(\mathsf{st}_1)$ using $\mathtt{mkey}(\mathsf{usk}^v_{i_b})$, and thereafter $\mathcal{A}$ returns $\mathsf{st}_2$;
  $\delta^* \leftarrow \mathcal{A}(\mathsf{st}_2 : \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
  If $\{i_0, i_1\} \cap \mathsf{U}^v_r \neq \emptyset$, then return $\xi$ where $\xi \xleftarrow{R} \{0,1\}$;
  If $\delta = \delta^*$ then return 1;
  return 0;
Experiment $\mathbf{Exp}^{\mathsf{link}}_{\mathcal{A}}(1^\nu)$
  $\langle \mathsf{st}_1, v, v', m, m', \gamma\rangle \leftarrow \mathcal{A}(\,:\mathsf{Q}_{\mathsf{b-gengroup}}, \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
  $i \xleftarrow{R} \mathsf{U}^v_b$, $i' \xleftarrow{R} \mathsf{U}^{v'}_b$ s.t. $\mathtt{mkey}(\mathsf{usk}^v_i) \neq \mathtt{mkey}(\mathsf{usk}^{v'}_{i'})$;
  $\sigma \leftarrow \textsc{Sign}^v_i(m), \sigma' \leftarrow \textsc{Sign}^{v'}_{i'}(m')$;
  $\lambda \leftarrow \mathcal{A}(\mathsf{st}_1, \sigma, \sigma', \gamma, \mathsf{usk}^v_i, \mathsf{usk}^{v'}_{i'} : \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
  If $\textsc{VerifyLink}^{v,v'}(\sigma, \sigma', \gamma, \lambda) = \mathtt{acc}$, then return 1;
  return 0;
```

Fig. 4. New experiments to handle multi-group scenarios

A.3 Correctness of Our Fair Traceable Multi-Group Signature Scheme

Theorem 1. *The proposed fair traceable multi-group signature scheme is correct.*

From now on, we prove Theorem 1.

For Join-Correctness, the first condition is satisfied because a joining user chooses x' and the GM issues an adaptive DL-representation for the x', The other two conditions are also satisfied by observing that $\mathsf{usk}_i = \langle A, e, x, x'\rangle$, $\mathsf{jlog}_i = \langle A, e, E_{\mathsf{fpk}}(x), \ldots\rangle$, $\mathsf{mref}(\mathsf{usk}) = \mathsf{mref}(\mathsf{jlog}) = A$ and $\mathsf{tkey}(\mathsf{usk}) = x = D_{\mathsf{fgsk}}(E_{\mathsf{fgpk}}(x)) = \mathsf{tkey}(\mathsf{jlog})$.

Sign-Correctness follows immediately from the completeness of the zero-knowledge proof that is employed for the signing algorithm.

For Open-Correctness, note that any signature $\sigma \leftarrow \textsc{Sign}^v_{\mathrm{U}}(m)$ contains the values T_1, T_2 that constitute an ElGamal encryption over $QR(n)$ of the value A_i of the user's membership certificate $\langle A_i, e_i, x_i, x'_i\rangle$ with FAs' public key. The decryption of such value ($\textsc{OpenSignature}$) by using FAs' decryption shares ($\textsc{OpenSigKShare}$) equals the A_i value in jlog^v_i ($\textsc{OpenRefCheck}$). With overwhelming probability, user's certificate is unique for each user, thus for any other user $\mathrm{U}_{i'} \neq \mathrm{U}_i$ we have $A_{i'} \neq A_i$, and therefore the decryption of such value is not equal to the $A_{i'}$ in $\mathsf{jlog}^v_{i'}$.

For Trace-Correctness, observe that jlog^v_i contains the verifiable encryption of x_i, where $\langle A_i, e_i, x_i, x'_i\rangle$ is the membership certificate of user U_i. The decryption of such value ($\textsc{RevealTKey}$) by FAs' decryption shares ($\textsc{RevealTShare}$) yields the tracing trapdoor x_i for such a member U_i. Now observe that any $\sigma \leftarrow \textsc{Sign}^v_{\mathrm{U}_i}(m)$ contains the values T_4, T_5 that satisfy the property $T_4 = T_5^{x_i}$; thus it holds that $\textsc{Trace}^v(\sigma, x_i) = \mathtt{acc}$. On the other hand, for any other user $\mathrm{U}_{i'} \neq \mathrm{U}_i$, it holds with overwhelming probability that $x_{i'} \neq x_i$, and thus if $\sigma \leftarrow \textsc{Sign}^v_{\mathrm{U}_{i'}}(m)$, then we have that $T_4 = g^{kx_{i'}}$ and $T_5 = g^k$ for some $k \in M$, which implies $T_4 = T_5^{x_i}$. Since $x_{i'} \neq x_i$, the tracing condition ($T_4 = T_5^{x_i}$) does not hold.

For Claim-Correctness, observe that the $\textsc{Claim}^v_{\mathrm{U}}$ algorithm produces a non-interactive proof of knowledge for the discrete-logarithm of the value T_6 base T_7 for $\sigma \leftarrow \textsc{Sign}^v_{\mathrm{U}}(m)$; the correctness follows from the completeness of the PoK.

For ClaimLink-Correctness, observe that the $\textsc{ClaimLink}_{\mathrm{U}}^{v_1,v_2}$ algorithm produces a non-interactive proof of knowledge for the *unique* discrete-logarithm of the value T_6 base T_7 for both $\sigma^{v_1} \leftarrow \textsc{Sign}_{\mathrm{U}}^{v_1}(m)$ and $\sigma^{v_2} \leftarrow \textsc{Sign}_{\mathrm{U}}^{v_2}(m)$. If such a unique value exists, the correctness follows from the completeness of the PoK.

A.4 Security of Our Fair Traceable Multi-Group Signature Scheme

Theorem 2. *Under the assumptions described in Section* **??**, *our FTMGS scheme is secure in the random oracle model.*

From now on, we prove Theorem 2. The proof consists of several lemmas, from which the theorem directly follows. Hereafter, we suppress the use of inner sphere notation, and use the notation $\varLambda, M, \varGamma$ (this does not affect the argumentation of th proof).

Lemma 2. *For any PPT A, we have*

$$\Pr[\mathcal{A}(\,:\ldots,\mathsf{Q}_{\mathsf{a-join}},\mathsf{Q}_{\mathsf{a-authjoin}},\ldots) = \mathcal{A}(\,:\ldots,\mathsf{Q}_{\mathsf{t-join}},\mathsf{Q}_{\mathsf{t-authjoin}},\ldots)] = 1 - neg(\nu).$$

Proof: Consider an output $\mathsf{usk} = \langle A, e, x, x'\rangle$ after a joining protocol. The secure implementation ImpRP due to [22] for drawing a random x guarantees that the distribution of x from $\mathsf{Q}_{\mathsf{a-join*}}$ and the distribution of x from $\mathsf{Q}_{\mathsf{t-join*}}$ are computationally indistinguishable. The values e, x' are identically distributed, and A is simply $(a^x b^{x'} a_0)^{1/e}$. Therefore, the distribution of the DL-representation output by $\mathsf{Q}_{\mathsf{a-join*}}$ are computationally indistinguishable from that by $\mathsf{Q}_{\mathsf{t-join*}}$. □

Thanks to Lemma 2, we will use $\mathsf{Q}_{\mathsf{t-join*}}$ instead of $\mathsf{Q}_{\mathsf{a-join*}}$ hereafter.

Lemma 3. *Under the Strong-RSA assumption, for any PPT algorithm $\mathcal{A}$,*
$\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{mis}}(1^\nu) = 1] \leq neg(\nu)$.

Proof: First observe that Join-Correctness holds even if the joining user is malicious from the soundness of $E_i^{\wp}$ proofs.

Let $\mathcal{A}$ be an adversary such that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{mis}}(1^\nu) = 1]$ is non-negligible in ν. We, using $\mathcal{A}$, construct an algorithm $\mathcal{B}$ that solves the adaptive one-more representation problem. Denote by Omrp_v the v-th instance of one-more representation problem, i.e., $\langle n^{(v)}, a^{(v)}, b^{(v)}, a_0^{(v)}\rangle$. Then, after $\mathcal{B}$ fetches polynomially many Omrp instances, it solves one of them. $\mathcal{B}$ simulates oracles provided for $\mathcal{A}$ as follows:

- $\mathsf{Q}_{\mathsf{p-gengroup}}()$. Let $v := \mathsf{N}_g + 1$. $\mathcal{B}$ brings the v-th instance of one-more representation problem, $\langle n^{(v)}, a^{(v)}, b^{(v)}, a_0^{(v)}\rangle$. Then it sets
  $\mathsf{gpk} = \langle n^{(v)}, a^{(v)}, b^{(v)}, a_0^{(v)}, g, h, y, \hat{n}, \hat{g}, \hat{y}\rangle$, where other values are appropriately chosen as described in our scheme. Finally, $\mathcal{B}$ updates bookkeeping variables and returns (v, gpk).
- $\mathsf{Q}_{\mathsf{stat}}()$. $\mathcal{B}$ simply returns the specified booking variables.
- $\mathsf{Q}_{\mathsf{p-join}}(v), \mathsf{Q}_{\mathsf{p-joinauth}}(v_1, v_2)$. $\mathcal{B}$ chooses $A \in_R QR(n)$, a prime $e \in_R \varGamma$, $x \in_R M$, and $x' \in_R \varLambda$ (In the case of $\mathsf{Q}_{\mathsf{p-joinauth}}$, $x' = \mathsf{usk}_{i_1}^{v_1}$ for $i_1 \in_R \mathsf{U}_p^{v_1}$). Let $i := \mathsf{N}_u^v + 1$. $\mathcal{B}$ sets $\mathsf{usk}_i^v = \langle A, e, x, x'\rangle$, and jlog_i^v accordingly; i.e., $C_i = b^{x'}$, $X_i = a^x$, $E_i = (\hat{g}^{\hat{r}}, \hat{y}^{\hat{r}} \cdot h^x)$ for $\hat{r} \in_R \mathbb{Z}_{\hat{n}/4}$, and $E_i^{\wp}$ can be constructed easily. Finally, $\mathcal{B}$ updates bookkeeping variables.

– $\mathsf{Q}_{\mathsf{t-join}}(v, x'), \mathsf{Q}_{\mathsf{t-joinauth}}(v_1, v, i_1, x')$. $\mathcal{B}$ invokes Q_{rep} of an instance Omrp_{v_2}, and obtains a DL-representation $\mathsf{usk}_i^v = \langle A, e, x, x' \rangle$. $\mathcal{B}$ sets $\mathsf{usk}_i^v = \langle A, e, x, x' \rangle$, and jlog_i^v accordingly; i.e., $C_i = b^{x'}$, $X_i = a^x$, $E_i = (\hat{g}^{\hat{r}}, \hat{y}^{\hat{r}} \cdot h^x)$ for $\hat{r} \in_R \mathbb{Z}_{\hat{n}/4}$, and $E_i^{\wp}$ can be constructed easily. Finally, $\mathcal{B}$ updates bookkeeping variables.
– $\mathsf{Q}_{\mathsf{sig}}(v, i, m)$. If $i \notin \mathsf{U}_p^v$, $\mathcal{B}$ returns fail. Otherwise, $\mathcal{B}$ finds an entry usk_i^v (which may not be a DL-representation), and generates a signature using the simulator of the NIZK proof used in SIGN algorithm.
– $\mathsf{Q}_{\mathsf{reveal}}(v, i)$. $\mathcal{B}$ returns $\mathsf{tkey}(\mathsf{usk}_i^v)$ from uinfo_i^v.

Observe that $\mathcal{A}$'s view of its interaction with $\mathcal{B}$ is indistinguishable from the interaction with oracles in the security definition. Now, suppose that $\mathcal{A}$ exits with a successful output $\langle m, \sigma_1^* \rangle$. Then by the *generalized forking lemma* [22] $\mathcal{B}$ can get another valid output $\langle m, \sigma_2^* \rangle$. Note that the protocol used in SIGN is a proof of knowledge, which implies existence of an extractor for the protocol. Therefore, we can extract a witness $\langle x^*, x'^*, e^*, r^*, h'^* \rangle$ from σ_1^* and σ_2^*. The signature σ_1^* will be of the following form:

$$\langle T_1^*, \ldots, T_7^*, c, s_{x^*}, s_{x'^*}, s_{e^*}, s_{r^*}, s_{h'^*} \rangle.$$

Now we have two cases:

(i) $\forall i \in \mathsf{U}_a^v$: $\text{OPEN}^v(\sigma_1^*) \neq \mathsf{mref}(\mathsf{jlog}_i^v)$. This means that $A^* = \left(T_1/T_2^{\log_g^y}\right)$ is different from any A_i of $\mathsf{jlog}_i^v \in \mathsf{U}_a^v$, which implies $\langle A^*, e^*, x^*, x'^* \rangle$ is a DL-representation that is different from the ones generated by Q_{rep} oracle. Therefore, we solve the adaptive one-more representation problem.

(ii) $\forall i \in \mathsf{U}_a^v$: $\text{TRACE}^v(\sigma, \mathsf{tkey}(\mathsf{jlog}_i^v)) = \perp$. This means that, for any $i \in \mathsf{U}_a^v$, the value $x_i = \mathsf{tkey}(\mathsf{jlog}_i^v)$ is different from $x^* = \log_{T_5}^{T_4}$, which implies $\langle A^*, e^*, x^*, x'^* \rangle$ is a DL-representation that is different from the ones generated by Q_{rep} oracle. Therefore, we solve the adaptive one-more representation problem.

□

Lemma 4. *Under the discrete logarithm assumption and semantic security of sCS encryption scheme, for any PPT algorithm $\mathcal{A}$,* $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{fra}}(1^\nu) = 1] \leq neg(\nu)$.

Proof: Let $\mathcal{A}$ be an adversary such that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{fra}}(1^\nu) = 1]$ is non-negligible in ν.

Denote by DLP an instance of discrete logarithm problem (z, Z, n, p, q), i.e., a problem to find an δ such that $z^\delta = Z \pmod{n}$, where $z, Z \in QR(n)$ and $n = pq$ for safe primes p and q. We, using $\mathcal{A}$, construct an algorithm $\mathcal{B}$ that solves DLP problems with non-negligible probability. $\mathcal{B}$ plays one of two games at randomly. Observe that playing the two games at random covers all possible behaviors of the algorithm $\mathcal{A}$.

Game-1

Let V be the number of groups the adversary generated using $\mathsf{Q}_{\mathsf{b-gengroup}}$. $\mathcal{B}$ first chooses $j \in_R \{1, \ldots, V\}$, and simulates $\mathsf{Q}_{\mathsf{b-gengroup}}$ oracle as follows. Hereafter, we assume all the booking variables are appropriately updated.

– $\mathsf{Q}_{\mathsf{b-gengroup}}()$.
Suppose the adversary makes ℓ-th query on this oracle. If $\ell \neq j$, then $\mathcal{B}$ executes the SETUP algorithm as described in our scheme. Otherwise, $\mathcal{B}$ uses the DLP instance to construct the keys,

i.e.,
$\mathsf{gpk} = \langle n, a = z^{\alpha}, b = z, a_0 = z^{\beta}, g, h, y, \hat{n}, \hat{g}, \hat{y}\rangle$ and $\mathsf{gsk} = \langle p, q\rangle$, where $\alpha, \beta \in_R M$ and other values are chosen as described in SETUP of our scheme. $\mathcal{B}$ returns $(\ell, \mathsf{gpk}, \mathsf{gsk})$.

Now, $\mathcal{A}$ returns $\langle \mathsf{st}_1, v, v'\rangle$. If $j \neq v$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ proceeds and simulates the other oracles provided for $\mathcal{A}$ as follows.

– $\mathsf{Q}_{\mathsf{b-join}}(\ell)$.
If $\ell \neq v$, then $\mathcal{B}$, as a joining user, executes JOIN protocol with $\mathcal{A}$ as described in our scheme. Otherwise, $\mathcal{B}$ chooses $\hat{x}'_i \in_R \Lambda$ and executes JOIN protocol using $C_i = b^{\hat{x}'_i} \cdot Z$ with the adversary who takes the role of GM^{ℓ}, where $i := \mathsf{N}^{\ell}_u + 1$. During the protocol, the proof related to C_i (i.e., $E^{\wp}_i$) is simulated. When $\mathcal{B}$ obtains $\langle A_i, e_i, x_i\rangle$ at the end of the joining protocol, it sets $\mathsf{usk}^{\ell}_i = \langle A_i, e_i, x_i, \hat{x}'_i\rangle$, and $\mathsf{jlog}^{\ell}_i = \perp$.
– $\mathsf{Q}_{\mathsf{b-joinauth}}(\ell_1, \ell_2)$.
Let $i_2 := \mathsf{N}^{\ell_2}_u + 1$. If $v \notin \{\ell_1, \ell_2\}$, then $\mathcal{B}$, as a joining user, executes JOINONAUTH protocol with $\mathcal{A}$ as described in our scheme.
If $\ell_1 = v$, then $\mathcal{B}$ firstly chooses $i_1 \in_R \mathsf{U}^{\ell_1}_b$, and computes a simulated signature auth_{i_1} on a random message m using $\mathsf{Q}_{\mathsf{sig}}(\ell_1, i_1, m)$. Then, $\mathcal{B}$ chooses $u_{i_2} \in_R M, x'_{i_2} \in_R \Lambda$, and executes JOINONAUTH with the adversary. During the protocol, the proof related to C_{i_2} (i.e., $E^{\wp}_i$) is simulated. When $\mathcal{B}$ obtains $\langle A_{i_2}, e_{i_2}, x_{i_2}\rangle$ at the end of the joining protocol, it sets $\mathsf{usk}^{\ell_2}_{i_2} = \langle A_{i_2}, e_{i_2}, x_{i_2}, x'_{i_2}\rangle$, and $\mathsf{jlog}^{\ell_2}_{i_2} = \perp$.
Otherwise (i.e., $\ell_2 = v$), $\mathcal{B}$ firstly chooses $i_1 \in_R \mathsf{U}^{\ell_1}_b$, and computes a signature auth_{i_1} on a random message m by executing $\mathrm{SIGN}^{\ell_1}_{i_1}(m)$. Then, $\mathcal{B}$ chooses $\hat{x}'_{i_2} \in_R \Lambda$, and executes JOINONAUTH with the adversary using $C_{i_2} = a^{u_{i_2}} b^{\hat{x}'_{i_2}} \cdot Z$. During the protocol, the proof related to C_{i_2} (i.e., $E^{\wp}_i$) is simulated. When $\mathcal{B}$ obtains $\langle A_{i_2}, e_{i_2}, x_{i_2}\rangle$ at the end of the joining protocol, it sets $\mathsf{usk}^{v_2}_{i_2} = \langle A_{i_2}, e_{i_2}, x_{i_2}, \hat{x}'_{i_2}\rangle$, and $\mathsf{jlog}^{v_2}_{i_2} = \perp$.
– $\mathsf{Q}_{\mathsf{stat}}()$. $\mathcal{B}$ simply returns the specified booking variables.
– $\mathsf{Q}_{\mathsf{sig}}(\ell, i, m)$. If $\ell \neq v$, $\mathcal{B}$ executes $\mathrm{SIGN}^{\ell}_i(m)$. Otherwise, $\mathcal{B}$ finds an entry usk^{ℓ}_i and generates a signature using the simulator of the proof used in SIGN algorithm Specifically, the simulator chooses $T_1, \ldots, T_5$ as described in our scheme, but

$$T_6 = \left(b^{\hat{x}'_i} Z\right)^{k'}, \quad T_7 = b^{k'},$$

where $k' \in M$. Note that the distribution of these T_6 and T_7 is computationally indistinguishable to that of T_6 and T_7 in the description of our scheme.
– $\mathsf{Q}_{\mathsf{reveal}}(\ell, i)$. $\mathcal{B}$ returns $\mathsf{tkey}(\mathsf{usk}^{\ell}_i)$ from uinfo^{ℓ}_i.

Let $\langle m^*, \sigma^*, \gamma^*, \pi^*, m'^*\sigma'^*, \lambda^*\rangle$ be a successful output from $\mathcal{A}$. It holds that $\mathrm{VERIFY}^{v}(m^*, \sigma^*) = \mathsf{acc}$. Now we have the following cases:

1. $\sigma \notin \mathsf{sigs}^v$ and $\exists i \in \mathsf{U}^v_b : \mathrm{OPEN}^v(\sigma) = \mathsf{mref}(\mathsf{usk}^v_i)$.
By applying the *generalized forking lemma* [22], $\mathcal{B}$ can get another valid output $\langle m, \sigma^*_2, \cdots\rangle$ such that that $\mathrm{VERIFY}^{v}(m^*, \sigma^*_2) = \mathsf{acc}$. Note that the protocol used in SIGN is a proof of knowledge, which implies existence of an extractor for the protocol. Therefore, we can extract a witness $\langle x^*, x'^*, e^*, r^*, h'^*\rangle$ from σ^*, σ^*_2. The signature σ^* will be of the following form:

$$\langle T^*_1, \ldots, T^*_7, c, s_{x^*}, s_{x'^*}, s_{e^*}, s_{r^*}, s_{h'^*}\rangle.$$

Let $A^* = \left(T_1/T_2^{\log_g^y}\right)$. Then, it holds that $(A^*)^{e^*} = a_0 a^{x^*} b^{x'^*}$.
Since $\textsc{Open}^v(\sigma) = \mathsf{mref}(\mathsf{usk}_i^v)$, we have $A^* = A_i$. Moreover we have

$$A^* = \left(a_0 a^{x^*} b^{x'^*}\right)^{1/e^*} = z^{\frac{\beta+\alpha x^* + x'^*}{e^*}},$$

and

$$A_i = \left(a_0 a^{x_i} b^{\hat{x}_i} Z\right)^{1/e_i} = z^{\frac{\beta+\alpha x_i + \hat{x}_i + \delta}{e_i}}.$$

Or equivalently

$$(\beta + \alpha x^* + x'^*) \cdot (e^*)^{-1} = (\beta + \alpha x_i + \hat{x}_i + \delta) \cdot (e_i)^{-1} \pmod{p'q'}.$$

So we can find δ, which is $\log_z Z$.

2. $\sigma \notin \mathsf{sigs}^v$ and $\exists i \in \mathsf{U}_b^v \setminus \mathsf{U}_r^v$: $\textsc{Trace}^v(\textsc{Reveal}^v(i), \sigma) = \mathtt{acc}$.
   $\mathcal{B}$ fails in this case.
3. $\sigma \in \mathsf{sigs}^v$ and $\textsc{VerifyClaim}^v(\sigma, \gamma, \pi) = \mathtt{acc}$.
   Let $\langle i, \sigma\rangle$ is the entry in the sigs^v. Note that the values T_6 and T_7 of σ has the following form:

$$T_6 = \left(b^{\hat{x}'_i} Z\right)^{k'}, \quad T_7 = b^{k'},$$

   which means $\log_{T_7} T_6 = \hat{x}'_i + \delta$. By applying the *generalized forking lemma* [22] $\mathcal{B}$ can extract the witness x'^* for the proof in $\textsc{Claim}$ such that $x'^* = \log_{T_7} T_6$, which means $\hat{x}'_i + \delta = x'^*$.
4. $\sigma \in \mathsf{sigs}^v$ and $\textsc{Verify}^{v'}(m', \sigma') = \mathtt{acc}$ and $\textsc{VerifyLink}^{v,v'}(\sigma, \sigma', \lambda) = \mathtt{acc}$.
   By using the similar argument in the case 3, δ can be found.

Game-2
Let V be the number of groups the adversary generated using $\mathsf{Q_{b-gengroup}}$. $\mathcal{B}$ first chooses $j_g \in_R \{1, \ldots, V\}$, and simulates $\mathsf{Q_{b-gengroup}}$ oracle as follows. Again, we assume all the booking variables are appropriately updated.

– $\mathsf{Q_{b-gengroup}}()$.
  Suppose the adversary makes ℓ-th query on this oracle. If $\ell \neq j_g$, then $\mathcal{B}$ executes the $\textsc{Setup}$ algorithm as described in our scheme. Otherwise, $\mathcal{B}$ uses the DLP instance to construct the keys, i.e.,
  $\mathsf{gpk} = \langle n, a = z, b = z^\alpha, a_0 = z^\beta, g, h, y, \hat{n}, \hat{g}, \hat{y}\rangle$ and $\mathsf{gsk} = \langle p, q\rangle$, where $\alpha, \beta \in_R M$ and other values are chosen as described in $\textsc{Setup}$ of our scheme. $\mathcal{B}$ returns $(\ell, \mathsf{gpk}, \mathsf{gsk})$.

Now, $\mathcal{A}$ returns $\langle \mathsf{st}_1, v, v'\rangle$. If $j_g \neq v$, $\mathcal{B}$ aborts; otherwise, $\mathcal{B}$ proceeds. Let N be the number of users in G^v the adversary generated by invoking $\mathsf{Q_{b-join}}(v)$, $\mathsf{Q_{b-joinauth}}(v', v)$. Now $\mathcal{B}$ chooses $j_u \in_R \{1, \ldots, N\}$, and simulates the other oracles provided for $\mathcal{A}$ as follows.

– $\mathsf{Q_{b-join}}(\ell)$.
  Let $i := \mathsf{N}_u^\ell + 1$. If $\ell \neq v$ or $j_u \neq i$, then $\mathcal{B}$, as a joining user, executes $\textsc{Join}$ protocol with $\mathcal{A}$ as described in our scheme. Otherwise, $\mathcal{B}$ executes $\textsc{Join}$ protocol with the adversary who takes the role of GM^v. During the protocol, $\mathcal{B}$ simulates ImpRP protocol such that output is Z (i.e., $a^x = Z$, where x is unknown) by using the simulator of ImpRP. Moreover $\mathcal{B}$ chooses $r_x \in M$ and computes $E_i = E_{\mathsf{fpk}}(r_x)$, and simulates $E_i^{\wp}$. Note that the adversary does not notice that $\mathcal{B}$ used r_x instead of x from the semantic security of the $E_{\mathsf{fpk}}()$. When $\mathcal{B}$ obtains $\langle A_i, e_i\rangle$ at the end of the joining protocol, it sets $\mathsf{usk}_i^\ell = \langle A_i, e_i, r_x, x'\rangle$ (x' are chosen honestly), and $\mathsf{jlog}_i^\ell = \perp$.

- $\mathsf{Q}_{\mathsf{b-joinauth}}(\ell_1, \ell_2)$.
  Let $i := \mathsf{N}_u^\ell + 1$. If $\ell_2 \neq v$ or $j_u \neq i$, then $\mathcal{B}$, as a joining user, executes JoinOnAuth protocol with $\mathcal{A}$ as described in our scheme. Otherwise, $\mathcal{B}$ firstly chooses $i_1 \in_R \mathsf{U}_b^{\ell_1}$, and computes a signature auth_{i_1} on a random message m by executing $\textsc{Sign}_{i_1}^{\ell_1}(m)$. Then $\mathcal{B}$ executes JoinOnAuth protocol with the adversary who takes the role of GM^v. During the protocol, $\mathcal{B}$ simulates ImpRP protocol such that output is Z (i.e., $a^x = Z$, where x is unknown) by using the simulator of ImpRP. Moreover $\mathcal{B}$ chooses $r_x \in M$ and computes $E_i = E_{\mathsf{fpk}}(r_x)$, and simulates $E_i^{\wp}$. When $\mathcal{B}$ obtains $\langle A_i, e_i \rangle$ at the end of the joining protocol, it sets $\mathsf{usk}_i^\ell = \langle A_i, e_i, r_x, x_i' = \mathtt{mkey}(\mathsf{usk}_{i_1}^{\ell_1}) \rangle$, and $\mathsf{jlog}_i^\ell = \perp$.
- $\mathsf{Q}_{\mathsf{stat}}()$. $\mathcal{B}$ simply returns the specified booking variables.
- $\mathsf{Q}_{\mathsf{sig}}(\ell, i, m)$. If $\ell \neq v$, $\mathcal{B}$ executes $\textsc{Sign}_i^\ell(m)$. Otherwise, $\mathcal{B}$ finds an entry usk_i^ℓ and generates a signature using the simulator of the proof used in Sign algorithm Specifically, the simulator chooses $T_1, \ldots, T_5$ as described in our scheme, but

$$T^4 = Z^k, \quad T^5 = a^k,$$

  where $k \in M$. Note that the distribution of these T_4 and T_5 is computationally indistinguishable to that of T_4 and T_5 in the description of our scheme.
- $\mathsf{Q}_{\mathsf{reveal}}(\ell, i)$. If $\ell = v$ and $i = j_u$, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns $\mathsf{tkey}(\mathsf{usk}_i^\ell)$ from uinfo_i^ℓ.

Let $\langle m^*, \sigma^*, \gamma^*, \pi^*, m'^* \sigma'^*, \lambda^* \rangle$ be a successful output from $\mathcal{A}$.

1. $\sigma \notin \mathsf{sigs}^v$ and $\exists i \in \mathsf{U}_b^v : \textsc{Open}^v(\sigma) = \mathsf{mref}(\mathsf{usk}_i^v)$.
   This means $A^* = A_i$. Let $\langle x^*, x'^*, e^*, r^*, h'^* \rangle$ be the extracted witness for the signature σ^*:
$$\langle T_1^*, \ldots, T_7^*, c, s_{x^*}, s_{x'^*}, s_{e^*}, s_{r^*}, s_{h'^*} \rangle.$$
   It also holds that $(A^*)^{e^*} = a_0 a^{x^*} b^{x'^*}$.
   If $i = j_u$, we have
$$A^* = \left(a_0 a^{x^*} b^{x'^*} \right)^{1/e^*} = z^{\frac{\beta + x^* + \alpha x'^*}{e^*}},$$
   and
$$A_i = \left(a_0 Z b^{x_i'} \right)^{1/e_i} = z^{\frac{\beta + \delta + \alpha x_i'}{e_i}}.$$
   Or equivalently
$$(\beta + x^* + \alpha x'^*) \cdot (e^*)^{-1} = (\beta + \delta + \alpha x_i') \cdot (e_i)^{-1} \pmod{p'q'}.$$
   So we can find δ, which is $\log_z Z$. This happens with probability $1/N$.
2. $\sigma \notin \mathsf{sigs}^v$ and $\exists i \in \mathsf{U}_b^v \setminus \mathsf{U}_r^v$: $\textsc{Trace}^v(\textsc{Reveal}^v(i), \sigma) = \mathtt{acc}$.
   If $i = j_u$, we know that $x^* = \delta$. This happens with probability $1/N$.
3. $\sigma \in \mathsf{sigs}^v$ and $\textsc{VerifyClaim}^v(\sigma, \gamma, \pi) = \mathtt{acc}$.
   $\mathcal{B}$ fails in this case.
4. $\sigma \in \mathsf{sigs}^v$ and $\textsc{VerifyLink}^{v,v'}(\sigma, \sigma', \lambda) = \mathtt{acc}$.
   $\mathcal{B}$ fails in this case.

As a result, using $\mathcal{A}$, the algorithm $\mathcal{B}$ can solve the discrete logarithm problem by playing the above games with non-negligible probability. □

Lemma 5. *For honest users who joined with an authentication string, the scheme remains secure against framing attack even the GM maliciously manages the* jlog *database.*

Proof: This lemma holds from the unforgeability of $E_i^{\wp}$: the group manager cannot forge another valid $E_i^{\wp}$ with the same auth and different C_i, X_i. □

Now we prove the anonymity of our scheme.

Lemma 6. *Under the DDH assumption and semantic security of sCS encryption, For any PPT algorithm* $\mathcal{A}$, $\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{anon}}(1^\nu) = 1] - 1/2\right| \leq neg(\nu)$.

Proof: First, we introduce a non-adpative version of the experiment:

```
Experiment $\mathbf{Exp}_{\mathcal{B}}^{\mathsf{anon-na}}(1^\nu)$
  $\langle \mathsf{st}_1, v, i_0, i_1\rangle \leftarrow \mathcal{B}()$;
  $\langle \mathsf{st}_2, m\rangle \leftarrow \mathcal{B}(\mathsf{st}_1 : \mathsf{Q}_{\mathsf{b-gengroup}}, \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
  If $i_0 \notin \mathsf{U}_b^v$ or $i_1 \notin \mathsf{U}_b^v$, then return $\xi$ where $\xi \xleftarrow{R} \{0,1\}$;
  $\delta \xleftarrow{R} \{0,1\}$; $\sigma \leftarrow \textsc{Sign}_{i_b}^v(m)$;
  $\delta^* \leftarrow \mathcal{B}(\mathsf{st}_2, \sigma : \mathsf{Q}_{\mathsf{stat}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{b-authjoin}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
  If $\{i_0, i_1\} \cap \mathsf{U}_r^v \neq \emptyset$, then return $\xi$ where $\xi \xleftarrow{R} \{0,1\}$;
  If $\delta = \delta^*$ then return 1;
  return 0;
```

Define

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(1^\nu) &:= \left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{anon}}(1^\nu) = 1] - 1/2\right| \\ \mathsf{Adv}_{\mathcal{B}}^{\mathsf{anon-na}}(1^\nu) &:= \left|\Pr[\mathbf{Exp}_{\mathcal{B}}^{\mathsf{anon-na}}(1^\nu) = 1] - 1/2\right|.\end{aligned}$$

We prove the lemma by showing the following two statements are true (proof by contradiction); (i) Let $\mathcal{A}$ be an adversary against such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(1^\nu)$ is non-negligble. Then we can construct an algorithm $\mathcal{B}$ such that $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{anon-na}}(1^\nu)$ is non-negligible using $\mathcal{A}$. (ii) For any PPT algorithm $\mathcal{B}$, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{anon-na}}(1^\nu) = neg(\nu)$.

We first show that (i) holds. We construct an algorithm $\mathcal{B}$ as follows:

- $\mathcal{B}$ selects $\langle v, i_0, i_1\rangle$ randomly.
- All the oracle queries from $\mathcal{A}$ are relayed to the oracles provided for $\mathcal{B}$.
- When $\mathcal{A}$ returns $\langle v^*, i_0^*, i_1^*, m^*\rangle$, $\mathcal{B}$ checks if $\langle v, i_0, i_1\rangle \overset{?}{=} \langle v^*, i_0^*, i_1^*\rangle$. If so, $\mathcal{B}$ proceeds execution by returning m to the outer world; otherwise $\mathcal{B}$ aborts.
- Now $\mathcal{B}$ gets a challenge signature σ from the outer world. $\mathcal{B}$ just forwards σ to $\mathcal{A}$.
- When $\mathcal{A}$ outputs δ^*, $\mathcal{B}$ outputs the same δ^*.

Let V be the number of groups generated by $\mathcal{A}$, and let $N = \max_{j\in\{1,\ldots,V\}} |\mathsf{U}_b^j|$. Then it holds that

$$\mathsf{Adv}_{\mathcal{B}}^{\mathsf{anon-na}}(1^\nu) \geq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(1^\nu)/(V \cdot N^2) = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(1^\nu)/poly(\nu),$$

which is non-negligible.

Now we go over to (ii). To show that $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{anon-na}}(1^\nu)$ is negligible, we consider a series of games.

Game H_1. This game is actually $\mathbf{Exp}_{\mathcal{B}}^{\mathsf{anon-na}}(1^{\nu})$. Simulation of all the oracles is done as described in our scheme.

Game H_2. The difference between H_1 and H_2 lies in the simulation of $\mathsf{Q}_{\mathsf{b-join}*}$ oracles.

– $\mathsf{Q}_{\mathsf{b-join}}(\ell)$. Let $i := \mathsf{N}_u^{\ell} + 1$. If $\ell \neq v$ or $i \notin \{i_0, i_1\}$, then H_2, as a joining user, executes JOIN protocol with $\mathcal{B}$ as described in our scheme. Otherwise, during the joining protocol, H_2 encrypts a different message from x_i in E_i, i.e.,

$$C_i = b^{x'_i}, X_i = a^{x_i}, E_i = E_{\mathsf{fgpk}}(R_{x_i}),$$

and then simulates $E_i^{\wp}$ using the simulator. Other parts are remains as they are.

– $\mathsf{Q}_{\mathsf{b-joinauth}}(\ell_1, \ell_2)$. Simulation is similar to $\mathsf{Q}_{\mathsf{b-join}}$.

Let $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{sCS}}$ be the advantage of $\mathcal{A}$ on the IND-CPA experiment of sCS encryption scheme. Since revealing $\mathsf{tkey}(\mathsf{jlog}_{i_0}^{v})$ or $\mathsf{tkey}(\mathsf{jlog}_{i_1}^{v})$ is not allowed, it holds that

$$\Big|\Pr[H_1(\cdot) = 1] - \Pr[H_2(\cdot) = 1]\Big| \leq 2 \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{sCS}}.$$

Game H_3. The difference between H_2 and H_3 lies in the challenge signature σ. In H_2, the challenge signature has the following form:

$$T_1 = A_{i_b} y^w,\ T_2 = g^w,\ T_3 = g^{e_{i_b}} h^w,\ T_4 = g^{x_{i_b} k},\ T_5 = g^k,\ T_6 = g^{x'_{i_b} k'},\ T_7 = g^{k'},$$

where $\langle A_{i_b}, e_{i_b}, x_{i_b}, x'_{i_b} \rangle$ is $\mathsf{usk}_{i_b}^{v}$. In H_3, however, the challege signature is as follows:

$$T_1 = R_1,\ T_2 = y^w,\ T_3 = g^{e_{i_b}} h^w,\ T_4 = g^{x_{i_b} k},\ T_5 = g^k,\ T_6 = g^{x'_{i_b} k'},\ T_7 = g^{k'},$$

where $R_1, R_2 \in QR(n)$. It holds that

$$\Big|\Pr[H_2(\cdot) = 1] - \Pr[H_3(\cdot) = 1]\Big| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH-KF}}.$$

Game H_4. The challenge signature is modified further in this game:

$$T_1 = R_1,\ T_2 = R_2,\ T_3 = R_3,\ T_4 = g^{x_{i_b} k},\ T_5 = g^k,\ T_6 = g^{x'_{i_b} k'},\ T_7 = g^{k'},$$

where $R_3 \in QR(n)$. It holds that

$$\Big|\Pr[H_3(\cdot) = 1] - \Pr[H_4(\cdot) = 1]\Big| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH-KF}}.$$

Game H_5. Further modification of the challenge signature:

$$T_1 = R_1,\ T_2 = R_2,\ T_3 = R_3,\ T_4 = R_4,\ T_5 = R_5,\ T_6 = g^{x'_{i_b} k'},\ T_7 = g^{k'},$$

where $R_4, R_5 \in QR(n)$. It holds that

$$\Big|\Pr[H_4(\cdot) = 1] - \Pr[H_5(\cdot) = 1]\Big| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH-KF}}.$$

Game H_6**.** Another modification of the challenge signature:

$$T_1 = R_1,\ T_2 = R_2,\ T_3 = R_3,\ T_4 = R_4,\ T_5 = R_5,\ T_6 = R_6,\ T_7 = R_7,$$

where $R_6, R_7 \in QR(n)$. It holds that

$$\Big| \Pr[H_5(\cdot) = 1] - \Pr[H_6(\cdot) = 1] \Big| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH-KF}}.$$

Moreover, H_6 does not retain any information about i_b, which implies that $\Pr[H_6(\cdot) = 1] = 1/2$.

In overall, we have

$$\mathsf{Adv}_{\mathcal{B}}^{\mathsf{anon-na}}(1^\nu) \leq 2 \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{sCS}} + 4 \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH-KF}}.$$

□

Lemma 7. *Under the CG-DDH assumption, for any PPT algorithm $\mathcal{A}$,* $\Big| \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{j-anon}}(1^\nu) = 1] - 1/2 \Big| = neg(\nu)$.

Proof: Applying similar argument used to prove the anonymity lemma, we only have to show join-anonymity for non-adaptive adversaries.

```
Experiment $\mathbf{Exp}_{\mathcal{B}}^{\mathsf{j-anon-na}}(1^\nu)$
  $\langle \mathsf{st}_1, v, v', i_0, i_1 \rangle \leftarrow \mathcal{B}()$;
  $\mathsf{st}_2 \leftarrow \mathcal{B}(\mathsf{st}_1 : \mathsf{Q}_{\mathsf{b-gengroup}}, \mathcal{Q}_{\mathsf{j-anon}})$;
  If $i_0$ or $i_1$ is not in $\mathsf{U}_b^v$ or has already joined $G^{v'}$,
    then return $\xi$ where $\xi \overset{R}{\leftarrow} \{0,1\}$;
  $\delta \overset{R}{\leftarrow} \{0,1\}$;
  Execute JoinOnAuth$^{v'}$ with $\mathcal{A}(\mathsf{st}_1)$ using $\mathtt{mkey}(\mathsf{usk}_{i_\delta}^v)$,
  and at the end of the protocol $\mathcal{A}$ returns $\mathsf{st}_2$;
  $\delta^* \leftarrow \mathcal{B}(\mathsf{st}_2 : \mathcal{Q}v_{\mathsf{j-anon}})$;
  If $\{i_0, i_1\} \cap \mathsf{U}_r^v \neq \emptyset$, then return $\xi$ where $\xi \overset{R}{\leftarrow} \{0,1\}$;
  If $\delta = \delta^*$ then return 1;
  return 0;
```

Let $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{j-anon-na}}(1^\nu) := |\Pr[\mathbf{Exp}_{\mathcal{B}}^{\mathsf{j-anon-ns}}(1^\nu) = 1] - 1/2|$. To show that $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{j-anon-na}}(1^\nu)$ is negligible, we consider a series of games.

Game H_1**.** This game is actually $\mathbf{Exp}_{\mathcal{B}}^{\mathsf{j-anon-na}}(1^\nu)$. Simulation of all the oracles is done as described in our scheme.

Game H_2**.** The difference between H_1 and H_2 lies in the challenge protocol JoinOnAuth$^{v'}$. In H_1, the adversary $\mathcal{B}$, playing as the GM$^{v'}$, wlll receive:

$$\mathsf{auth}_{i_\delta} = \langle T_1, \ldots, T_6 = g^{{x'}_{i_\delta} k'}, T_7 = g^{k'}, \ldots \rangle, \quad \text{(in group } G^v\text{)}$$
$$C_{i'} = b^{{x'}_{i_\delta}}, X_{i'} = a^{x_{i'}}, E_{i'} = E_{\mathsf{fgpk}}(x_{i'}),\ E_{i'}^{\wp} \quad \text{(in } G^{v'}\text{)}.$$

However, H_2 gives to $\mathcal{B}$:

$$\begin{array}{ll}\mathsf{auth}_{i_b} = \langle T_1, \ldots, T_6 = g^{{x'_{i_b}}^{k'}}, T_7 = g^{k'}, \ldots \rangle, & \text{(in group } G^v\text{)} \\ C_{i'} = b^{r_{x'}}, X_{i'} = a^{x_{i'}}, E_{i'} = E_{\mathsf{fgpk}}(x_{i'}),\ E^{\wp}_{i'} & \text{(in } G^{v'}\text{)}\end{array}$$

where $r_{x'} \in_R M$. From the CG-DDH assumption and the zero-knowledge property of $E^{\wp}_i$, we have

$$\Big| \Pr[H_2(\cdot) = 1] - \Pr[H_1(\cdot) = 1] \Big| = neg(\nu).$$

Observe that Lemma 6 gurantees auth_{i_δ} remains anonymous. Moreover $C_{i'}, E_{i'}$ does not retain any information about i_δ, which implies that $\Big| \Pr[H_2(\cdot) = 1] - 1/2 \Big| = neg(\nu)$.

In overall, we have

$$\mathsf{Adv}^{\mathsf{j-anon-na}}_{\mathcal{B}}(1^\nu) = \Big| \Pr[H_1(\cdot) = 1] - 1/2 \Big| = neg(\nu).$$

□

Lemma 8. *For any PPT algorithm $\mathcal{A}$,* $\Pr[\mathbf{Exp}^{\mathsf{link}}_{\mathcal{A}}(1^\nu) = 1] = neg(\nu)$*.*

Proof: This lemma simply holds from the soundness of the proof of knowledge used in ClaimLink algorithm. □

Lemma 9. *Our scheme satisfies trusted fairness.*

Proof: This holds simply from the fact that we used threshold cryptosystem. Therefore, even if $\zeta - 1$ fairness authorities collaborate toghether, they cannot decrypt any messages. This means that the view of an adversary with $\mathsf{Q}_{\mathsf{fa-share}}$ is computationally indistinguishable to the view without the oracle. Note that we assumed that fairness authorities do not abort, which made us use more simplified threshold schemes.

□

B The Adaptive One-More Representation Problem

We prove the Lemma 1 in this section. Firstly, to make our life easier, we use the following the lemma due to [23]. For completeness, we prove the lemma. Recall that the Strong-RSA problem is, given an RSA modulus n and $z \in \mathbb{Z}^*_n$, to find a pair $\langle u, e \rangle$ such that $u^e = z \pmod{n}$.

Lemma 10. *Let $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$ with p, q, p', q' all prime numbers. Suppose we know $y, z \in QR(n)$ and $t, m \in \mathbb{Z}$ such that $y^t = z^m \pmod{n}$ with $\gcd(t, m) < t$ and $t > 1$. Then we can find $e > 1$ and $u \in \mathbb{Z}^*_n$ such that $z = u^e \pmod{n}$, or we can factor n.*

Proof:

- Case (i): $\gcd(t, m) = 1$.
  We can compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha t + \beta m = 1$. From this, in turn, we obtain:

$$z = z^{\alpha t + \beta m} = (z^\alpha)^t)(z^\beta)^t = (z^\alpha y^\beta)^t$$

  and thus, we return the pair $\langle u, e \rangle = \langle z^\alpha y^\beta, t \rangle$ as a solution.

– Case (ii): $\gcd(t, m) = \delta > 1$ and $\gcd(\delta, p'q') = 1$.
  It follows that $\delta \leq \min(|t|, |m|)$ and if $t' = t/\delta$ and $m' = m/\delta$, it holds that $(y^{t'})^{\delta} = (z^{m'})^{\delta}$. Since $\gcd(\delta, p'q') = 1$ we obtain that $y^{t'} = z^{m'}$ with $\gcd(t', m') = 1$; moreover $t' > 1$ since $\gcd(t, m) = \delta < t$. Thus we reduce the case (ii) to case (i).
– Case (iii): $\gcd(\delta, p'q') > 1$.
  It follows that δ is a multple of p' (WLOG). Then we can factor n as follows: choose a random integer w less than n; if $\gcd(w, n) > 1$ then we are done; otherwise $w \in \mathbb{Z}_n^*$ and w is a square modulo p with probability about 1/2. It follows that $w^{p'} = (w^{1/2})^{2p'} = (w^{1/2})^{p-1} = 1 \pmod p$. Now compute the integer $U = w^{\delta} = w^{\pi p'} \pmod n$, where $\delta = \pi p'$ for some $\pi \in \mathbb{Z}$. Then we have $U = w^{\pi p'} = (w^{p'})^{\pi} = 1 \pmod p$. It follows that there exists an $r \in \mathbb{Z}$ such that $U - 1 = rp$. Observe that it has to be that $r < q$ since $U < n$. From this we obtain that $\gcd(U - 1, n) = p$.

□

Now, we start to prove Lemma 1. Recall that Q_{rep} is an oracle that, on input $x'_i \in \Lambda$, ouputs A_i, e_i, x_i such that $A_i^{e_i} = a^{x_i} b^{x'_i} a_0$ holds with $x_i \in M$, and a prime $e_i \in \Gamma$ (i.e., $\langle A_i, e_i, x_i, x'_i \rangle$ is a DL-representation). We show that, if there is a PPT algorithm $\mathcal{A}$ that solves the adaptive one-more representation problem with non-negligible probability, there is a PPT algorithm $\mathcal{B}$ that solves the Strong-RSA with non-negligible probability using $\mathcal{A}$. This proof is very similar to the non-adaptive case [22].

Suppose that $\mathcal{A}$ outputs $\langle \hat{A}, \hat{e}, \hat{x}, \hat{x}' \rangle$. The algorithm $\mathcal{B}$ is composed of four games that are played at random. Each game may fail according to the following specifications:

1. Game 1 will fail if $\hat{e}$ has a non-trivial common divisor with any of the values $e_1, \ldots, e_k$.
2. Game 2 will fail if (i) $\hat{e}$ is relatively prime to all values $e_1, \ldots, e_k$, and (ii) it is not possible to find a $j \in [1, K]$ for which it holds that $a^{\hat{x} - x_j} b^{\hat{x}' - x'_j} = 1$ with either $\hat{x} \neq x_j$ or $\hat{x}' \neq x'_j$.
3. Game 3 will fail if (i) $\hat{e}$ is relatively prime to all values $e_1, \ldots, e_k$, and (ii) for a pre-selected value j it does not hold that e_j divides $\hat{e}$ and it does not hold that $x_j = \hat{x}$ and $x'_j = \hat{x}'$.
4. Game 4 will fail if (i) $\hat{e}$ is relatively prime to all values $e_1, \ldots, e_k$ and (ii) for a pre-selected value j it does not hold that e_j divides $\hat{e}$ and it does not hold that $a^{\hat{x} - x_j} b^{\hat{x}' - x'_j} \neq 1$.

Observe that playing the above games at random covers all possible behaviors of the algorithm $\mathcal{A}$ with respect to the relation of the output DL-representation to the initial ones. We follow the notation described in the Preliminaries. The description of the games follows:

Game-1

1. $\mathcal{B}$ selects prime numbers $e_1, \cdots, e_k$ randomly from Γ, sets $a = z^{e_1 \cdots e_K}$, $a_0 = a^r$ and $b = a^{r'}$ where $r, r' \in_R M$. $\mathcal{B}$ gives $\langle n, a_0, a, b \rangle$ to $\mathcal{A}$. Note that a_0, a, b are indistinguishable from random elements from $QR(n)$.
2. When $\mathcal{A}$ invokes $Q_{\mathsf{rep}}(x'_i)$, $\mathcal{B}$ selects $x_i \in M$, and computes $A_i = z^{(x_i + r + r' x'_i) \frac{e_1 \cdots e_K}{e_i}}$. It returns $\langle A_i, e_i, x_i, x'_i \rangle$, which is a DL-representation. In this way, $\mathcal{B}$ can simulate Q_{rep} at most K times.
3. If $\mathcal{A}$ aborts, then $\mathcal{B}$ also aborts. If $\mathcal{A}$ outputs a valid DL-representation $\langle \hat{A}, \hat{e}, \hat{x}, \hat{x}' \rangle$ such that $\gcd(\hat{e}, e_i) > 1$ for some $i \in [1, K]$, then $\mathcal{B}$ aborts. Otherwise, now $\mathcal{B}$ has

$$\hat{A}^{\hat{e}} = a_0 a^{\hat{x}} b^{\hat{x}'} = z^{(r + \hat{x} + r' \hat{x}') e_1 \cdots e_K}.$$

If we let $\tilde{e} := (r+\hat{x}+r'\hat{x}')e_1 \cdots e_K$, we have $\hat{A}^{\hat{e}} = z^{\tilde{e}}$. Compute $\delta := \gcd(\hat{e}, \tilde{e}) = \gcd(\hat{e}, r+\hat{x}+r'\hat{x}')$. Since $r, r', \hat{x} \in M$, $\hat{x}' \in \Lambda$, and $\hat{e} \in \Gamma$, we have

$$\delta \leq r + \hat{x} + r'\hat{x'} < \hat{e}.$$

Therefore, from Lemma 10, we can solve the problem.

Game-2

1. $\mathcal{B}$ selects prime numbers $e_1, \cdots, e_k$ randomly from Γ. Then it flips a random coin bit $\sigma \in_R \{0,1\}$, if $\sigma = 0$ it sets $v_a = z$ and $v_b = z^{r'}$, otherwise it sets $v_a = z^{r'}$ and $v_b = z$, where $r' \in_R M$. Then it sets $a = v_a^{e_1 \cdots e_K}$, $a_0 = a^r$ and $b = v_b^{e_1 \cdots e_K}$ where $r \in_R M$. $\mathcal{B}$ gives $\langle n, a_0, a, b \rangle$ to $\mathcal{A}$. Note that a_0, a, b are indistinguishable from random elements from $QR(n)$.
2. When $\mathcal{A}$ invokes $Q_{\mathsf{rep}}(x_i')$, $\mathcal{B}$ selects $x_i \in M$, and computes $A_i = (v_a^r v_a^{x_i} v_b^{x_i'})^{\frac{e_1 \cdots e_K}{e_i}}$. It returns $\langle A_i, e_i, x_i, x_i' \rangle$, which is a DL-representation. In this way, $\mathcal{B}$ can simulate Q_{rep} at most K times.
3. If $\mathcal{A}$ aborts, then $\mathcal{B}$ also aborts. Otherwise $\mathcal{A}$ outputs a valid DL-representation $\langle \hat{A}, \hat{e}, \hat{x}, \hat{x}' \rangle$.
4. If $\gcd(\hat{e}, e_i) = 1$ for all $i \in [1, K]$, then $\mathcal{B}$ aborts.
5. If $a^{\hat{x}-x_i} \neq b^{x_i'-\hat{x'}}$ or $\hat{x} - x_i = x_i' - \hat{x'} = 0$ for all $i \in [1, K]$, then $\mathcal{B}$ aborts. Otherwise, there is an index j such that $a^{\hat{x}-x_j} = b^{x_j'-\hat{x'}}$ and either $\hat{x} - x_j$ or $x_j' - \hat{x'}$ is not zero. Denote $\Delta := \hat{x} - x_j$ and $\Delta' := x_j' - \hat{x'}$. Then we have

$$v_a^{e_1 \cdots e_K \Delta} = a^{\Delta} = b^{\Delta'} = v_b^{e_1 \cdots e_K \Delta'},$$

which implies

$$v_a^{\Delta} = v_b^{\Delta'}$$

since $\gcd(e_1 \cdots e_K, p'q') = 1$ (otherwise we can factor n.)
6. When $\Delta \neq 0$ and $\Delta' \neq 0$, compute Compute $\delta := \gcd(\Delta, \Delta')$, and α, β such that $\delta = \alpha\Delta + \beta\Delta'$. We know that $\delta \leq \Delta'$.
If $\delta < \Delta'$ and $\sigma = 0$, then we have

$$z = v_a = v_a^{\frac{\alpha\Delta + \beta\Delta'}{\delta}} = (v_b^{\alpha} v_a^{\beta})^{\Delta'/\delta}.$$

we found $u = v_a^{\alpha} v_b^{\beta}$ and $e = \Delta'/\delta > 1$ such that $z = u^e$.
If $\delta = \Delta'$ and $\sigma = 1$, then Δ/Δ' is an integer, which implies

$$z = v_b = v_a^{\Delta/\Delta'}.$$

So we found $u = v_a$ and $e = \Delta/\Delta' > 1$ such that $z = u^e$.
7. When $\Delta = 0$ and $\Delta' \neq 0$ then we have

$$v_b^{\Delta'} = 1,$$

which means Δ' is non-trivial divisor of $|QR(n)|$. So we can factor n (see Lemma 10), and solve the Strong-RSA problem. The case where $\Delta \neq 0$ and $\Delta' = 0$ is symmetric.

Game-3

1. $\mathcal{B}$ selects prime numbers $e_1, \cdots, e_k$ randomly from $\mathit{\Gamma}$, and j randomly from $[1, K]$. Then it sets $a = v_a^{\frac{e_1 \cdots e_K}{e_j}}$, $a_0 = A_j^{e_j}/a^s$ and $b = v_b^{\frac{e_1 \cdots e_K}{e_j}}$ where $A_j = z^{\frac{e_1 \cdots e_K}{e_j}}$, $v_a = z^r$ and $v_b = v_a^{r'}$ with $r \in_R M$, $r' \in_R [1, 2^{\nu/4}]$, $s \in_R [r' \cdot 2^{\nu/4}, 2^{\nu/2}]$. $\mathcal{B}$ gives $\langle n, a_0, a, b\rangle$ to $\mathcal{A}$. Note that a_0, a, b are indistinguishable from random elements from $QR(n)$.
2. $\mathcal{B}$ answers to $\mathcal{A}$'s query with $Q_{\mathsf{rep}}(x_i')$. If $i = j$ (i.e., j-th query), then $\mathcal{B}$ computes $x_j = s - r'x_j'$ (note that $x_j \in M$) and returns $\langle A_j, e_j, x_j, x_j'\rangle$; otherwise $\mathcal{B}$ selects $x_i \in_R M$ and computes
$$A_i = (z^{e_j} v_a^{x_i} v_b^{x_i'} v_a^{-s})^{\frac{e_1 \cdots e_K}{e_i e_j}} = \left(\left(\frac{z^{e_j}}{v_a^s}\right)^{\frac{e_1 \cdots e_K}{e_j}}\right)^{1/e_i} \cdot (a^{x_i} b^{x_i'})^{1/e_i}.$$
It returns $\langle A_i, e_i, x_i, x_i'\rangle$, which is a DL-representation. In this way, $\mathcal{B}$ can simulate Q_{rep} at most K times.
3. If $\mathcal{A}$ aborts, then $\mathcal{B}$ also aborts. Otherwise $\mathcal{A}$ outputs a valid DL-representation $\langle \hat{A}, \hat{e}, \hat{x}, \hat{x}'\rangle$.
4. If $\gcd(\hat{e}, e_j) = 1$, then $\mathcal{B}$ aborts.
5. If $\hat{x} \neq x_j$ or $\hat{x'} \neq x_j'$, then $\mathcal{B}$ aborts. Otherwise, we have $\hat{x} = x_j$ and $\hat{x}' = x_j'$, which implies
$$\hat{A}^{\hat{e}} = a_0 a^{\hat{x}} b^{\hat{x}'} = a_0 a^{x_j} b^{x_j'} = A_j^{e_j} = z^{e_1 \cdots e_K}.$$
Compute $\delta = \gcd(\hat{e}, e_1 \cdots e_K)$. If $\delta < \hat{e}$, we are done from Lemma 10. If $\delta = \hat{e}$, it means either $\hat{e} = e_j$ (impossible; then the representation output would not be fresh) or $\hat{e} = e_j e_{j'}$ (also impossible from the sphere restriction).

Game-4

1. $\mathcal{B}$ selects prime numbers $e_1, \cdots, e_k$ randomly from $\mathit{\Gamma}$, and j randomly from $[1, K]$. Then it sets $a = z^{\frac{e_1 \cdots e_K}{e_j}} \pmod n$, $a_0 = A_j^{e_j}/a^s$ and $b = a^{r'}$ where $A_j = a^r$ with $r \in_R M$, $r' \in_R [1, 2^{\nu/8}]$, $s \in_R [r' \cdot 2^{\nu/4}, 2^{\nu/2}]$. $\mathcal{B}$ gives $\langle n, a_0, a, b\rangle$ to $\mathcal{A}$. Note that a_0, a, b are indistinguishable from random elements from $QR(n)$.
2. $\mathcal{B}$ answers to $\mathcal{A}$'s query with $Q_{\mathsf{rep}}(x_i')$. If $i = j$ (i.e., j-th query), then $\mathcal{B}$ computes $x_j = s - r'x_j'$ (note that $x_j \in M$) and returns $\langle A_j, e_j, x_j, x_j'\rangle$; otherwise $\mathcal{B}$ selects $x_i \in_R M$ and computes
$$A_i = z^{(x_i + r'x_i' + re_j - s)\frac{e_1 \cdots e_K}{e_i e_j}} = z^{(x_i + r'x_i' + re_j - s)\frac{e_1 \cdots e_K}{e_i e_j}} = \left(a^{re_j - s} a^{x_i} b^{x_i'}\right)^{1/e_i}.$$
It returns $\langle A_i, e_i, x_i, x_i'\rangle$, which is a DL-representation. In this way, $\mathcal{B}$ can simulate Q_{rep} at most K times.
3. If $\mathcal{A}$ aborts, then $\mathcal{B}$ also aborts. Otherwise $\mathcal{A}$ outputs a valid DL-representation $\langle \hat{A}, \hat{e}, \hat{x}, \hat{x}'\rangle$.
4. If $\gcd(\hat{e}, e_j) \neq e_j$, then $\mathcal{B}$ aborts. Otherwise, we have $\hat{e} = te_j$ for some $t \in \mathbb{Z}$. Next we check whether $\hat{x} - x_j + r'(\hat{x'} - x_j') \neq 0$ and in this case we proceed as follows. Let $Z := \hat{A}^t/A_j$ and $\gamma := \hat{x} - x_j + r'(\hat{x'} - x_j')$. Then we have
$$Z^{e_j} = \left(\frac{\hat{A}^t}{A_j}\right)^{e_j} = \frac{\hat{A}^{\hat{e}}}{A_j^{e_j}} = \frac{a_0 a^{\hat{x}} b^{\hat{x'}}}{a_0 a^{x_j} b^{x_j'}} = a^{\hat{x} - x_j + r'(\hat{x'} - x_j')} = a^{\gamma} = z^{\gamma \frac{e_1 \cdots e_K}{e_j}}.$$
If $a^{\gamma} = 1$, then we can factor n (see Lemma 10). Otherwise, let $\tilde{e} := \gamma\frac{e_1 \cdots e_K}{e_j}$. The it follows $Z^{e_j} = z^{\tilde{e}}$. Compute $\delta := \gcd(e_j, |\tilde{e}|) = \gcd(e_j, |\gamma|)$. Since $e_j \in \mathit{\Gamma}$ is a prime number, $\delta = 1$. Therefore by Lemma 10 we can solve the Strong-RSA problem.

As a result, using $\mathcal{A}$, the algorithm $\mathcal{B}$ can solve the Strong-RSA problem by playing the above games. If α is the success probability of $\mathcal{A}$, it is easy to see that the above algorithm will solve the Strong-RSA problem with success probability at least $\alpha/2K$.