Efficient Group Signature Scheme without Pairings

Ke Gu[†,‡], Bo Yin[‡]

[†] School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China

[‡] School of Information Science and Engineering, Central South University, Changsha 410083, China

Abstract

Group signature is a useful cryptographic primitive, which makes every group member sign messages on behalf of a group they belong to. Namely group signature allows that group member anonymously signs any message without revealing his/her specific identity. However, group signature may make the signers abuse their signing rights if there are no measures of keeping them from abusing signing rights in the group signature schemes. So, group manager must be able to trace (or reveal) the identity of the signer by the signature when the result of the signature needs to be arbitrated, and some revoked group members must fully lose their capability of signing a message on behalf of the group they belong to. A practical model meeting the requirement is verifier-local revocation, which supports the revocation of group member. In this model, the verifiers receive the group member revocation messages from the trusted authority when the relevant signatures need to be verified.

Although currently many group signature schemes have been proposed, most of them are constructed on pairings. In this paper, we present an efficient group signature scheme without pairings under the model of verifier-local revocation, which is based on the modified EDL signature (first proposed by D. Chaum et al. in Crypto 92). Compared with other group signature schemes, the proposed scheme does not employ pairing computation and has the constant signing time and signature size, whose security can be reduced to the computational Diffie-Hellman (CDH) assumption in the random oracle model. Also, we give a formal security model for group signature and prove that the proposed scheme has the properties of traceability and anonymity.

Index Terms

group signature, EDL signature, pairings, security model

I. Introduction

A. Background

Group signature [1] allows group member (signer) to hide his identifying information to a group when group member signs messages, thus group signature only reveals the fact that a message was signed by possible one of group members (a list of possible signers). Additionally, in a practical group signature scheme, the group must be constructed by a group manager, who can revoke the anonymity of any signer or identify the real group signer. Because a list of possible signers must be constructed to form a group, some intricate problems need to be solved, such as joining the new members and the revocation of group members. Ateniese et al. [2] first proposed an efficient and provably coalition-resistant group signature scheme. However, the security of coalition-resistant group signature was not formalized. In [3], Bellare et al. summarized the requirements of group signature and showed the security definitions of group signature. Boneh et al. [4] proposed a short group signature scheme in the random oracle model.

In public key cryptography, the management of public keys is a critical problem. For example, certificate authority (CA) generates a digital certificate, which assures that public key belongs to the corresponding user. Then, in a group signature scheme based on public key cryptography, a group public key is corresponding to multi-distributing private keys (signing keys), the joining and revocation of group member is an intricate problem [5,6,7,8]. For large group, it is inefficient to update group public key and distributing private keys when a user joins or exits a group. Bresson et al. [5] proposed that the signer may prove that his group certificate does not belong to a list of revoked certificates. However, the length of group signature is proportional to the number of revoked group members. Camenisch et al. [7] proposed a different way to handle this problem by using accumulators[1]. However, in some pairing-based accumulators [9,10], the size of public keys linearly grows with the maximal number of accumulations.

The method of verifier-local revocation was proposed by Brickell in [11]. Boneh et al. [8] gave the formal definitions of verifier-local revocation. In this kind of approaches [12,13,14,15], the verifiers receive the revocation list of group members from the authority (such as private key generator) when a signature needs to be verified, and non-revoked group members do not need to

[1]An accumulator is a kind of ”hash” function mapping a set of values to a short, constant-size string while allowing to efficiently prove that a specific value was accumulated.

update their distributing private keys. So, the length of signature does not depend on the number of revoked group members in this model, and the verifiers only need to perform an additional computing to test that whether the signature was signed by a revoked group member on the revocation list of group members. Of course, this kind of approaches increase the verification cost being proportional to the size of the revocation list.

In 2009, Nakanishi et al. [16] proposed a revocable group signature scheme with constant complexities for signing and verifying. Also, group members do not need to update their distributing private keys. However, the size of public keys linearly grows with the maximal number N of users in their scheme. In 2012, Libert et al. [17,18] proposed two group signature schemes based on public key cryptography, which have many useful properties [18]: O$(\log N)$-size group public keys, revocation lists of size O(r) ((r) is the number of revoked users), constant membership certificate size, constant signature size and verification time. However, their schemes need to employ pairing computation.

Additionally, with a rapid development of identity-based cryptography [21,22,23,24], some researchers proposed many identity-based signature schemes in the random oracle model or standard model [23,25,26,27]. So, with these identity-based signature (IBS) schemes, a lot of variants, such as the identity-based ring signature schemes [28,29,30], the identity-based group signature schemes [19,20], etc, have also been proposed. In 2011, Ibraimi et al. [19] proposed an identity-based group signature with membership revocation in the standard model. However, their security model is not enough complete for identity-based group signature, some notions are confused. And their scheme is not fully identity-based group signature scheme, the master key of the system is still constructed on public key cryptography. In 2014, Emura et al. [20] proposed an γ-hiding revocable group signature scheme in the random oracle model. Because their scheme introduces the notion of attributes, their scheme is enough complex and inefficient.

EDL signature

The EDL signature [31] and its variant [32] are respectively proposed in 1992 and 1999. Because the computations of the EDL signature do not employ pairings, the efficiency of the schemes is very high. In 2003, Goh et al. [33] proved the security of the EDL signature may be reduced to the CDH assumption in the random oracle model. In 2005, Chevallier-Mames [34] further improved the efficiency of the EDL signature by offline/online computation and signature coupon [35], whose security may also be reduced to the CDH assumption in the random oracle model.

B. Our Contributions

In this paper, we present a public key-based group signature scheme without pairings under the model of verifier-local revocation. Also, we give the formal security models for group signature. Under our security models, the proposed scheme is proved to have the properties of anonymity and traceability with enough security in the random oracle model. In this paper, our contributions are as follows:

- We present a public key-based (and verifier-local revocation) group signature scheme without pairings, which is based on the modified EDL signature. By modifying the EDL signature from [33,34], we twice use the modified EDL signature to build a complete group signature scheme: a) we first use the modified EDL signature to construct the partial member private keys when the users join a group; b) we again use the modified EDL signature to generate the valid signatures.
- We present a framework for group signature and show a detailed security model. We introduce the Libert et al.'s models [18,19] to our security model. In our security model, we consider three situations for the security of group signature. Under our security model, the proposed group signature scheme is proved to be secure and has a security reduction to the simple standard assumption (computational Diffie-Hellman assumption) in the random oracle model. So, no poly-time adversary can produce a valid group signature on any messages when the adversary may adaptively be permitted to choose messages after executing group-setup oracle, join-user oracle, revoke-user oracle, signature oracle and trace-user oracle.
- Compared with other group signature schemes proposed by [12,14,18,19,20], the proposed group signature scheme is not based on pairing computation, and has the constant signing time and signature size (the comparisons of the schemes are given in Section 6).

C. Outline

The rest of this paper is organized as follows. In Section 2, we review the bilinear pairings and complexity assumptions on which we build. In Section 3, we show a framework for group signature. In Section 4, we set up the security models for group signature. In Section 5, we propose a group signature scheme under our proposed signature framework. In Section 6, we analyze the efficiency and security of the proposed scheme. Finally, we draw our conclusions in Section 7.

II. Preliminaries

Definition 2.1 Computational Diffie-Hellman (CDH) Problem: *Let $\mathbb{G}_1$ be a group of prime order q and g be a generator of $\mathbb{G}_1$; for all $(g, g^a, g^b) \in \mathbb{G}_1$, with $a, b \in \mathbb{Z}_q$, the CDH problem is to compute $g^{a \cdot b}$.*

Definition 2.2 *The $(\hbar, \varepsilon)$-CDH assumption holds if no $\hbar$-time algorithm can solve the CDH problem with probability at least ε.*

III. A Framework for Group Signature

Definition 3.1 Group Signature Scheme: *Let* **GS=(*System-Setup*, *Generate-Key*, *Group-Setup*, *Join-User*, *Revoke-User*, *Sign*, *Verify*, *Trace-User*)** *be a group signature scheme. In* **GS**, *all algorithms are described as follows:*

1) ***System-Setup****: The randomized algorithm run by the trusted authority inputs a security parameter 1^k, and then outputs all system parameters GK on the security parameter 1^k.*
2) ***Generate-Key****: The randomized algorithm run by a group member generates his public/private key pair (pk_i, sk_i) with $i \in \{1, 2......n\}$, where n is the maximal number of users in a group, pk_i is the public key of the group member i and sk_i is the private key of the group member i.*
3) ***Group-Setup****: The randomized algorithm run by the trusted authority inputs (GK, $Infor \in \{0, 1\}^*$), and then outputs a group private key sk_g to a group manager, where $Infor$ is a group public identity information (or $Infor$ is seen as the public key of group), sk_g is a group private key on the management of the group manager.*
4) ***Join-User****: The randomized algorithm run by the group manager inputs (GK, sk_g, pk_i), and then outputs a member private key csk_i to a group member, where csk_i is the member private key of the group member and $i \in \{1, 2......n\}$.*
5) ***Revoke-User****: The randomized algorithm run by the group manager inputs (GK, sk_g, pk_i, RL_{pk}^t), and then outputs an updated revocation list RL_{pk}^{t+1}, where pk_i is the public key of the revoked user, $RL_{pk}^t = \{...(pk_j, \Re_{pk_j})...\}$ is a revocation list in the duration t*

(pk_j is the public key of the revoked user and $\Re_{pk_j}$ is a credential on the corresponding public key).

6) ***Sign****: The randomized algorithm is a standard group signature algorithm. Signer needs to sign a message $\mathfrak{M} \in \{0,1\}^*$. The algorithm run by a group member inputs (GK, csk_i, $\mathfrak{M}$), and then outputs a signature σ, where $\sigma \in \{0,1\}^* \cup \{\perp\}$, csk_i is the member private key of the group member with $i \in \{1, 2......n\}$.*

7) ***Verify****: The signature receivers verify a standard group signature σ. The deterministic algorithm run by a signature verifier inputs (GK, $\mathfrak{M}$, $Infor$, σ, RL_{pk}^t), and then outputs the boolean value, $accept$ or $reject$.*

8) ***Trace-User****: The group manager traces a real group member (signer) on group signature σ. The deterministic algorithm run by the group manager inputs (GK, $\mathfrak{M}$, $Infor$, sk_g, σ, RL_{pk}^t), and then outputs the corresponding public key of the real signer or $\perp$.*

The *correctness* of **GS** requires that for any $GK \leftarrow$***System-Setup***(1^k), $sk_g \leftarrow$ ***Group-Setup***$(GK$, $Infor \in \{0,1\}^*)$, $csk_i \leftarrow$***Join-User***$(GK$, sk_g, $pk_i)$ for all i with $i \in \{1, 2......n\}$, $\mathfrak{M} \in \{0,1\}^*$, then

Pr[***Verify***$(GK$, $\mathfrak{M}$, $Infor$, ***Sign***$(GK$, csk_i, $\mathfrak{M})$, $RL_{pk}^t)$=1]=1.

The *traceability* of **GS** requires that for any $GK \leftarrow$***System-Setup***(1^k), $sk_g \leftarrow$ ***Group-Setup***$(GK$, $Infor \in \{0,1\}^*)$, $csk_i \leftarrow$***Join-User***$(GK$, sk_g, $pk_i)$ for all i with $i \in \{1, 2......n\}$, $\mathfrak{M} \in \{0,1\}^*$, then

Pr[***Trace-User***$(GK$, $\mathfrak{M}$, $Infor$, sk_g, ***Sign***$(GK$, csk_i, $\mathfrak{M})$, $RL_{pk}^t)$=pk_i]=1,

where the public key pk_i belongs to the group named by the identity information $Infor$.

IV. Security Model

According to [18,19], we consider that a secure group signature scheme must meet the following three security requirements:

1) **Unforgeability**: A valid group signature must be signed by a valid group member (signer). Therefore, no poly-time adversary can produce a valid group signature on any messages when the adversary may adaptively be permitted to choose messages after executing group

setup oracle, joining user oracle, revoking user oracle, signature oracle and tracing user oracle.

2) **Anonymity**: A valid group signature can only reveal that one group identity possessed by a group manager satisfies the signature. It means a valid group signature can hide the identifying information of real signer to one group.
3) **Traceability**: In some situations, a valid group signature needs to reveal the identity (or public key) of real signer from one group. It means a valid group signature can trace a real signer. Then we split the requirement to the following two small security notions[2] [18]:
   a) the first one is called security against *misidentification attacks*, which requires that even if the adversary can introduce (or corrupt) and revoke any user, a valid group signature can not reveal the identifying information outside the set of the identities of unrevoked adversarially-controlled users.
   b) the second one is called security against *framing attacks*, which requires that an honest user is only responsible for the messages that he signed, namely there is no situation that a valid group signature can reveal the identity of a real group member (signer) but this signer did not sign this signature.

Based on the above three situations, we propose a complete security model for group signature. To make our security model easier to understand, we construct several algorithms interacting with adversary, which may make attack experiments to the group signature schemes in the above three situations. In our security model, we maximize adversary's advantage, and assume that all attacking conditions needed by adversary hold and adversary may forge signatures after limitedly querying oracles in the above three situations.

In our security model, we assume there are n users in a group signature scheme ($n \in \mathbb{N}$ is a maximal number of group members), and at least one user u^* of n users is not corrupted by adversary. And we maximize adversary's advantage, where adversary can get all useful information except for the private key of u^*.

All symbols and parameters are defined as follows in the algorithms:

(1) U^a is a set of users that were registered by an adversary in this game, where the user $u_i^a \in U^a$ with $i \in \{1, 2......\}$, pk_i^a is the public key of the user u_i^a.

[2]The two security notions are more detailedly expanded from the correctness of traceability.

(2) U^b is a set of honest users when an adversary acts a dishonest group manager in this game, where the user $u_i^b \in U^b$ with $i \in \{1, 2......\}$, pk_i^b is the public key of the user u_i^b.

(3) k is a secure parameter, $\mathcal{A}$ represents an adversary.

Definition 4.1 Unforgeability of A Group Signature Scheme: *Let* **GS***=(**System-Setup**, **Generate-Key**, **Group-Setup**, **Join-User**, **Revoke-User**, **Sign**, **Verify**, **Trace-User**) be a group signature scheme. Additionally, we set that k is a secure parameter, and **Pr**($\mathcal{B}_{U\_GS}$(k,$\mathcal{A}$)=1) is the probability that the algorithm $\mathcal{B}_{U\_GS}$ returns 1. Then the advantage that the adversary $\mathcal{A}$ breaks* **GS** *is defined as follows:*

$$\mathbf{Adv}_{GS}^{u\_gs-uf}(k, q_g, q_j, q_s, \hbar)=\mathbf{Pr}(\mathcal{B}_{u\_gs}(k,\mathcal{A})=1),$$

where q_g is the maximal number of "Group-Setup" oracle queries, q_j is the maximal number of "Join-User" oracle queries, q_s is the maximal number of "Sign" oracle queries and $\hbar$ is the running time of $\mathcal{B}$. If the advantage that the adversary breaks **GS** *is negligible, then the scheme* **GS** *is secure.*

According to the Definition 4.1, the algorithm $\mathcal{B}_{U\_GS}$ is described as follows:

1.**Setup**: Running ***System-Setup***, $GK \leftarrow$***System-Setup***(1^k), and then GK is passed to $\mathcal{A}$.

2.**Queries**: $\mathcal{A}$ makes queries to the following oracles for polynomially many times:

- ***Group-Setup***(): Given the public parameters GK and the identity information $Infor$ of the group, the oracle returns a group private key sk_g to $\mathcal{A}$.
- ***Join-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$) and the public key pk_i of the group member, the oracle returns a group member private key csk_i to $\mathcal{A}$, where sk_g is a group private key on the identity $Infor$ of the group.
- ***Sign***(): Given the public parameters GK, the group member private key csk_i (or the public key pk_i) and the message $\mathfrak{M}$, the oracle returns a signature σ to $\mathcal{A}$, where $\sigma \in \{0,1\}^* \cup \{\perp\}$.

3.**Forgery**: $\mathcal{A}$ outputs its forgery, ($\mathfrak{M}^*$, σ^*) for $Infor^*$ and $RL_{pk^*}^t$, where the identity $Infor^*$ and the revocation list $RL_{pk^*}^t$ are arbitrary forgeries generated by $\mathcal{A}$. It succeeds if

(a) $1 \leftarrow$***Verify***(GK, $\mathfrak{M}^*$, $Infor^*$, σ^*, $RL_{pk^*}^t$);

(b) $\mathcal{A}$ did not query ***Group-Setup*** on input $Infor^*$, did not query ***Join-User*** on inputs sk_g^* and pk^*, and did not query ***Sign*** on inputs csk^* and $\mathfrak{M}^*$, where the public key pk^* belongs to the group named by the identity $Infor^*$.

Definition 4.2 Traceability of A Group Signature Scheme: *Let* **GS**=*(**System-Setup**, **Generate-Key**, **Group-Setup**, **Join-User**, **Revoke-User**, **Sign**, **Verify**, **Trace-User**) be a group signature scheme, which meets the requirement of unforgeability.* **GS** *is traceable if the following conditions can be satisfied*:

1) *For all valid generated GK ←**System-Setup**(1^k), sk_g ←**Group-Setup**(GK, $Infor$), csk_i ← **Join-User**(GK, sk_g, pk_i) with $i \in \{0,1\}$, then σ_0 =**Sign**(GK, csk_0, $\mathfrak{M}$) and σ_1 =**Sign**(GK, csk_1, $\mathfrak{M}$), the outputs of **Trace-User**(GK, $\mathfrak{M}$, $Infor$, sk_g, σ_0, RL^t_{pk}) and **Trace-User**(GK, $\mathfrak{M}$, $Infor$, sk_g, σ_1, RL^t_{pk}) are distinguishable in polynomially many times.*
2) *We set that k is a secure parameter, and **Pr**($\mathcal{B}_{TM\_GS}$(k,$\mathcal{A}$)=1) is the probability that the algorithm $\mathcal{B}_{TM\_GS}$ returns 1, and that **Pr**($\mathcal{B}_{TF\_GS}$(k,$\mathcal{A}$)=1) is the probability that the algorithm $\mathcal{B}_{TF\_GS}$ returns 1. Then the advantage that the adversary $\mathcal{A}$ breaks* **GS** *is defined as follows:*

$$\mathbf{Adv}_{GS}^{t\_gs-mf}(k, q_g, q_j, q_r, q_s, \hbar)=\mathbf{Pr}(\mathcal{B}_{tm\_gs}(k,\mathcal{A})=1)||\mathbf{Pr}(\mathcal{B}_{tf\_gs}(k,\mathcal{A})=1),$$

where q_g is the maximal number of "Group-Setup" oracle queries, q_j is the maximal number of "Join-User" oracle queries, q_r is the maximal number of "Revoke-User" oracle queries, q_s is the maximal number of "Sign" oracle queries and $\hbar$ is the running time of $\mathcal{B}$. If the advantage that the adversary breaks **GS** *is negligible, then the scheme* **GS** *is secure.*

According to the Definition 4.2, the algorithm $\mathcal{B}_{TM\_GS}$ is described as follows:

1.**Setup**: Running ***System-Setup***, GK ←***System-Setup***(1^k), and then GK is passed to $\mathcal{A}$.

2.**Queries**: $\mathcal{A}$ makes queries to the following oracles for polynomially many times:

- ***Join-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$) and the public key $pk_{u_i^a}$ of the group member u_i^a, the oracle returns a group member private key $csk_{u_i^a}$ to $\mathcal{A}$, where sk_g is a group private key on the identity $Infor$ of the group and the user (group member) u_i^a is added to the set U^a.
- ***Revoke-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$), the public key $pk_{u_i^a}$ of the revoked group member u_i^a and the revocation list RL^t_{pk} of the last duration t, the oracle returns an updated revocation list RL^{t+1}_{pk}.
- ***Sign***(): Given the public parameters GK, the group member private key $csk_{u_i^a}$ (or the public

key $pk_{u_i^a}$) and the message $\mathfrak{M}$, the oracle returns a signature σ to $\mathcal{A}$, where $\sigma \in \{0,1\}^* \cup \{\perp\}$, and the user u_i^a is added to the set U^a if $u_i^a \notin U^a$.

3.**Forgery**: $\mathcal{A}$ outputs its forgery, ($\mathfrak{M}^*$, σ^*) for $Infor^*$ and $RL_{pk^*}^t$, where the identity $Infor^*$ and the revocation list $RL_{pk^*}^t$ are arbitrary forgeries generated by $\mathcal{A}$. It succeeds if

(a) $1 \leftarrow$***Verify***(GK, $\mathfrak{M}^*$, $Infor^*$, σ^*, $RL_{pk^*}^t$);

(b) $\mathcal{A}$ did not query ***Join-User*** on inputs sk_g^* and pk^*, did not query ***Revoke-User*** on inputs sk_g^*, pk^* and $RL_{pk^*}^{t-1}$, and did not query ***Sign*** on inputs csk^* and $\mathfrak{M}^*$, where the public key pk^* of the user u_{pk^*} belongs to the group named by the identity $Infor^*$ and $u_{pk^*} \notin U^a \setminus \{u_{pk_i}^a \mid pk_i \in RL_{pk^*}^t\}$;

(c) $pk^* \leftarrow$***Trace-User***(GK, $\mathfrak{M}^*$, $Infor^*$, sk_g^*, σ^*, $RL_{pk^*}^t$).

And then the algorithm $\mathcal{B}_{TF\_GS}$ is described as follows:

1.**Setup**: Running ***System-Setup***, $GK \leftarrow$***System-Setup***(1^k), and then GK is passed to $\mathcal{A}$.

2.**Queries**: $\mathcal{A}$ makes queries to the following oracles for polynomially many times:

- ***Group-Setup***(): Given the public parameters GK and the identity $Infor$ of the group, the oracle returns a group private key sk_g to $\mathcal{A}$.
- ***Join-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$) and the public key $pk_{u_i^b}$ of the group member u_i^b, the oracle returns a group member private key $csk_{u_i^b}$ to $\mathcal{A}$, where sk_g is a group private key on the identity $Infor$ of the group and the user (group member) u_i^b is added to the set U^b where $U^b \neq \varnothing$.
- ***Revoke-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$), the public key $pk_{u_i^b}$ of the revoked group member u_i^b and the revocation list RL_{pk}^t of the last duration t, the oracle returns an updated revocation list RL_{pk}^{t+1}.
- ***Sign***(): Given the public parameters GK, the group member private key $csk_{u_i^b}$ (or the public key $pk_{u_i^b}$) and the message $\mathfrak{M}$, the oracle returns a signature σ to $\mathcal{A}$, where $\sigma \in \{0,1\}^* \cup \{\perp\}$, and the user u_i^b is added to the set U^b if $u_i^b \notin U^b$.

3.**Forgery**: $\mathcal{A}$ outputs its forgery, ($\mathfrak{M}^*$, σ^*) for $Infor^*$ and $RL_{pk^*}^t$, where the identity $Infor^*$ and the revocation list $RL_{pk^*}^t$ are arbitrary forgeries generated by $\mathcal{A}$. It succeeds if

(a) $1 \leftarrow$***Verify***(GK, $\mathfrak{M}^*$, $Infor^*$, σ^*, $RL_{pk^*}^t$);

(b) $\mathcal{A}$ did not query ***Group-Setup*** on input $Infor^*$, did not query ***Join-User*** on inputs sk_g^* and pk^*, did not query ***Revoke-User*** on inputs sk_g^*, pk^* and $RL_{pk^*}^{t-1}$, and did not query

Sign on inputs csk^* and $\mathfrak{M}^*$, where the public key pk^* of the user $u^b_{pk^*}$ belongs to the group named by the identity $Infor^*$ and $u^b_{pk^*} \in U^b$;

(c) $pk^* \leftarrow$***Trace-User***$(GK$, $\mathfrak{M}^*$, $Infor^*$, sk^*_g, σ^*, $RL^t_{pk^*})$.

Definition 4.3 Anonymity of A Group Signature Scheme: *Let* **GS**=*(**System-Setup**, **Generate-Key**, **Group-Setup**, **Join-User**, **Revoke-User**, **Sign**, **Verify**, **Trace-User**) be a group signature scheme. Additionally, we set that k is a secure parameter, and **Pr**($\mathcal{B}_{A\_GS}(k,\mathcal{A})$=1) is the probability that the algorithm $\mathcal{B}_{A\_GS}$ returns 1. Then the advantage that the adversary $\mathcal{A}$ breaks* **GS** *is defined as follows:*

$$\mathbf{Adv}^{a\_gs}_{GS}(k, q_g, q_j, q_r, q_s, \hbar)=|\mathbf{Pr}(\mathcal{B}_{a\_gs}(k,\mathcal{A})=1)-\tfrac{1}{2}|,$$

where q_g is the maximal number of "Group-Setup" oracle queries, q_j is the maximal number of "Join-User" oracle queries, q_r is the maximal number of "Revoke-User" oracle queries, q_s is the maximal number of "Sign" oracle queries and $\hbar$ is the running time of $\mathcal{B}$. If the advantage that the adversary breaks **GS** *is negligible, then the scheme* **GS** *is secure.*

According to the Definition 4.3, the algorithm $\mathcal{B}_{A\_GS}$ is described as follows:

1.**Setup**: Running ***System-Setup***, $GK \leftarrow$***System-Setup***(1^k), and then GK is passed to $\mathcal{A}$.

2.**Queries Phase 1**: $\mathcal{A}$ makes queries to the following oracles for polynomially many times:

- ***Group-Setup***(): Given the public parameters GK and the identity information $Infor$ of the group, the oracle returns a group private key sk_g to $\mathcal{A}$.
- ***Join-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$) and the public key pk_i of the group member, the oracle returns a group member private key csk_i to $\mathcal{A}$, where sk_g is a group private key on the identity $Infor$ of the group.
- ***Revoke-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$), the public key pk_i of the revoked group member and the revocation list RL^t_{pk} of the last duration t, the oracle returns an updated revocation list RL^{t+1}_{pk}.
- ***Sign***(): Given the public parameters GK, the group member private key csk_i (or the public key pk_i) and the message $\mathfrak{M}$, the oracle returns a signature σ to $\mathcal{A}$, where $\sigma \in \{0,1\}^* \cup \{\perp\}$.

3.**Challenge**: $\mathcal{A}$ sends to the challenger its forgeries ($\mathfrak{M}^*$, $Infor^*$, $RL^t_{pk^*}$) and two group member public keys pk^*_0 and pk^*_1 that belong to the group named by the group identity $Infor^*$. The forgeries satisfy the following conditions:

(a) $\mathcal{A}$ did not query ***Group-Setup*** on input $Infor^*$;

(b) $\mathcal{A}$ did not query ***Join-User*** on inputs $Infor^*$, pk_0^* (and pk_1^*);

(c) $\mathcal{A}$ did not query ***Revoke-User*** on inputs $Infor^*$, pk_0^* (and pk_1^*) and $RL_{pk^*}^{t-1}$.

The challenger picks a random bit $x \in \{0,1\}$, and then runs and outputs $\sigma^* \leftarrow$***Sign***(GK, csk_x^*, $\mathfrak{M}^*$) to $\mathcal{A}$.

4.**Queries Phase 2**: $\mathcal{A}$ makes queries to the following oracles for polynomially many times again:

- ***Group-Setup***(): Given the public parameters GK and the identity information $Infor$ of the group (where $Infor \neq Infor^*$), the oracle returns a group private key sk_g to $\mathcal{A}$.
- ***Join-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$) and the public key pk_i of the group member (where $sk_g \neq sk_g^*$ and $pk_i \notin \{pk_0^*, pk_1^*\}$), the oracle returns a group member private key csk_i to $\mathcal{A}$, where sk_g is a group private key on the identity $Infor$ of the group.
- ***Revoke-User***(): Given the public parameters GK, the group private key sk_g (or the identity $Infor$), the public key pk_i of the revoked group member and the revocation list RL_{pk}^t of the last duration t, the oracle returns an updated revocation list RL_{pk}^{t+1} (where $\mathcal{A}$ did not query ***Revoke-User*** on inputs sk_g^*, pk_0^* (and pk_1^*)).
- ***Sign***(): Given the public parameters GK, the group member private key csk_i (or the public key pk_i) and the message $\mathfrak{M}$, the oracle returns a signature σ to $\mathcal{A}$, where $\sigma \in \{0,1\}^* \cup \{\perp\}$.

5.**Guess**: $\mathcal{A}$ outputs a bit $x' \in \{0,1\}$ and succeeds if $x' = x$.

V. Group Signature Scheme Based on EDL Signature

Let **GS**=(***System-Setup***, ***Generate-Key***, ***Group-Setup***, ***Join-User***, ***Revoke-User***, ***Sign***, ***Verify***, ***Trace-User***) be a group signature scheme. In **GS**, all algorithms are described as follows:

1) **GS.*System-Setup***: The algorithm run by the trusted authority inputs a security parameter 1^k. Then, let $\mathbb{G}_1$ be group of prime order q and module p, and g be a generator of $\mathbb{G}_1$. The size of the group is determined by the security parameter. And four hash functions, $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_1 : \mathbb{G}_1 \to \mathbb{G}_1$, $H_2 : \mathbb{G}_1^4 \times \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_3 : \mathbb{G}_1^3 \times \{0,1\}^* \to \mathbb{Z}_q^*$ can be defined. Finally, the algorithm outputs the public parameters GK=($\mathbb{G}_1$, g, H_0, H_1, H_2, H_3).

2) **GS.*Generate-Key***: The algorithm run by a group member generates his public/private key pair (pk_l, sk_l) with $l \in \{1, 2......n\}$, where n is the maximal number of users in a

group. The algorithm randomly chooses $sk_l \in \mathbb{Z}_q^*$, and then computes $pk_l = g^{sk_l}$.

3) **GS.*Group-Setup***: The algorithm run by the trusted authority inputs (GK, $Infor \in \{0,1\}^*$), where $Infor$ is a group public identity information. The algorithm randomly chooses $d \in \mathbb{Z}_q^*$, computes and outputs a group private key $sk_g = d \cdot H_0(Infor)$ to a group manager, and then publishes the group public key $pk_g = g^d$.

4) **GS.*Join-User***: The algorithm run by the group manager inputs (GK, sk_g, pk_l), and then the following steps are finished:

   a) The algorithm run by the group manager randomly chooses $a \in \mathbb{Z}_q^*$, computes

   $$u_1 = g^a,\ h_1 = H_1(u_1),$$
   $$x_1 = h_1^{sk_g},\ v_1 = h_1^a,$$
   $$c_1 = H_2(u_1, x_1, v_1, pk_g, Infor),\ r = a + c_1 \cdot sk_g.$$

   The algorithm outputs a partial member private key $\delta = (x_1, c_1, r)$ to a group member whose public key is pk_l, and then saves the tuple (pk_l, u_1), where u_1 is used to trace the real signer.

   b) The algorithm run by a group member with the public key pk_l and the private key sk_l verifies the partial member private key $\delta = (x_1, c_1, r)$ by the following computations:

   $$u_1' = g^r \cdot (pk_g)^{-c_1 \cdot H_0(Infor)},$$
   $$h_1' = H_1(u_1'),\ v_1' = (h_1')^r \cdot (x_1)^{-c_1},$$
   $$c_1' = H_2(u_1', x_1, v_1', pk_g, Infor),$$

   and then checks $c_1' = c_1$. If the equation $c_1' = c_1$ is correct, the group member accepts δ, otherwise the group member requires that the group manager must resend δ. Finally, the algorithm computes and outputs the group member private key $csk_l = \{u_1', \delta = (x_1, c_1, r)\}$ to the group member, where $u_1' = u_1 = g^a$.

5) **GS.*Revoke-User***: The algorithm run by the group manager inputs (GK, sk_g, pk_l, RL_{pk}^t), where pk_l is the public key of the revoked user. The algorithm computes $rv_l = (pk_l)^{\frac{1}{c_1}}$, where rv_l is a credential on the corresponding public key pk_l. Finally, the algorithm outputs and adds a tuple $[pk_l, rv_l]$ to the revocation list RL_{pk}^t, and then an updated revocation list RL_{pk}^{t+1} is published by a secure approach.

6) **GS.*Sign***: A group member with the group member private key csk_l needs to sign a message $\mathfrak{M} \in \{0,1\}^*$. The algorithm run by the group member inputs (GK, csk_l, $\mathfrak{M}$), and then randomly chooses $k, f \in \mathbb{Z}_q^*$, computes[3]

$$u_2 = g^k \cdot (u_1')^f,\ h_2 = H_1(u_2),\ v_2 = h_2^{f \cdot r + k},$$
$$c_1'' = c_1 \cdot f,\ c_2 = H_3(u_2, v_2, pk_g, c_1'', \mathfrak{M}, Infor),$$
$$y = f \cdot r + c_2 \cdot f \cdot sk_l,$$
$$x_2 = sk_l \cdot f - \tfrac{k}{c_2},\ x_3 = g^k,\ x_4 = g^{sk_l \cdot f}.$$

Finally, the algorithm outputs a signature $\sigma = \{c_1'', c_2, x_2, x_3, x_4, y\}$.

7) **GS.*Verify***: The signature receivers verify a group signature σ. The algorithm run by a signature verifier inputs (GK, $\mathfrak{M}$, $Infor$, σ, RL_{pk}^t), and then the following steps are finished:

a) The algorithm computes the following equations:

$$u_2' = g^y \cdot (pk_g)^{-c_1'' \cdot H_0(Infor)} \cdot g^{-x_2 \cdot c_2},$$
$$h_2' = H_1(u_2'),\ v_2' = (h_2')^y \cdot (h_2')^{-x_2 \cdot c_2},$$
$$c_2' = H_3(u_2', v_2', pk_g, c_1'', \mathfrak{M}, Infor),$$

and then checks $c_2' = c_2$. If the equation $c_2' = c_2$ is correct, then the algorithm runs into the next step, otherwise the algorithm outputs the boolean value $reject$.

b) The algorithm finishes the following steps on the revocation list RL_{pk}^t:

- Check the equation $g^{x_2} = (x_3)^{-\frac{1}{c_2}} \cdot x_4$; if the equation is correct, then the algorithm continues, otherwise the algorithm outputs the boolean value $reject$;
- Compute the equation $u_2'' = g^y \cdot (pk_g)^{-c_1'' \cdot H_0(Infor)} \cdot x_3 \cdot (x_4)^{-c_2}$, then check the equation $u_2'' = u_2'$; if the equation is correct, then the algorithm continues, otherwise the algorithm outputs the boolean value $reject$;
- Compute $rv_l' = (rv_l)^{c_1'' \cdot c_2} = (pk_l)^{\frac{1}{c_1} \cdot c_1 \cdot f \cdot c_2} = (pk_l)^{c_2 \cdot f} = g^{sk_l \cdot c_2 \cdot f}$, and $rv_l'' = g^{x_2 \cdot c_2} \cdot x_3 = g^{sk_l \cdot f \cdot c_2 - k} \cdot x_3 = g^{sk_l \cdot c_2 \cdot f}$, and then check $rv_l' = rv_l''$; if the equation $rv_l' = rv_l''$ is correct, then the algorithm directly outputs the boolean value $reject$; otherwise, if the algorithm does not find the correcting

[3] c_1'' may be also seen as $\{0,1\}^*$ in the computation of $H_3()$.

equation $rv'_l = rv''_l$ on the revocation list RL^t_{pk}, then the algorithm outputs the boolean value $accept$.

Remark: $rv'_l = rv''_l$ can denote whether the group member (signer) has been revoked.

8) **GS.*Trace-User***: The group manager traces a real group member (signer) on group signature σ, which can be verified by **GS.*Verify***. The algorithm run by the group manager computes the following equation:

$$[\frac{g^{c_1\cdot(y-x_2\cdot c_2)}}{(pk_g)^{c''_1\cdot c_1}\cdot(x_3)^{c_1}}]^{\frac{1}{c''_1}} = [\frac{g^{c_1\cdot(f\cdot r+k)}}{(pk_g)^{c''_1\cdot c_1}\cdot(x_3)^{c_1}}]^{\frac{1}{c''_1}} = [\frac{g^{c_1\cdot f\cdot(a+c_1\cdot sk_g)+c_1\cdot k}}{(pk_g)^{c''_1\cdot c_1}\cdot(x_3)^{c_1}}]^{\frac{1}{c''_1}} = [\frac{g^{c''_1\cdot a}\cdot g^{sk_g\cdot c''_1\cdot c_1}\cdot g^{k\cdot c_1}}{(pk_g)^{c''_1\cdot c_1}\cdot(x_3)^{c_1}}]^{\frac{1}{c''_1}}$$
$$= g^a = u_1.$$

Finally, the algorithm finds and outputs the corresponding public key pk_l by u_1.

VI. Analysis of the Proposed Scheme

A. Efficiency

In the proposed scheme, $\sigma = \{c''_1, c_2, x_2, x_3, x_4, y\}$, where

$$c''_1 = c_1 \cdot f,\ c_2 = H_3(u_2, v_2, pk_g, c''_1, \mathfrak{M}, Infor),$$
$$y = f\cdot r + c_2\cdot f\cdot sk_l,\ x_2 = sk_l\cdot f - \frac{k}{c_2},$$
$$x_3 = g^k \text{ and } x_4 = g^{sk_l\cdot f}.$$

Thus, the length of signature is $2\cdot|\mathbb{G}_1| + 4\cdot|\mathbb{Z}^*_q|$, where $|\mathbb{G}_1|$ is the size of element in $\mathbb{G}_1$ and $|\mathbb{Z}^*_q|$ is the size of element in $\mathbb{Z}^*_q$. Additionally, the signing and verifying procedure is mainly based on integer multiplication and hash computation, so if we assume that the time for integer multiplication and hash computation can be ignored, then signing a message for a group signature only needs to compute 5 exponentiations in $\mathbb{G}_1$ and 1 multiplication in $\mathbb{G}_1$, and verification requires at most $2\cdot L_r + 8$ exponentiations in $\mathbb{G}_1$ and $L_r + 6$ multiplications in $\mathbb{G}_1$, where L_r is the number of the revoked users in the revocation list RL^t_{pk}[4].

In this paper, we compare the proposed scheme (the scheme of Section 5) with the other group signature schemes [12,14,18,19,20]. Table 1 shows the comparisons of the schemes. Compared

[4]We only consider the bad thing that the revoked user is the last one in the revocation list when verification starts from the first one to the last one.

with other schemes, although our scheme is constructed in the random oracle model, our scheme does not employ pairing computation and has the constant signing time and signature size.

TABLE I
COMPARISONS OF THE SIX SCHEMES

	Signature Size	Signature Cost	Verification Cost	Model
Scheme [12]	$O(1)$	$O(1)$	$O(L_r)$	random oracle
Scheme [14]	$O(1)$	$O(1)$	$O(L_r)$	without random oracle
Scheme [18]	$O(1)$	$O(1)$	$O(1)$	without random oracle
Scheme [19]	$O(1)$	$O(L_m)$	$O(L_m + L_k)$	without random oracle
Scheme [20]	$O(1)$	$O(L_m)$	$O(1)$	random oracle
Our Scheme	$O(1)$	$O(1)$	$O(L_r)$	random oracle

caption: L_m is the length of signed message, L_k is the length of user identity, L_r is the number of revoked users in the revocation list.

B. Security

In the section, we show the proposed scheme (the scheme of Section 5) has the unforgeability, traceability and anonymity under the adaptive chosen message attacks, which can be reduced to the CDH assumption. Our proofs for the following theorems are based on the security models of Section 4 (We defer the proofs to Appendix A).

Theorem 6.1 *The scheme of Section 5 is ($\hbar$, ε, q_g, q_j, q_s)-unforgeable (according to the Definition 4.1), assuming that the ($\hbar'$, ε')-CDH assumption holds in $\mathbb{G}_1$, where:*

$$\varepsilon' = \varepsilon - \frac{q_g}{2^{n_q}} - q_j \cdot \left(\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}}\right) - \frac{q_s \cdot q_h}{2^{6 \cdot n_q}} - \frac{q_s \cdot (q_h + q_s)}{2^{n_q}},$$
$$\hbar' = \hbar + O((q_h + q_g + 4 \cdot q_j + 12 \cdot q_s) \cdot C_{exp} + 4 \cdot q_s \cdot C_{mul}),$$

and q_h is the maximal number of "Hash" oracle queries, q_g is the maximal number of "Group-Setup" oracle queries, q_j is the maximal number of "Join-User" oracle queries, q_s is the maximal number of "Sign" oracle queries, C_{mul} and C_{exp} are respectively the time for a multiplication and an exponentiation in $\mathbb{G}_1$.

Theorem 6.2 *The scheme of Section 5 is a traceable group signature scheme when it is unforgeable (Theorem 6.1 holds) and satisfies the following conditions (according to the Definition*

4.2):

a) *The outputs of "Trace-User" oracle are distinguishable in polynomially many times*;

b) *The scheme of Section 5 is ($\hbar''$, ε'', q_g, q_j, q_r, q_s)-secure, assuming that the ($\hbar'$, ε')-CDH assumption holds in $\mathbb{G}_1$, where:*

$$\varepsilon'' = [\varepsilon' + q_j \cdot (\tfrac{1}{2^{n_q}} + \tfrac{2\cdot q_h}{2^{n_q}}) + q_r \cdot (\tfrac{1}{2^{n_q}} + \tfrac{2\cdot q_h}{2^{n_q}}) + \tfrac{q_s\cdot q_h}{2^{6\cdot n_q}} + \tfrac{q_s\cdot(q_h+q_s)}{2^{n_q}}] \parallel$$
$$[\varepsilon' + \tfrac{q_g}{2^{n_q}} + q_j \cdot (\tfrac{1}{2^{n_q}} + \tfrac{2\cdot q_h}{2^{n_q}}) + q_r \cdot (\tfrac{1}{2^{n_q}} + \tfrac{2\cdot q_h}{2^{n_q}}) + \tfrac{q_s\cdot q_h}{2^{6\cdot n_q}} + \tfrac{q_s\cdot(q_h+q_s)}{2^{n_q}}],$$
$$\hbar'' = \mathbf{MAX}\{\hbar' - O((q_h + 4\cdot q_j + 5\cdot q_r + 12\cdot q_s)\cdot C_{exp} + 4\cdot q_s\cdot C_{mul}),\quad \hbar' - O((q_h + g_g + 4\cdot q_j + 5\cdot q_r + 12\cdot q_s)\cdot C_{exp} + 4\cdot q_s\cdot C_{mul})\}.$$

and q_h is the maximal number of "Hash" oracle queries, q_g is the maximal number of "Group-Setup" oracle queries, q_j is the maximal number of "Join-User" oracle queries, q_r is the maximal number of "Revoke-User" oracle queries, q_s is the maximal number of "Sign" oracle queries, C_{mul} and C_{exp} are respectively the time for a multiplication and an exponentiation in $\mathbb{G}_1$.

Theorem 6.3 *The scheme of Section 5 is ($\hbar$, ε, q_g, q_j, q_r, q_s)-anonymous (according to the Definition 4.3), assuming that the ($\hbar'$, ε')-CDH assumption holds in $\mathbb{G}_1$, where:*

$$\varepsilon' = \varepsilon - \tfrac{q_{g_1}+q_{g_2}}{2^{n_q}} - (q_{j_1}+q_{j_2})\cdot(\tfrac{1}{2^{n_q}}+\tfrac{2\cdot q_h}{2^{n_q}}) - (q_{r_1}+q_{r_2})\cdot(\tfrac{1}{2^{n_q}}+\tfrac{2\cdot q_h}{2^{n_q}}) - \tfrac{(q_{s_1}+q_{s_2})\cdot q_h}{2^{6\cdot n_q}} - \tfrac{(q_{s_1}+q_{s_2})\cdot(2\cdot q_h+q_{s_1}+q_{s_2})}{2^{n_q}},$$
$$\hbar' = \hbar + O((q_h+q_{g_1}+q_{g_2}+4\cdot(q_{j_1}+q_{j_2})+5\cdot(q_{r_1}+q_{r_2})+12\cdot(q_{s_1}+q_{s_2}))\cdot C_{exp}+4\cdot(q_{s_1}+q_{s_2})\cdot C_{mul}),$$

and q_h is the maximal number of "Hash" oracle queries, q_{g_1} and q_{g_2} are respectively the maximal numbers of "Group-Setup" oracle queries in the Queries Phase 1 and 2, q_{j_1} and q_{j_2} are respectively the maximal numbers of "Join-User" oracle queries in the Queries Phase 1 and 2, q_{r_1} and q_{r_2} are respectively the maximal numbers of "Revoke-User" oracle queries in the Queries Phase 1 and 2, q_{s_1} and q_{s_2} are respectively the maximal numbers of "Sign" oracle queries in the Queries Phase 1 and 2, C_{mul} and C_{exp} are respectively the time for a multiplication and an exponentiation in $\mathbb{G}_1$.

VII. Conclusions

In this paper, by modifying the EDL signature, we present a public key-based group signature scheme in the random oracle, which is based on the model of verifier-local revocation. Also, we give the security models for group signature. Under our security models, the proposed scheme is proved to have the properties of anonymity and traceability with enough security. Compared with other group signature schemes proposed by [12,14,18,19,20], the proposed group signature scheme does not employ pairing computation and has the constant signature size, so the proposed scheme is efficient. However, because the proposed scheme is not enough efficient in revoking verification of signatures, the work about group signature still needs to be further progressed.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No.61402055, No.61462048, No.61504013), and the Scientific Research Project of Hunan Provincial Education Department (No.15C0041, No.13B132, No.15A007, No.15C0779).

APPENDIX A
SECURITY PROOF

(**Proof of Theorem 6.1**).

Proof: Let **GS** be a group signature scheme of Section 5. Additionally, let $\mathcal{A}$ be an ($\hbar$, ε, q_g, q_j, q_s)-adversary attacking **GS**. From the adversary $\mathcal{A}$, we construct an algorithm $\mathcal{B}$, for (g, g^a, g^b)$\in \mathbb{G}_1$, the algorithm $\mathcal{B}$ is able to use $\mathcal{A}$ to compute $g^{a \cdot b}$. Thus, we assume the algorithm $\mathcal{B}$ can solve the CDH with probability at least ε' and in time at most $\hbar'$, contradicting the ($\hbar'$, ε')-CDH assumption. Such a simulation may be created in the following way:

Setup: The trusted authority system inputs a security parameter 1^k. Then, let $\mathbb{G}_1$ be group of prime order q and module p, and g be a generator of $\mathbb{G}_1$. The size of the group is determined by the security parameter. Also, $H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ can directly be computed on no querying. $H_1 : \mathbb{G}_1 \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_1^4 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_3 : \mathbb{G}_1^3 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ can be simulated by the algorithms H_1 Queries, H_2 Queries and H_3 Queries, where we set that g^b ($\mathcal{B}$ does not know b) is used to answer the query on H_1 Queries. Additionally, we assume that the user u^* is a challenger, whose public key is $pk^* = g^a$ ($\mathcal{B}$ does not know a where a is seen as the corresponding private key). Finally, the algorithm outputs the public parameters GK=($\mathbb{G}_1$, g, H_0).

Queries: When running the adversary $\mathcal{A}$, the relevant queries can occur according to the Definition 4.1. The algorithm $\mathcal{B}$ answers these in the following way:

- **H_1 Queries**: If this query is fresh, then the algorithm chooses random $s \in Z_q^*$, computes and outputs $(g^b)^s = g^{b \cdot s}$ to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple $(s, g^{b \cdot s})$ to U_List.

- **H_2 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **H_3 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **Group-Setup Queries**: Given the public parameters GK and the identity information $Infor$ of the group, the algorithm randomly chooses $d \in \mathbb{Z}_q^*$, computes and outputs a group private key $sk_g = d \cdot H_0(Infor)$ and a group public key $pk_g = g^d$ to $\mathcal{A}$.
- **Join-User Queries**: Given the public parameters GK and the group identity $Infor$, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d \cdot H_0(Infor)},\ v_1 = h_1^t,$$
$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor),\ r = t + c_1 \cdot d \cdot H_0(Infor).$$

  The algorithm outputs a partial member private key $\delta = (x_1, c_1, r)$ to $\mathcal{A}$. Because the algorithm does not know the private key of the queried group member, the algorithm only outputs a partial member private key to $\mathcal{A}$. However, the adversary $\mathcal{A}$ is easy to compute out the complete group member private key when the adversary $\mathcal{A}$ corrupted some group members or registered some controlled group member to the simulation system.
- **Sign Queries**: Given the public parameters GK, the identity information $Infor$ of the group, the public key pk_l and the message $\mathfrak{M}$, the following setups are finished:
  a) The algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),\ x_1 = h_1^{d \cdot H_0(Infor)},$$
$$v_1 = h_1^t,\ c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$

  b) The algorithm randomly chooses $c_2, y, f, k \in \mathbb{Z}_q^*$, computes

$$u_2 = g^y \cdot g^{-d \cdot c_1 \cdot f \cdot H_0(Infor)} \cdot g^{-k},$$

  and then queries the oracle **H_1 Queries** for u_2, if u_2 has been queried, then the algorithm aborts; otherwise the algorithm continues.

  c) The algorithm randomly chooses $j \in \mathbb{Z}_q^*$, computes

$$v_2 = h_2^y \cdot g^{-k \cdot j},$$

  where we set $h_2 = H_1(u_2) = g^j$ (satisfy the condition that $DL_{h_2}((h_2)^k) = DL_g(g^k) = k$).

d) The algorithm queries the oracle **H_3 Queries**, if the tuple $(u_2, v_2, g^d, c_1{\cdot}f, \mathfrak{M}, Infor)$ has been queried, then the algorithm aborts; otherwise the algorithm continues.

e) The algorithm computes $x_2 = \frac{k}{c_2}$, $x_3 = g^{-k} \cdot (pk_l)^f$, $x_4 = (pk_l)^{\frac{f}{c_2}}$, and then outputs a group signature $\sigma = \{c_1'', c_2, x_2, x_3, x_4, y\}$ to the adversary $\mathcal{A}$, and saves the tuple (t, d, c_2, f, k) to S_List.

Forgery: If the algorithm $\mathcal{B}$ does not abort as a consequence of one of the queries above, the adversary $\mathcal{A}$ will, with probability at least ε, return a forgery ($\mathfrak{M}^*$, σ^*, $Infor^*$, $RL^t_{pk^*}$) for the challenger u^*, where the identity $Infor^*$ and the revocation list $RL^t_{pk^*}$ are arbitrary forgeries generated by $\mathcal{A}$. And the forgery satisfies the following condition:

(a) $1 \leftarrow$***Verify***$(GK$, $\mathfrak{M}^*$, $Infor^*$, σ^*, $RL^t_{pk^*})$;

(b) $\mathcal{A}$ did not query ***Group-Setup*** on input $Infor^*$, did not query ***Join-User*** on input $Infor^*$, and did not query ***Sign*** on inputs $Infor^*$, pk^* and $\mathfrak{M}^*$ where the public key pk^* of the challenger u^* belongs to the group named by the identity $Infor^*$.

Then, if the adversary $\mathcal{A}$ did not query the oracle **H_1 Queries**, or U_List is empty or S_List is empty, then the algorithm $\mathcal{B}$ aborts.

Otherwise, the algorithm $\mathcal{B}$ can get $h_2 = H_1(*) = g^{b \cdot s}$. So, when the condition $DL_{h_2}((h_2)^{a \cdot f \cdot c_2 - k}) = DL_g(g^{a \cdot f \cdot c_2 - k}) = a \cdot f \cdot c_2 - k$ holds, we can get the followings:

$$h_2^{x_2 \cdot c_2} = (h_2)^{(a \cdot f - \frac{k}{c_2}) \cdot c_2} = (g^{b \cdot s})^{(a \cdot f - \frac{k}{c_2}) \cdot c_2} = (g^{b \cdot s})^{(a \cdot f \cdot c_2 - k)} = g^{a \cdot b \cdot s \cdot f \cdot c_2 - b \cdot s \cdot k},$$

then $\mathcal{B}$ computes and outputs $(h_2^{x_2 \cdot c_2} \cdot g^{b \cdot s \cdot k})^{\frac{1}{c_2 \cdot s \cdot f}} = g^{a \cdot b}$, which is the solution to the given CDH problem.

Now, we analyze the probability of the algorithm $\mathcal{B}$ not aborting. For the simulation to complete without aborting, we require that all ***Group-Setup*** queries and all ***Join-User*** queries are fresh, and all ***Sign*** queries do not abort. So, if the algorithm $\mathcal{B}$ does not abort, then the following conditions must hold:

(a) All ***Group-Setup*** queries are fresh, because $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$ is uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_0 is $\frac{1}{2^{n_q}}$, then the failure probability of the queries is at most $\frac{q_g}{2^{n_q}}$.

(b) All ***Join-User*** queries are fresh, similarly the collision probability of H_0 is $\frac{1}{2^{n_q}}$, and because $t, d \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$ and the collision probability of H_2 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$, then the failure probability of the queries is at most $q_j \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}})$.

(c) All ***Sign*** queries do not abort, then we may get the followings:

- The algorithm may abort in the setup b), namely u_2 has been queried on the oracle **H_1 Queries**. So, as $t, d, c_2, y, f, k \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q^6$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{6 \cdot n_q}} = \frac{q_h}{2^{6 \cdot n_q}}$, then the failure probability of the queries is at most $\frac{q_s \cdot q_h}{2^{6 \cdot n_q}}$;
- The algorithm may abort in the setup d), namely the tuple $(u_2, v_2, g^d, c_1 \cdot f, \mathfrak{M}, Infor)$ has been queried on the oracle **H_3 Queries**. So, as $j \in \mathbb{Z}_q^*$ is uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_3 is $(q_h + q_s) \cdot \frac{1}{2^{n_q}} = \frac{q_h + q_s}{2^{n_q}}$, then the failure probability of the queries is at most $\frac{q_s \cdot (q_h + q_s)}{2^{n_q}}$.

Therefore, from the above analysis, we get that the algorithm $\mathcal{B}$ can compute $g^{a \cdot b}$ from the forgery as shown above, with probability at least $\varepsilon' = \varepsilon - \frac{q_g}{2^{n_q}} - q_j \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}}) - \frac{q_s \cdot q_h}{2^{6 \cdot n_q}} - \frac{q_s \cdot (q_h + q_s)}{2^{n_q}}$. The time complexity of the algorithm $\mathcal{B}$ is $\hbar' = \hbar + O((q_h + q_g + 4 \cdot q_j + 12 \cdot q_s) \cdot C_{exp} + 4 \cdot q_s \cdot C_{mul})$, where we assume that the time for integer addition, integer multiplication and hash computation can both be ignored.

Thus, Theorem 6.1 follows.

(**Proof of Theorem 6.2**).

Proof: According to the Definition 4.2, we need to divide the proof to the following three parts:

1) *Correctness (the outputs of "Trace-User" oracle are distinguishable)*:

From the algorithm **GS.*Trace-User***, we may know that

$$[\frac{g^{c_1 \cdot (y - x_2 \cdot c_2)}}{(pk_g)^{c_1'' \cdot c_1} \cdot (x_3)^{c_1}}]^{\frac{1}{c_1''}} = [\frac{g^{c_1 \cdot (f \cdot r + k)}}{(pk_g)^{c_1'' \cdot c_1} \cdot (x_3)^{c_1}}]^{\frac{1}{c_1''}} = [\frac{g^{c_1 \cdot f \cdot (a + c_1 \cdot sk_g) + c_1 \cdot k}}{(pk_g)^{c_1'' \cdot c_1} \cdot (x_3)^{c_1}}]^{\frac{1}{c_1''}} = [\frac{g^{c_1'' \cdot a} \cdot g^{sk_g \cdot c_1'' \cdot c_1} \cdot g^{k \cdot c_1}}{(pk_g)^{c_1'' \cdot c_1} \cdot (x_3)^{c_1}}]^{\frac{1}{c_1''}}$$
$$= g^a = u_1.$$

So, for any potential public key pk_l, the algorithm **GS.*Trace-User*** run by the group manager can find and output the corresponding public key pk_l by u_1.

2) *Misidentification Attacks*:

Let **GS** be a group signature scheme of Section 5. Additionally, let $\mathcal{A}$ be an ($\hbar$, ε, q_j, q_r, q_s)-adversary attacking **GS**. From the adversary $\mathcal{A}$, we construct an algorithm $\mathcal{B}$, for (g, g^a, g^b)$\in \mathbb{G}_1$, the algorithm $\mathcal{B}$ is able to use $\mathcal{A}$ to compute $g^{a \cdot b}$. Thus, we assume the algorithm $\mathcal{B}$ can solve the CDH with probability at least ε' and in time at most $\hbar'$, contradicting the ($\hbar'$, ε')-CDH assumption. Such a simulation may be created in the following way:

Setup: The trusted authority system inputs a security parameter 1^k. Then, let $\mathbb{G}_1$ be group of prime order q and module p, and g be a generator of $\mathbb{G}_1$. The size of the group is determined by the security parameter. Also, $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$ can directly be computed on no querying. $H_1 : \mathbb{G}_1 \to \mathbb{G}_1$, $H_2 : \mathbb{G}_1^4 \times \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_3 : \mathbb{G}_1^3 \times \{0,1\}^* \to \mathbb{Z}_q^*$ can be simulated by the algorithms H_1 Queries, H_2 Queries and H_3 Queries, where we set that g^b ($\mathcal{B}$ does not know b) is used to answer the query on H_1 Queries. Additionally, we assume that the user u^* is a challenger, whose public key is $pk^* = g^a$ ($\mathcal{B}$ does not know a where a is seen as the corresponding private key). Finally, the algorithm outputs the public parameters GK=($\mathbb{G}_1$, g, H_0).

Queries: When running the adversary $\mathcal{A}$, the relevant queries can occur according to the algorithm $\mathcal{B}_{TM\_GS}$ of the Definition 4.2. The algorithm $\mathcal{B}$ answers these in the following way:

- **H_1 Queries**: If this query is fresh, then the algorithm chooses random $s \in Z_q^*$, computes and outputs $(g^b)^s = g^{b \cdot s}$ to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple $(s, g^{b \cdot s})$ to U_List.
- **H_2 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **H_3 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **Join-User Queries**: Given the public parameters GK and the group identity $Infor$, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d \cdot H_0(Infor)},\ v_1 = h_1^t,$$
$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor),\ r = t + c_1 \cdot d \cdot H_0(Infor).$$

The algorithm outputs a partial member private key $\delta = (x_1, c_1, r)$ to $\mathcal{A}$. Similarly, because the algorithm does not know the private key of the queried group member, the algorithm only outputs a partial member private key to $\mathcal{A}$. However, the adversary $\mathcal{A}$ is easy to compute out the complete group member private key when the adversary $\mathcal{A}$ corrupted some group members or registered some controlled group member to the simulation system, and the users (group members) u_i^a are added to the set U^a.

- **Revoke-User Queries**: Given the public parameters GK, the group identity $Infor$, the public key $pk_{u_i^a}$ of the revoked group member and the revocation list RL_{pk}^t of the last duration t, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes
$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d \cdot H_0(Infor)},\ v_1 = h_1^t,$$
$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$
Then, the algorithm computes $rv_{u_i^a} = (pk_{u_i^a})^{\frac{1}{c_1}}$, where $rv_{u_i^a}$ is a credential on the corresponding public key $pk_{u_i^a}$. Finally, the algorithm outputs and adds a tuple $[pk_{u_i^a}, rv_{u_i^a}]$ to the revocation list RL_{pk}^t, and then an updated revocation list RL_{pk}^{t+1} is published to the adversary $\mathcal{A}$.
- **Sign Queries**: Given the public parameters GK, the identity information $Infor$ of the group, the public key $pk_{u_i^a}$ and the message $\mathfrak{M}$, the following setups are finished:

a) The algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes
$$u_1 = g^t,\ h_1 = H_1(u_1),\ x_1 = h_1^{d \cdot H_0(Infor)},$$
$$v_1 = h_1^t,\ c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$

b) The algorithm randomly chooses $c_2, y, f, k \in \mathbb{Z}_q^*$, computes
$$u_2 = g^y \cdot g^{-d \cdot c_1 \cdot f \cdot H_0(Infor)} \cdot g^{-k},$$
and then queries the oracle **H_1 Queries** for u_2, if u_2 has been queried, then the algorithm aborts; otherwise the algorithm continues.

c) The algorithm randomly chooses $j \in \mathbb{Z}_q^*$, computes
$$v_2 = h_2^y \cdot g^{-k \cdot j},$$
where we set $h_2 = H_1(u_2) = g^j$ (satisfy the condition that $DL_{h_2}((h_2)^k) = DL_g(g^k) = k$).

d) The algorithm queries the oracle **H_3 Queries**, if the tuple $(u_2, v_2, g^d, c_1 \cdot f, \mathfrak{M}, Infor)$ has been queried, then the algorithm aborts; otherwise the algorithm continues.

e) The algorithm computes $x_2 = \frac{k}{c_2}$, $x_3 = g^{-k} \cdot (pk_{u_i^a})^f$, $x_4 = (pk_{u_i^a})^{\frac{f}{c_2}}$, and then outputs a group signature $\sigma = \{c_1'', c_2, x_2, x_3, x_4, y\}$ to the adversary $\mathcal{A}$, saves the tuple (t, d, c_2, f, k) to S_List, and the user u_i^a is added to the set U^a if $u_i^a \notin U^a$.

Forgery: If the algorithm $\mathcal{B}$ does not abort as a consequence of one of the queries above, the adversary $\mathcal{A}$ will, with probability at least ε, return a forgery ($\mathfrak{M}^*$, σ^*, $Infor^*$, $RL_{pk^*}^t$) for the challenger u^*, where the identity $Infor^*$ and the revocation list $RL_{pk^*}^t$ are arbitrary forgeries generated by $\mathcal{A}$. And the forgery satisfies the following condition:

(a) $1 \leftarrow$***Verify***$(GK$, $\mathfrak{M}^*$, $Infor^*$, σ^*, $RL_{pk^*}^t)$;

(b) $\mathcal{A}$ did not query ***Join-User*** on input $Infor^*$, did not query ***Revoke-User*** on inputs $Infor^*$, pk^* and $RL_{pk^*}^{t-1}$, and did not query ***Sign*** on inputs $Infor^*$, pk^* and $\mathfrak{M}^*$, where the public key pk^* belongs to the group named by the identity $Infor^*$ and $u^* \notin U^a \setminus \{u_{pk_i}^a \mid pk_i \in RL_{pk^*}^t\}$;

(c) $pk^* \leftarrow$***Trace-User***$(GK$, $\mathfrak{M}^*$, $Infor^*$, $*$, σ^*, $RL_{pk^*}^t)$.

Then, if the adversary $\mathcal{A}$ did not query the oracle **H_1 Queries**, or U_List is empty or S_List is empty, then the algorithm $\mathcal{B}$ aborts.

Otherwise, the algorithm $\mathcal{B}$ can get $h_2 = H_1(*) = g^{b \cdot s}$. So, when the condition $DL_{h_2}((h_2)^{a \cdot f \cdot c_2 - k}) = DL_g(g^{a \cdot f \cdot c_2 - k}) = a \cdot f \cdot c_2 - k$ holds, we can get the followings:

$$h_2^{x_2 \cdot c_2} = (h_2)^{(a \cdot f - \frac{k}{c_2}) \cdot c_2} = (g^{b \cdot s})^{(a \cdot f - \frac{k}{c_2}) \cdot c_2} = (g^{b \cdot s})^{(a \cdot f \cdot c_2 - k)} = g^{a \cdot b \cdot s \cdot f \cdot c_2 - b \cdot s \cdot k},$$

then $\mathcal{B}$ computes and outputs $(h_2^{x_2 \cdot c_2} \cdot g^{b \cdot s \cdot k})^{\frac{1}{c_2 \cdot s \cdot f}} = g^{a \cdot b}$, which is the solution to the given CDH problem.

Now, we analyze the probability of the algorithm $\mathcal{B}$ not aborting. For the simulation to complete without aborting, we require that all ***Join-User*** queries and all ***Revoke-User*** queries are fresh, and all ***Sign*** queries do not abort. So, if the algorithm $\mathcal{B}$ does not abort, then the following conditions must hold:

(a) All ***Join-User*** queries are fresh, the collision probability of H_0 is $\frac{1}{2^{n_q}}$, and because $t, d \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$ and the collision probability of H_2 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$, then the failure probability of the queries is at most $q_j \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}})$.

(b) All ***Revoke-User*** queries are fresh, similarly the collision probability of H_0 is $\frac{1}{2^{n_q}}$, and because $t, d \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$ and the collision probability of H_2 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$, then the failure probability of the queries is at most $q_r \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}})$.

(c) All ***Sign*** queries do not abort, then we may get the followings:

- The algorithm may abort in the setup b), namely u_2 has been queried on the oracle **H_1 Queries**. So, as $t, d, c_2, y, f, k \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q^6$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{6 \cdot n_q}} = \frac{q_h}{2^{6 \cdot n_q}}$, then the failure probability of the queries is at most $\frac{q_s \cdot q_h}{2^{6 \cdot n_q}}$;
- The algorithm may abort in the setup d), namely the tuple $(u_2, v_2, g^d, c_1 \cdot f, \mathfrak{M}, Infor)$ has been queried on the oracle **H_3 Queries**. So, as $j \in \mathbb{Z}_q^*$ is uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_3 is $(q_h + q_s) \cdot \frac{1}{2^{n_q}} = \frac{q_h + q_s}{2^{n_q}}$, then the failure probability of the queries is at most $\frac{q_s \cdot (q_h + q_s)}{2^{n_q}}$.

Therefore, from the above analysis, we get that the algorithm $\mathcal{B}$ can compute $g^{a \cdot b}$ from the forgery as shown above, with probability at least $\varepsilon' = \varepsilon - q_j \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}}) - q_r \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}}) - \frac{q_s \cdot q_h}{2^{6 \cdot n_q}} - \frac{q_s \cdot (q_h + q_s)}{2^{n_q}}$. The time complexity of the algorithm $\mathcal{B}$ is $\hbar' = \hbar + O((q_h + 4 \cdot q_j + 5 \cdot q_r + 12 \cdot q_s) \cdot C_{exp} + 4 \cdot q_s \cdot C_{mul})$, where we assume that the time for integer addition, integer multiplication and hash computation can both be ignored.

3) *framing attacks*:

Let **GS** be a group signature scheme of Section 5. Additionally, let $\mathcal{A}$ be an ($\hbar$, ε, q_g, q_j, q_r, q_s)-adversary attacking **GS**. From the adversary $\mathcal{A}$, we construct an algorithm $\mathcal{B}$, for (g, g^a, g^b)$\in \mathbb{G}_1$, the algorithm $\mathcal{B}$ is able to use $\mathcal{A}$ to compute $g^{a \cdot b}$. Thus, we assume the algorithm $\mathcal{B}$ can solve the CDH with probability at least ε' and in time at most $\hbar'$, contradicting the ($\hbar'$, ε')-CDH assumption. Such a simulation may be created in the following way:

Setup: The trusted authority system inputs a security parameter 1^k. Then, let $\mathbb{G}_1$ be group of prime order q and module p, and g be a generator of $\mathbb{G}_1$. The size of the group is determined by the security parameter. Also, $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$ can directly be computed on no querying. $H_1 : \mathbb{G}_1 \to \mathbb{G}_1$, $H_2 : \mathbb{G}_1^4 \times \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_3 : \mathbb{G}_1^3 \times \{0,1\}^* \to \mathbb{Z}_q^*$ can be simulated by the algorithms H_1 Queries, H_2 Queries and

H_3 Queries, where we set that g^b ($\mathcal{B}$ does not know b) is used to answer the query on H_1 Queries. Additionally, we assume that the user u^* is a challenger, whose public key is $pk^* = g^a$ ($\mathcal{B}$ does not know a where a is seen as the corresponding private key). Finally, the algorithm outputs the public parameters GK=($\mathbb{G}_1$, g, H_0).

Queries: When running the adversary $\mathcal{A}$, the relevant queries can occur according to the algorithm $\mathcal{B}_{TF\_GS}$ of the Definition 4.2. The algorithm $\mathcal{B}$ answers these in the following way:

- **H_1 Queries**: If this query is fresh, then the algorithm chooses random $s \in Z_q^*$, computes and outputs $(g^b)^s = g^{b \cdot s}$ to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple $(s, g^{b \cdot s})$ to U_List.
- **H_2 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **H_3 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **Group-Setup Queries**: Given the public parameters GK and the identity information $Infor$ of the group, the algorithm randomly chooses $d \in \mathbb{Z}_q^*$, computes and outputs a group private key $sk_g = d \cdot H_0(Infor)$ and a group public key $pk_g = g^d$ to $\mathcal{A}$.
- **Join-User Queries**: Given the public parameters GK and the group identity $Infor$, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d \cdot H_0(Infor)},\ v_1 = h_1^t,$$
$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor),\ r = t + c_1 \cdot d \cdot H_0(Infor).$$

  The algorithm outputs a partial member private key $\delta = (x_1, c_1, r)$ to $\mathcal{A}$, which is related to the honest user u_i^b. And the user u_i^b is added to the set U^b where $U^b \neq \varnothing$.
- **Revoke-User Queries**: Given the public parameters GK, the group identity $Infor$, the public key $pk_{u_i^b}$ of the revoked group member and the revocation list RL_{pk}^t of the last duration t, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d \cdot H_0(Infor)},\ v_1 = h_1^t,$$

$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$

Then, the algorithm computes $rv_{u_i^b} = (pk_{u_i^b})^{\frac{1}{c_1}}$, where $rv_{u_i^b}$ is a credential on the corresponding public key $pk_{u_i^b}$. Finally, the algorithm outputs and adds a tuple $[pk_{u_i^b}, rv_{u_i^b}]$ to the revocation list RL^t_{pk}, and then an updated revocation list RL^{t+1}_{pk} is published to the adversary $\mathcal{A}$.

- **Sign Queries**: Given the public parameters GK, the identity information $Infor$ of the group, the public key $pk_{u_i^b}$ and the message $\mathfrak{M}$, the following setups are finished:

  a) The algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes
  $$u_1 = g^t,\ h_1 = H_1(u_1),\ x_1 = h_1^{d \cdot H_0(Infor)},$$
  $$v_1 = h_1^t,\ c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$

  b) The algorithm randomly chooses $c_2, y, f, k \in \mathbb{Z}_q^*$, computes
  $$u_2 = g^y \cdot g^{-d \cdot c_1 \cdot f \cdot H_0(Infor)} \cdot g^{-k},$$
  and then queries the oracle **H_1 Queries** for u_2, if u_2 has been queried, then the algorithm aborts; otherwise the algorithm continues.

  c) The algorithm randomly chooses $j \in \mathbb{Z}_q^*$, computes
  $$v_2 = h_2^y \cdot g^{-k \cdot j},$$
  where we set $h_2 = H_1(u_2) = g^j$ (satisfy the condition that $DL_{h_2}((h_2)^k) = DL_g(g^k) = k$).

  d) The algorithm queries the oracle **H_3 Queries**, if the tuple $(u_2, v_2, g^d, c_1 \cdot f, \mathfrak{M}, Infor)$ has been queried, then the algorithm aborts; otherwise the algorithm continues.

  e) The algorithm computes $x_2 = \frac{k}{c_2}$, $x_3 = g^{-k} \cdot (pk_{u_i^b})^f$, $x_4 = (pk_{u_i^b})^{\frac{f}{c_2}}$, and then outputs a group signature $\sigma = \{c_1'', c_2, x_2, x_3, x_4, y\}$ to the adversary $\mathcal{A}$, saves the tuple (t, d, c_2, f, k) to S_List, and the user u_i^b is added to the set U^b if $u_i^b \notin U^b$.

Forgery: If the algorithm $\mathcal{B}$ does not abort as a consequence of one of the queries above, the adversary $\mathcal{A}$ will, with probability at least ε, return a forgery ($\mathfrak{M}^*$, σ^*, $Infor^*$, $RL^t_{pk^*}$) for the challenger u^*, where the identity $Infor^*$ and the revocation list $RL^t_{pk^*}$ are arbitrary forgeries generated by $\mathcal{A}$. And the forgery satisfies the following condition:

(a) $1 \leftarrow$***Verify***$(GK, \mathfrak{M}^*, Infor^*, \sigma^*, RL^t_{pk^*})$;

(b) $\mathcal{A}$ did not query ***Group-Setup*** on input $Infor^*$, did not query ***Join-User*** on input $Infor^*$, did not query ***Revoke-User*** on inputs $Infor^*$, pk^* and $RL^{t-1}_{pk^*}$, and did not query ***Sign*** on inputs $Infor^*$, pk^* and $\mathfrak{M}^*$, where the public key pk^* belongs to the group named by the identity $Infor^*$ and $u^* \in U^b$;

(c) $pk^* \leftarrow$***Trace-User***$(GK, \mathfrak{M}^*, Infor^*, *, \sigma^*, RL^t_{pk^*})$.

Then, if the adversary $\mathcal{A}$ did not query the oracle **H_1 Queries**, or U_List is empty or S_List is empty, then the algorithm $\mathcal{B}$ aborts.

Otherwise, the algorithm $\mathcal{B}$ can get $h_2 = H_1(*) = g^{b\cdot s}$. So, when the condition $DL_{h_2}((h_2)^{a\cdot f\cdot c_2 - k}) = DL_g(g^{a\cdot f\cdot c_2 - k}) = a \cdot f \cdot c_2 - k$ holds, we can get the followings:

$$h_2^{x_2\cdot c_2} = (h_2)^{(a\cdot f - \frac{k}{c_2})\cdot c_2} = (g^{b\cdot s})^{(a\cdot f - \frac{k}{c_2})\cdot c_2} = (g^{b\cdot s})^{(a\cdot f\cdot c_2 - k)} = g^{a\cdot b\cdot s\cdot f\cdot c_2 - b\cdot s\cdot k},$$

then $\mathcal{B}$ computes and outputs $(h_2^{x_2\cdot c_2} \cdot g^{b\cdot s\cdot k})^{\frac{1}{c_2\cdot s\cdot f}} = g^{a\cdot b}$, which is the solution to the given CDH problem.

Now, we analyze the probability of the algorithm $\mathcal{B}$ not aborting. For the simulation to complete without aborting, we require that all ***Group-Setup*** queries, all ***Join-User*** queries and all ***Revoke-User*** queries are fresh, and all ***Sign*** queries do not abort. So, if the algorithm $\mathcal{B}$ does not abort, then the following conditions must hold:

(a) All ***Group-Setup*** queries are fresh, because $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$ is uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_0 is $\frac{1}{2^{n_q}}$, then the failure probability of the queries is at most $\frac{q_g}{2^{n_q}}$.

(b) All ***Join-User*** queries are fresh, the collision probability of H_0 is $\frac{1}{2^{n_q}}$, and because $t, d \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$ and the collision probability of H_2 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$, then the failure probability of the queries is at most $q_j \cdot (\frac{1}{2^{n_q}} + \frac{2\cdot q_h}{2^{n_q}})$.

(c) All ***Revoke-User*** queries are fresh, similarly the collision probability of H_0 is $\frac{1}{2^{n_q}}$, and because $t, d \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$ and the collision probability of H_2 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$, then the failure probability of the queries is at most $q_r \cdot (\frac{1}{2^{n_q}} + \frac{2\cdot q_h}{2^{n_q}})$.

(d) All ***Sign*** queries do not abort, then we may get the followings:

- The algorithm may abort in the setup b), namely u_2 has been queried on the oracle **H_1 Queries**. So, as $t, d, c_2, y, f, k \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q^6$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{6 \cdot n_q}} = \frac{q_h}{2^{6 \cdot n_q}}$, then the failure probability of the queries is at most $\frac{q_s \cdot q_h}{2^{6 \cdot n_q}}$;
- The algorithm may abort in the setup d), namely the tuple $(u_2, v_2, g^d, c_1 \cdot f, \mathfrak{M}, Infor)$ has been queried on the oracle **H_3 Queries**. So, as $j \in \mathbb{Z}_q^*$ is uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_3 is $(q_h+q_s)\cdot\frac{1}{2^{n_q}} = \frac{q_h+q_s}{2^{n_q}}$, then the failure probability of the queries is at most $\frac{q_s\cdot(q_h+q_s)}{2^{n_q}}$.

Therefore, from the above analysis, we get that the algorithm $\mathcal{B}$ can compute $g^{a\cdot b}$ from the forgery as shown above, with probability at least $\varepsilon' = \varepsilon - \frac{q_g}{2^{n_q}} - q_j \cdot (\frac{1}{2^{n_q}} + \frac{2\cdot q_h}{2^{n_q}}) - q_r \cdot (\frac{1}{2^{n_q}} + \frac{2\cdot q_h}{2^{n_q}}) - \frac{q_s\cdot q_h}{2^{6\cdot n_q}} - \frac{q_s\cdot(q_h+q_s)}{2^{n_q}}$. The time complexity of the algorithm $\mathcal{B}$ is $\hbar' = \hbar + O((q_h + g_g + 4\cdot q_j + 5\cdot q_r + 12\cdot q_s)\cdot C_{exp} + 4\cdot q_s\cdot C_{mul})$, where we assume that the time for integer addition, integer multiplication and hash computation can both be ignored.

So, from the above proofs, we may get that

$$\varepsilon'' = [\varepsilon' + q_j \cdot (\tfrac{1}{2^{n_q}} + \tfrac{2\cdot q_h}{2^{n_q}}) + q_r \cdot (\tfrac{1}{2^{n_q}} + \tfrac{2\cdot q_h}{2^{n_q}}) + \tfrac{q_s\cdot q_h}{2^{6\cdot n_q}} + \tfrac{q_s\cdot(q_h+q_s)}{2^{n_q}}] \parallel$$
$$[\varepsilon' + \tfrac{q_g}{2^{n_q}} + q_j \cdot (\tfrac{1}{2^{n_q}} + \tfrac{2\cdot q_h}{2^{n_q}}) + q_r \cdot (\tfrac{1}{2^{n_q}} + \tfrac{2\cdot q_h}{2^{n_q}}) + \tfrac{q_s\cdot q_h}{2^{6\cdot n_q}} + \tfrac{q_s\cdot(q_h+q_s)}{2^{n_q}}],$$
$$\hbar'' = \mathbf{MAX}\{\hbar' - O((q_h + 4\cdot q_j + 5\cdot q_r + 12\cdot q_s)\cdot C_{exp} + 4\cdot q_s\cdot C_{mul}),\quad \hbar' - O((q_h + g_g + 4\cdot q_j + 5\cdot q_r + 12\cdot q_s)\cdot C_{exp} + 4\cdot q_s\cdot C_{mul})\}.$$

Thus, Theorem 6.2 follows.

(**Proof of Theorem 6.3**).

Proof: Let **GS** be a group signature scheme of Section 5. Additionally, let $\mathcal{A}$ be an ($\hbar$, ε, q_g, q_j, q_r, q_s)-adversary attacking **GS**. From the adversary $\mathcal{A}$, we construct an algorithm $\mathcal{B}$, for (g, g^{a_0}, g^b) or (g, g^{a_1}, g^b) $\in \mathbb{G}_1$, the algorithm $\mathcal{B}$ is able to use $\mathcal{A}$ to compute $g^{a_0\cdot b}$ or $g^{a_1\cdot b}$. Thus, we assume the algorithm $\mathcal{B}$ can solve the CDH with probability at least ε' and in time at most $\hbar'$, contradicting the ($\hbar'$, ε')-CDH assumption. Such a simulation may be created in the following way:

Setup: The trusted authority system inputs a security parameter 1^k. Then, let $\mathbb{G}_1$ be group of prime order q and module p, and g be a generator of $\mathbb{G}_1$. The size of the group is determined by the security parameter. Also, $H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ can directly be computed on no querying. $H_1 : \mathbb{G}_1 \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_1^4 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_3 : \mathbb{G}_1^3 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ can be simulated by the algorithms H_1 Queries, H_2 Queries and H_3 Queries, where we set that g^b ($\mathcal{B}$ does not know b) is used to answer the query on H_1 Queries. Additionally, we assume that the users u_0^* and u_1^* are two challengers, whose public keys respectively are $pk_0^* = g^{a_0}$ and $pk_1^* = g^{a_1}$ ($\mathcal{B}$ does not know a_0 and a_1 where a_0 and a_1 are seen as the corresponding private keys). Finally, the algorithm outputs the public parameters GK=($\mathbb{G}_1$, g, H_0).

Queries Phase 1: When running the adversary $\mathcal{A}$, the relevant queries can occur according to the Definition 4.3. The algorithm $\mathcal{B}$ answers these in the following way:

- **H_1 Queries**: If this query is fresh, then the algorithm chooses random $s \in Z_q^*$, computes and outputs $(g^b)^s = g^{b \cdot s}$ to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple $(s, g^{b \cdot s})$ to U_List.
- **H_2 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **H_3 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **Group-Setup Queries**: Given the public parameters GK and the identity information $Infor$ of the group, the algorithm randomly chooses $d \in \mathbb{Z}_q^*$, computes and outputs a group private key $sk_g = d \cdot H_0(Infor)$ and a group public key $pk_g = g^d$ to $\mathcal{A}$.
- **Join-User Queries**: Given the public parameters GK and the group identity $Infor$, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d \cdot H_0(Infor)},\ v_1 = h_1^t,$$
$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor),\ r = t + c_1 \cdot d \cdot H_0(Infor).$$

  The algorithm outputs a partial member private key $\delta = (x_1, c_1, r)$ to $\mathcal{A}$. Similarly, the adversary $\mathcal{A}$ is easy to compute out the complete group member private key when the adversary $\mathcal{A}$ corrupted some group members or registered some controlled group member to the simulation system.
- **Revoke-User Queries**: Given the public parameters GK, the group identity $Infor$, the public key pk_l of the revoked group member and the revocation list RL_{pk}^t of the last

duration t, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d \cdot H_0(Infor)},\ v_1 = h_1^t,$$
$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$

Then, the algorithm computes $rv_l = (pk_l)^{\frac{1}{c_1}}$, where rv_l is a credential on the corresponding public key pk_l. Finally, the algorithm outputs and adds a tuple $[pk_l, rv_l]$ to the revocation list RL_{pk}^t, and then an updated revocation list RL_{pk}^{t+1} is published to the adversary $\mathcal{A}$.

- **Sign Queries**: Given the public parameters GK, the identity information $Infor$ of the group, the public key pk_l and the message $\mathfrak{M}$, the following setups are finished:
  a) The algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes
  $$u_1 = g^t,\ h_1 = H_1(u_1),\ x_1 = h_1^{d \cdot H_0(Infor)},$$
  $$v_1 = h_1^t,\ c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$
  b) The algorithm randomly chooses $c_2, y, f, k \in \mathbb{Z}_q^*$, computes
  $$u_2 = g^y \cdot g^{-d \cdot c_1 \cdot f \cdot H_0(Infor)} \cdot g^{-k},$$
  and then queries the oracle **H_1 Queries** for u_2, if u_2 has been queried, then the algorithm aborts; otherwise the algorithm continues.
  c) The algorithm randomly chooses $j \in \mathbb{Z}_q^*$, computes
  $$v_2 = h_2^y \cdot g^{-k \cdot j},$$
  where we set $h_2 = H_1(u_2) = g^j$ (satisfy the condition that $DL_{h_2}((h_2)^k) = DL_g(g^k) = k$).
  d) The algorithm queries the oracle **H_3 Queries**, if the tuple $(u_2, v_2, g^d, c_1 \cdot f, \mathfrak{M}, Infor)$ has been queried, then the algorithm aborts; otherwise the algorithm continues.
  e) The algorithm computes $x_2 = \frac{k}{c_2}$, $x_3 = g^{-k} \cdot (pk_l)^f$, $x_4 = (pk_l)^{\frac{f}{c_2}}$, and then outputs a group signature $\sigma = \{c_1'', c_2, x_2, x_3, x_4, y\}$ to the adversary $\mathcal{A}$, and saves the tuple (t, d, c_2, f, k) to S_List.

Challenge: $\mathcal{A}$ sends to the challengers its forgeries ($\mathfrak{M}^*$, $Infor^*$, $RL_{pk^*}^t$) and two group member public keys pk_0^* and pk_1^* that belong to the group named by the group identity $Infor^*$. The forgeries satisfy the following conditions:

(a) $\mathcal{A}$ did not query ***Group-Setup*** on input $Infor^*$;

(b) $\mathcal{A}$ did not query ***Join-User*** on inputs $Infor^*$;

(c) $\mathcal{A}$ did not query ***Revoke-User*** on inputs $Infor^*$, pk_0^* (and pk_1^*) and $RL_{pk^*}^{t-1}$.

The challengers pick a random bit $x \in \{0,1\}$, and then run and output $\sigma^* \leftarrow$***Sign***$(GK$, csk_x^*, $\mathfrak{M}^*)$ to $\mathcal{A}$.

Queries Phase 2: When running the adversary $\mathcal{A}$, the relevant queries can occur according to the Definition 4.3. The algorithm $\mathcal{B}$ answers these in the following way:

- **H_1 Queries**: If this query is fresh, then the algorithm chooses random $s \in Z_q^*$, computes and outputs $(g^b)^s = g^{b\cdot s}$ to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple $(s, g^{b\cdot s})$ to U_List.
- **H_2 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **H_3 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.
- **Group-Setup Queries**: Given the public parameters GK and the identity information $Infor$ of the group, the algorithm randomly chooses $d \in \mathbb{Z}_q^*$, computes and outputs a group private key $sk_g = d \cdot H_0(Infor)$ and a group public key $pk_g = g^d$ to $\mathcal{A}$.
- **Join-User Queries**: Given the public parameters GK and the group identity $Infor$, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d\cdot H_0(Infor)},\ v_1 = h_1^t,$$
$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor),\ r = t + c_1 \cdot d \cdot H_0(Infor).$$

  The algorithm outputs a partial member private key $\delta = (x_1, c_1, r)$ to $\mathcal{A}$.
- **Revoke-User Queries**: Given the public parameters GK, the group identity $Infor$, the public key pk_l of the revoked group member and the revocation list RL_{pk}^t of the last duration t, the algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t,\ h_1 = H_1(u_1),$$
$$x_1 = h_1^{d\cdot H_0(Infor)},\ v_1 = h_1^t,$$
$$c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$

  Then, the algorithm computes $rv_l = (pk_l)^{\frac{1}{c_1}}$, where rv_l is a credential on the corresponding public key pk_l. Finally, the algorithm outputs and adds a tuple $[pk_l, rv_l]$ to the revocation list RL_{pk}^t, and then an updated revocation list RL_{pk}^{t+1} is published to the adversary $\mathcal{A}$, where

$\mathcal{A}$ did not query **Revoke-User Queries** on inputs $Infor^*$, pk_0^* (and pk_1^*).

- **Sign Queries**: Given the public parameters GK, the identity information $Infor$ of the group, the public key pk_l and the message $\mathfrak{M}$, the following setups are finished:
  a) The algorithm randomly chooses $t, d \in \mathbb{Z}_q^*$, computes
  $$u_1 = g^t,\ h_1 = H_1(u_1),\ x_1 = h_1^{d \cdot H_0(Infor)},$$
  $$v_1 = h_1^t,\ c_1 = H_2(u_1, x_1, v_1, g^d, Infor).$$
  b) The algorithm randomly chooses $c_2, y, f, k \in \mathbb{Z}_q^*$, computes
  $$u_2 = g^y \cdot g^{-d \cdot c_1 \cdot f \cdot H_0(Infor)} \cdot g^{-k},$$
  and then queries the oracle **H_1 Queries** for u_2, if u_2 has been queried, then the algorithm aborts; otherwise the algorithm continues.
  c) The algorithm randomly chooses $j \in \mathbb{Z}_q^*$, computes
  $$v_2 = h_2^y \cdot g^{-k \cdot j},$$
  where we set $h_2 = H_1(u_2) = g^j$ (satisfy the condition that $DL_{h_2}((h_2)^k) = DL_g(g^k) = k$).
  d) The algorithm queries the oracle **H_3 Queries**, if the tuple $(u_2, v_2, g^d, c_1 \cdot f, \mathfrak{M}, Infor)$ has been queried, then the algorithm aborts; otherwise the algorithm continues.
  e) The algorithm computes $x_2 = \frac{k}{c_2}$, $x_3 = g^{-k} \cdot (pk_l)^f$, $x_4 = (pk_l)^{\frac{f}{c_2}}$, and then outputs a group signature $\sigma = \{c_1'', c_2, x_2, x_3, x_4, y\}$ to the adversary $\mathcal{A}$, and saves the tuple (t, d, c_2, f, k) to S_List.

Guess: If the algorithm $\mathcal{B}$ does not abort as a consequence of one of the queries above, the adversary $\mathcal{A}$ will, with probability at least ε, output a bit $x' \in \{0, 1\}$, and succeed ($x' = x$) and return a valid forgery ($\mathfrak{M}^*$, σ^*, $Infor^*$, $RL_{pk^*}^t$) for the challengers u_0^* and u_1^*, where the identity $Infor^*$ and the revocation list $RL_{pk^*}^t$ are arbitrary forgeries generated by $\mathcal{A}$.

Then, if the adversary $\mathcal{A}$ did not query the oracle **H_1 Queries**, or U_List is empty or S_List is empty, then the algorithm $\mathcal{B}$ aborts.

Otherwise, the algorithm $\mathcal{B}$ can get $h_2 = H_1(*) = g^{b \cdot s}$. So, when the condition $DL_{h_2}((h_2)^{a_{x'} \cdot f \cdot c_2 - k}) = DL_g(g^{a_{x'} \cdot f \cdot c_2 - k}) = a_{x'} \cdot f \cdot c_2 - k$ holds, we can get the followings:
$$h_2^{x_2 \cdot c_2} = (h_2)^{(a_{x'} \cdot f - \frac{k}{c_2}) \cdot c_2} = (g^{b \cdot s})^{(a_{x'} \cdot f - \frac{k}{c_2}) \cdot c_2} = (g^{b \cdot s})^{(a_{x'} \cdot f \cdot c_2 - k)} = g^{a_{x'} \cdot b \cdot s \cdot f \cdot c_2 - b \cdot s \cdot k},$$
then $\mathcal{B}$ computes and outputs $(h_2^{x_2 \cdot c_2} \cdot g^{b \cdot s \cdot k})^{\frac{1}{c_2 \cdot s \cdot f}} = g^{a_{x'} \cdot b}$, which is the solution to the given CDH problem.

Now, we analyze the probability of the algorithm $\mathcal{B}$ not aborting. For the simulation to complete without aborting, we require that all ***Group-Setup*** queries, all ***Join-User*** queries and all ***Revoke-User*** queries are fresh, and all ***Sign*** queries do not abort in the Queries Phase 1 and 2. So, if the algorithm $\mathcal{B}$ does not abort, then the following conditions must hold:

(a) All ***Group-Setup*** queries are fresh in the Queries Phase 1 and 2, because $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$ is uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_0 is $\frac{1}{2^{n_q}}$, then the failure probability of the queries is at most $\frac{q_{g_1}+q_{g_2}}{2^{n_q}}$.

(b) All ***Join-User*** queries are fresh in the Queries Phase 1 and 2, the collision probability of H_0 is $\frac{1}{2^{n_q}}$, and because $t, d \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$ and the collision probability of H_2 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$, then the failure probability of the queries is at most $(q_{j_1} + q_{j_2}) \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}})$.

(c) All ***Revoke-User*** queries are fresh in the Queries Phase 1 and 2, similarly the collision probability of H_0 is $\frac{1}{2^{n_q}}$, and because $t, d \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$ and the collision probability of H_2 is $q_h \cdot \frac{1}{2^{n_q}} = \frac{q_h}{2^{n_q}}$, then the failure probability of the queries is at most $(q_{r_1}+q_{r_2}) \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}})$.

(d) All ***Sign*** queries do not abort in the Queries Phase 1 and 2, then we may get the followings:

- The algorithm may abort in the setup b), namely u_2 has been queried on the oracle **H_1 Queries**. So, as $t, d, c_2, y, f, k \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q^6$, the collision probability of H_1 is $q_h \cdot \frac{1}{2^{6 \cdot n_q}} = \frac{q_h}{2^{6 \cdot n_q}}$, then the failure probability of the queries is at most $\frac{(q_{s_1}+q_{s_2}) \cdot q_h}{2^{6 \cdot n_q}}$;
- The algorithm may abort in the setup d), namely the tuple $(u_2, v_2, g^d, c_1 \cdot f, \mathfrak{M}, Infor)$ has been queried on the oracle **H_3 Queries**. So, as $j \in \mathbb{Z}_q^*$ is uniformly distributed in $\mathbb{Z}_q$, the collision probability of H_3 is $(2 \cdot q_h + q_{s_1} + q_{s_2}) \cdot \frac{1}{2^{n_q}} = \frac{2 \cdot q_h + q_{s_1} + q_{s_2}}{2^{n_q}}$, then the failure probability of the queries is at most $\frac{(q_{s_1}+q_{s_2}) \cdot (2 \cdot q_h + q_{s_1} + q_{s_2})}{2^{n_q}}$.

Therefore, from the above analysis, we get that the algorithm $\mathcal{B}$ can compute $g^{a \cdot b}$ from the forgery as shown above, with probability at least $\varepsilon' = \varepsilon - \frac{q_{g_1}+q_{g_2}}{2^{n_q}} - (q_{j_1}+q_{j_2}) \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}}) - (q_{r_1} + q_{r_2}) \cdot (\frac{1}{2^{n_q}} + \frac{2 \cdot q_h}{2^{n_q}}) - \frac{(q_{s_1}+q_{s_2}) \cdot q_h}{2^{6 \cdot n_q}} - \frac{(q_{s_1}+q_{s_2}) \cdot (2 \cdot q_h + q_{s_1} + q_{s_2})}{2^{n_q}}$. The time complexity of the algorithm $\mathcal{B}$ is $\hbar' = \hbar + O((q_h + q_{g_1} + q_{g_2} + 4 \cdot (q_{j_1} + q_{j_2}) + 5 \cdot (q_{r_1} + q_{r_2}) + 12 \cdot (q_{s_1} + q_{s_2})) \cdot C_{exp} + 4 \cdot (q_{s_1} + q_{s_2}) \cdot C_{mul})$, where we assume that the time for integer addition, integer multiplication and hash computation can both be ignored.

Thus, Theorem 6.3 follows.

REFERENCES

[1] D. Chaum, E. van Heyst. Group Signatures. In Eurocrypt'91, LNCS 547, pp. 257-265, Springer, 1991.

[2] G. Ateniese, J. Camenisch, M. Joye, G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Crypto'00, LNCS 1880, pp. 255-270, 2000.

[3] M. Bellare, D. Micciancio, B. Warinschi. Foundations of group signatures: Formal definitions, simplified require- ments, and a construction based on general assumptions. In Eurocrypt'03, LNCS 2656, pp. 614-629, 2003.

[4] D. Boneh, X. Boyen, H. Shacham. Short Group Signatures. In Crypto'04, LNCS 3152, pp. 41-55. Springer, 2004.

[5] E. Bresson, J. Stern. Efficient Revocation in Group Signatures. In PKC'01, LNCS 1992, pp. 190-206, 2001.

[6] G. Ateniese, D. Song, G. Tsudik. Quasi-Efficient Revocation in Group Signatures. In Financial Cryptography'02, LNCS 2357, pp. 183-197, 2002.

[7] J. Camenisch, A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In Crypto'02, LNCS 2442, pp. 61-76, Springer, 2002.

[8] D. Boneh, H. Shacham. Group signatures with verifier-local revocation. In ACM-CCS'04, pp. 168-177, 2004.

[9] L. Nguyen. Accumulators from Bilinear Pairings and Applications. In CT-RSA'05, LNCS 3376, pp. 275-292, 2005.

[10] J. Camenisch, M. Kohlweiss, C. Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In PKC'09, LNCS 5443, pp. 481-500, 2009.

[11] E. Brickell. An efficient protocol for anonymously providing assurance of the container of the private key. Sub- mission to the Trusted Computing Group. April, 2003.

[12] T. Nakanishi, N. Funabiki. Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. In Asiacrypt'05, LNCS 5443, pp. 533-548, 2009.

[13] S. Zhou, D. Lin. Shorter Verifier-Local Revocation Group Signatures from Bilinear Maps. In CANS'06, LNCS 4301, pp. 126-143, Springer, 2006.

[14] B. Libert, D. Vergnaud. Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model. In CANS'09, LNCS 5888, pp. 498-517, 2009.

[15] E. Brickell, J. Camenisch, L. Chen. Direct Anonymous Attestation. In ACM-CCS'04, pp. 132-145, 2004.

[16] T. Nakanishi, H. Fujii, Y. Hira, N. Funabiki. Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In PKC'09, LNCS 5443, pp. 463-480, 2009.

[17] B. Libert, T. Peters, M. Yung. Scalable Group Signatures with Revocation. Advances in Cryptology-EUROCRYPT 2012, LNCS 7323, Springer-Verlag, 2012, pp.609-627.

[18] B. Libert, T. Peters, M. Yung. Scalable Group Signatures with Almost-for-Free Revocation. Advances in Cryptology-CRYPTO2012, LNCS 7417, Springer-Verlag, 2012, pp.571-589.

[19] L. Ibraimi, S. Nikova, P. Hartel, W. Jonker. An Identity-Based Group Signature with Membership Revocation in the Standard Model. Available at: http:/doc.utwente.nl/72270/1/Paper.pdf.

[20] K. Emura, A. Miyaji, K. Omote. An r-Hiding Revocable Group Signature Scheme: Group Signatures with the Property of Hiding the Number of Revoked Users. Journal of Applied Mathematics, Volume 2014, Article ID 983040, 14 pages.

[21] D. Boneh, M. Franklin. Identity-based encryption from the Weil pairing. In: J.Kilian, ed. Advances in Cryptology-CRYPTO 2001. LNCS 2139, Berlin:Springer-Verlag,2001. 213-229.

[22] D. Boneh, M. Hanburg. Generalized identity based and broadcast encryption schemes. In: J. Pieprzyk, ed. Advances in Cryptology-ASIACRYPT 2008. LNCS 5350, Berlin:Springer-Verlag, 2008. 455-470.

[23] K. G. Paterson, J. C. N. Schuldt. Efficient identity-based signatures secure in the standard model, ACISP2006, LNCS 4058, Springer-Verlag, 2006, pp.207-222.

[24] B. Waters, Efficient identity-based encryption without random oracles, Advances in Cryptology-EUROCRYPT 2005, LNCS 3494, Springer-Verlag, 2005, pp.114-127.

[25] P. S. L. M. Barreto, B. Libert, N. McCullagh, J. Quisquater. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In B. Roy, editor(s), Asiacrypt 2005, LNCS 3788, Berlin:Springer-Verlag,2005. 515-532.

[26] J.C. Cha, J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In Y. Desmedt, editor, Public Key Cryptography - PKC 2003, LNCS 2567, Berlin:Springer-Verlag,2002. 18-30.

[27] F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg, H. Heys, editors, Selected Areas in Cryptography 9th Annual International Workshop, SAC 2002, LNCS 2595, Berlin:Springer-Verlag,2003. 310-324.

[28] M. H. Au, J. K. Liu, T. H. Yuen, D. S. Wong. ID-based ring signature scheme secure in the standard mode, In Proceeding of IWSEC 2006, pp.1-16.

[29] F. Zhang, K. Kim, ID-based blind signature and ring signature from pairings. Asiacrypt2002, LNCS 2501, Berlin:Springer-Verlag, 2002. 533-547.

[30] Man Ho Au, Joseph K. Liu, Willy Susilo, Tsz Hon Yuen: Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction. Theoretical Computer Science. 469: 1-14 (2013)

[31] D. Chaum, T. P. Pedersen. Wallet databases with observers. In Ernest Brickell, editor, Proceedings of Crypto 92, volume 0740 of LNCS, pages:89-105.

[32] M. Jakobsson, C. Schnorr. Efficient oblivious proofs of correct exponentiation. In Bart Preneel, editor, Proceedings of the IFIP Conference on Communications and Multimedia Security 99, volume 152,pages:71-86. Kluwer.

[33] E. J. Goh, S. Jarecki. A signature scheme as secure as the Diffie-Hellman problem. Advances in Cryptology-EUROCRYPT2003.LNCS2656,Berlin:Springer,2003,pages:401-415.

[34] B. Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. Advances in Cryptology-CRYPTO 2005,vol. 3621 of Lecture Notes in Computer Science, Springer, pages: 511-526.

[35] A. Shamir, Y. Tauman. Improved online/offline signature scheme. In Joe Killian, editor, Proceedings of Crypto 01, volume 2139 of LNCS,pages:355-367.