Shorter Verifier-Local Revocation Group Signatures From Bilinear Maps [* ** *]

Sujing Zhou and Dongdai Lin

SKLOIS Lab, Institute of Software,
Chinese Academy of Sciences, Beijing, P. R. China
`zhousujing@is.iscas.ac.cn`, `ddlin@is.iscas.ac.cn`

Abstract. We propose a new computational complexity assumption from bilinear map, based on which we construct Verifier-Local Revocation group signatures with shorter lengths than previous ones.
Keywords: LRSW Assumption; Group Signature; Verifier-Local Revocation; Bilinear Map.

1 Introduction

Group signature [1] is motivated by enabling members of a group to sign on behalf of the group without leaking their own identities, and at the same time the signer's identity can be discovered by the group manager (GM) when a dispute occurs.

In brief, a group signature scheme is a signature scheme that has multiple secret keys corresponding to a single public key. A group signature should at least include the following five algorithms: Setup, Join, GSig, GVer and Open. Setup is executed by the group manager (GM); Join is an interactive protocol between a group member and GM or a separate issuing authority (IA); GSig is an algorithm run by any group member; any one can execute GVer to check the validity of a given group signature; Open is used by GM or a separate opening authority (OA) to find the identity of the signer given a group signature.

Various applications have been found for group signature schemes, such as anonymous authentication, internet voting and bidding. But wide implementation of group signatures in the real world has been prevented because of some factors, among which is efficient membership revocation as pointed out in [2].

Nontrivial resolutions to membership revocation have been proposed with regard to specific group signature schemes. The resolutions can be classified into two categories. One is based on *witness* [3–5], another is based on revocation list (RL) [6, 7]. Resolutions based on witness is advantageous over the latter in that growing revocation lists are not needed to maintain, but in some applications

[*] Supported by 973 Project of China (No.2004CB318004), 863 Project of China (No. 2003AA144030) and NSFC90204016.

[**] This is the full version of the paper in CANS 06. Additionally a new scheme 4, a group signature without random oracles, is added.

RL based revocations are more suitable because they admit shorter signature size [8].

RL Based Revocation. In this category, a natural resolution is to let GM issue a revocation list of identities (public membership keys) RL, any group member proves in a zero-knowledge way that his identity hidden in the group signature is not equal to any one in RL[6]. The drawback is that signature size is linearly dependent on the size of RL.

[7] improved the above approach resulting in a scheme that signature size and computation are constant while the complexity of GVer is linearly dependent on the size of RL. In this resolution, GM publishes a RL which includes $V_i = f(pcert_i)$, i.e., evaluations of one way function f on partial certificate information $pcert_i$ which is unique to each group member. In signing a message, member i includes a random R, and $T = f'(V_i, R)$ (f' is another one way function which may equal f) in the group signature. Verifiers check if $T = f'(V_i, R)$ by trying every V_i in the current RL.

The idea of [7] is followed by [8, 9] etc., and is named *verifier-local revocation* (VLR) and formalized in [8]. Nakanishi et. al. [9], however, pointed out previous VLR schemes have a drawback of backward linkability, and proposed another VLR scheme based on [8] with the feature of backward unlinkability (BU), i.e., group signatures generated by the same group member is unlinkable except himself and GM, even after this member has been revoked (his/her revocation token is published).

Contributions. We propose a new computational complexity assumption from bilinear map, and a new standard signature, two new verifier-local revocation group signature, one without backward unlinkability, another with backward unlinkability, based on our assumption. The proposed group signature schemes are more efficient both in signature length and signature generation/verification than previous ones.

Organization. Our new complexity assumption and the new standard signature are described in Section 3. The proposed new group signatures from bilinear map are presented in Section 5, with corresponding security proofs provided in Appendixes.

2 Preliminaries

Suppose that $G_1 = \langle g \rangle$, $G_2 = \langle \tilde{g} \rangle$ and G_3 are multiplicative cyclic groups of prime order p, there exists an efficient non-degenerate bilinear map $e : G_1 \times G_2 \rightarrow G_3$, i.e., $e(u^a, v^b) = e(u, v)^{ab}$ for any $u \in G_1$, $v \in G_2$, $a, b \in Z_p$, and $e(g, \tilde{g}) \neq 1$.

Definition 1 (LRSW Assumption [10]) *Suppose G_1, G_2, G_3 are defined as above and generated by a setup algorithm. Let $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$, $O_{x,y}(.)$ be an oracle that, on input a value $m \in Z_p^*$, outputs a triple (a, a^y, a^{x+mxy}) for a randomly chosen $a \in G_1$. Then for any probabilistic polynomial time (PPT) bounded adversary $\mathcal{A}$, the following probability is negligible:*

$\Pr\{(p, G_1, G_2, G_3, e) \leftarrow Setup(1^k); x \xleftarrow{R} Z_p^*; y \xleftarrow{R} Z_p^*; \widetilde{X} = \tilde{g}^x; \widetilde{Y} = \tilde{g}^y; (m, a, b, c) \leftarrow \mathcal{A}^{O_{x,y}}(g, \tilde{g}, e, \widetilde{X}, \widetilde{Y}) : m \in Z_p^* \setminus Q \wedge a \in G_1 \wedge b = a^y \wedge c = a^{x+mxy}\} < \epsilon$, where Q is the set of queries that $\mathcal{A}$ has made to $O_{x,y}(.)$.

Definition 2 (DDH) *In the bilinear groups G_1, G_2 defined above, for any PPT bounded probabilistic algorithm $\mathcal{A}$, the following probability is negligible:*
$\Pr\{\mathcal{A}(g^a, g^b, g^{ab}) = 1\} - \Pr\{\mathcal{A}(g^a, g^b, g^c) = 1\} < \epsilon$.
The probability is taken over the coin of $\mathcal{A}$ and random choice of $a, b, c \in Z_p^*$.

Notations. PK$\{(\alpha, \beta, ...) : R(\alpha, \beta, ...)\}$. denotes a proof of knowledge, in which a prover can show that he knows the values of $(\alpha, \beta, ...)$ satisfying the relation $R(\alpha, \beta, ...)$.

SK$\{(\alpha, \beta, ...) : R(\alpha, \beta, ...)\}\{m\}$. denotes a signature of knowledge [11], a non-interactive version of the above proof of knowledge transformed in Fiat-Shamir method [12].

Because the easiness of transformation between PK and SK, they might be mentioned interchangeably in the sequel. We let VSK denote the corresponding verification of SK.

$x \xleftarrow{R} S$ denotes x is chosen uniformly at random from the set S. $x \xleftarrow{\$} A(.,.,.)$ denotes x is generated from executing algorithm A where random variables are chosen uniformly at random. G^k, $(Z_p^*)^k$ denote a k tuple from G and Z_p^* respectively. $|M|$ denotes the binary length of string M, $|S|$ denotes the number of elements in the set S.

3 A New Complexity Assumption

The idea of Assumption 1 comes from an effort to reduce the items in LRSW Assumption from three to two, so that the signature size based on the assumption will be shortened. After an analysis of all possible $(g^r, g^{f(r,x,y,m)})$, where $f(.) = c_0rx + c_1ry + c_2xy$, $c_i \in \{0, 1, m\}$ for $i = 0, 1, 2$, we found it seems unforgeable when $(c_0, c_1, c_2) \in \{(1, m, 1), (m, 1, m)\}$, and actually they are interchangeable to each other.

Assumption 1 (Our New Assumption) *Suppose G_1, G_2, G_3 are defined as in Section 2 and generated by a setup algorithm. Let $X = g^x$, $Y = g^y$, $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$, $x \neq y$, $O_{x,y}(.)$ be an oracle that, on input a value $m \in Z_p^*$, outputs a pair $(g^r, g^{r(x+my)+xy})$ for a randomly chosen $r \in Z_p^* \setminus \{1\}$. Then for any PPT bounded adversary $\mathcal{A}$, the following probability is negligible:*
$\Pr[(p, G_1, G_2, G_3, e, g, \tilde{g}) \leftarrow Setup(1^k); x \xleftarrow{R} Z_p^*; y \xleftarrow{R} Z_p^*; X = g^x; Y = g^y; \widetilde{X} = \tilde{g}^x; \widetilde{Y} = \tilde{g}^y; (m, a, b) \leftarrow \mathcal{A}^{O_{x,y}}(p, g, \tilde{g}, e, X, Y, \widetilde{X}, \widetilde{Y}) : m \in Z_p^* \backslash Q \wedge a = g^r \wedge a \notin \{1_{G_1}, g\} \wedge b = g^{r(x+my)+xy}] < \epsilon$,
where Q is the set of queries that $\mathcal{A}$ has made to $O_{x,y}(.)$, 1_{G_1} is the unit element of G_1.

Assumption 1 is hard in generic groups, i.e.,

Theorem 1. *Let $x \in Z_p^*$, $y \in Z_p^*$ and maps ξ_1, ξ_2, ξ_3 are chosen at random. Let $\mathcal{A}$ be an algorithm that solves the assumption in the generic group model, making a total of polynomial number Q_G queries to the oracles computing the group action in G_1, G_2, G_3, and the oracle computing the bilinear pairing e, and the oracle $O_{x,y}(.)$ as described in the above definition. Then the probability ε that $\mathcal{A}^{O_{x,y}}(p, \xi_1(1), \xi_2(1), \xi_1(x), \xi_1(y)), \xi_2(x), \xi_2(y)$ outputs $(m, \xi_1(r), \xi_1(r(x+my)+xy))$ is bounded as follows:*

$$\varepsilon \leq O(Q_G^2/p).$$

The proof (see Appendix A) follows similar proofs in [13, 14]. The relationship among Assumption 1, LRSW, and Strong Diffie-Hellman assumption [14] are still not clear. A new standard signature scheme can be obtained based on this assumption.

Scheme 1 Let G_1, G_2, G_3 and bilinear map e be the same as described in Section 2 and Assumption 1.

– KeyGen. Select $(x, y) \xleftarrow{R} Z_p^* \times Z_p^*$, $x \neq y$, set $X = g^x$, $Y = g^y$, $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$. The secret key is (x, y), public key is $(X, Y, \widetilde{X}, \widetilde{Y}, g, \tilde{g}, e, p)$.

– Sign. Given a message $m \in Z_p^*$, its signature is (U, V), where $U = g^r$, $V = g^{r(x+my)+xy}$, $r \xleftarrow{R} Z_p^* \setminus \{1\}$.

– Verify. Given a signature (U, V) of m, check if $e(V, \tilde{g}) = e(U, \widetilde{X}\widetilde{Y}^m)e(X, \widetilde{Y})$. If the equation holds, then accept (U, V) as a valid signature of m, otherwise reject it as invalid.

Lemma 1. *Signature scheme 1 is existentially unforgeable under Assumption 1.*

For a message $m \in \{0,1\}^*$ other than Z_p^*, apply a hash function $H : \{0,1\}^* \to Z_p^*$ to m, then run Sign and Verify on $H(m)$. Note that in algorithm Sign, Assumption 1 and the proposed VLR group signatures in the sequel, it is required $r > 1$, and further more r should be large enough to foil naive attacks against Discrete Logarithm, e.g., repeatedly multiply g to match a given g^r.

4 Definition of Verifier-Local Revocation Group Signature

We provide a variant definition of VLR group signature from [8, 9] as follows.

Definition 3 (VLR Group Signature) A VLR group signature scheme GS is a digital signature scheme comprising the following algorithms:

- Setup: an algorithm to get group public key *gpk* and group secret key *gsk* = (*ik*), where *ik* is secret key of IA. Each user, for example *i*, to join in the group has its *user secret key* and *user public key* pair (sk_i, pk_i). A publishable revocation list *RL* is maintained and initialized empty; a registration table *Reg*, kept secret to IA and OA, is initialized empty. Let the current time period be j, initialized to 0.

- Join: a probabilistic interactive protocol between IA and a user, in the end, user i obtains its *group signing key* gsk_i. Generally $gsk_i = (msk_i, mpk_i)$, where *member secret key* msk_i is selected jointly by i and IA, and kept secret to i, *member public key* of i or *member certificate* mpk_i is generated by IA. If Join is successful mpk_i is added into *Reg*.
- Revoke: To revoke member i at time period j, IA generates revocation token $grt_{i,j}$, adds it to RL_j.
- GSig: a probabilistic algorithm on input (gsk_i, j, m), where gsk_i is the group signing key of a member in the group, returns σ as a group signature on m at time period j.
- GVer: a deterministic algorithm on input $(gpk, RL_j, j, m, \sigma)$, where σ is purported to be a group signature on m at time period j when the revocation list is RL_j, returns 1 to accept the group signature as valid or 0 to deny the group signature as invalid.
- Open: on input a message-signature pair (m, j, σ), *Reg*, returns (i, π) indicating i is the purported identity of the group member who signed the signature when $i > 0$, or none of the members has generated σ when $i = 0$, and π is a proof of this claim.
- Judge: on input of $(gpk, RL_j, j, m, \sigma, i, \pi, pk_i)$, return 1 to accept the claim of π, or 0 to deny the claim.
- If j is constant, GS is a VLR group signature w/o backward unlinkability, otherwise it is a VLR scheme with backward unlinkability.

In the following paragraphs, we investigate a formal adversary model of VLR group signature based on [15, 8, 9].

Firstly we define the oracles similar to [15]. It is assumed that several global variables are maintained by the oracles: HU, a set of honest users; CU, a set of corrupted users; $GSet$, a set of message signature pairs; and $Chlist$, a list of challenged message signature pairs. Note that not all the oracles will be available to adversaries in defining a certain security feature.

$AddU(i)$: If $i \in HU \cup CU$, the oracle returns $\perp$, else adds i to HU, executes algorithm Join.

$CrptU(i,\ pk)$: If $i \in HU \cup CU$, the oracle returns $\perp$, else sets $pk_i = pk$, $CU \leftarrow CU \cup \{i\}$, and awaits an oracle query to $SndToI$.

$SndToI(i, M_{in})$: If $i \notin CU$, the oracle returns $\perp$; else it plays the role of IA in algorithm Join replying to M_{in}, a string sent from user i.

$SndToU(i, M_{in})$: If $i \in HU \cup CU$, the oracle returns $\perp$, else it plays the role of user i in algorithm Join, $HU \leftarrow HU \cup \{i\}$.

$USK(i)$: If $i \in HU$, the oracle returns sk_i and gsk_i, $CU \leftarrow CU \cup \{i\}$, $HU \leftarrow HU \setminus \{i\}$; else returns $\perp$.

$RReg(i)$: The oracle returns reg_i, the record in the registration table *Reg* corresponding to user i.

$WReg(i,\ s)$: The oracle sets $reg_i = s$ if i has not been added in reg.

$Revoke(i, j)$: The oracle returns revocation token grt_{ij} of member i at time period j.

GSig(i,j,m): If $i \notin HU$, the oracle returns $\perp$, else returns a group signature σ on m by user i at time period j. $GSet \leftarrow GSet \cup \{(i, j, m, \sigma)\}$.

Ch(b, i_0, i_1, m, j): If $i_0 \notin HU \cup CU$ or $i_1 \notin HU \cup CU$, the oracle returns $\perp$, else generates a valid group signature σ with i_b being the signer at time period j. $Chlist \leftarrow Chlist \cup \{(m, j, \sigma)\}$.

Open(m, j, σ): If $(m, j, \sigma) \in Chlist$, the oracle returns $\perp$, else if (m, j, σ) is valid, the oracle returns output of $\mathrm{Open}(Reg, m, j, \sigma)$.

CrptIA: The oracle returns the secret key ik of IA.

CrptOA: The oracle returns the registration table *Reg*.

We say an oracle is over another oracle if availability of the oracle implies functions of another oracle. For example, *WReg* is over *RReg* since the adversary can try to remember everything it has written to *Reg*; *CrptIA* is over *CrptU*, *SndToI* since knowledge of ik enables the adversary to act as the two oracles itself; *CrptIA* is also over *CrptOA*; *CrptOA* is over *Open* and *RReg* since OA has access to *Reg*. Note that we do not let *CrptIA* over *WReg* so as to provide flexibility when accesses to the database *Reg* are granted by an independent DBA (database administrator).

Correctness. For any adversary that is not computationally restricted, a group signature generated by an honest group member is always valid; algorithm Open will always correctly identify the signer given the above group signature; the output of Open will always be accepted by algorithm Judge.

Selfless-anonymity. This concept is named in [8]. Imagine a PPT adversary $\mathcal{A}$, whose goal is to distinguish the signer of a group signature $\sigma \leftarrow Ch(b, i_0, i_1, m, J)$ at time period J between $i_0, i_1 \in HU$, where i_0, i_1, m, J are all chosen by $\mathcal{A}$ itself.

Naturally the adversary $\mathcal{A}$ might want to get the group signing keys of some other honest group members except i_0, i_1 (through oracle *USK*); it might want to obtain some group signatures signed by i_0, i_1 at the time period J(through oracle *GSig*); it might want to see some outputs of OA (through oracle *Open* except (J, m, σ)); it might also try to corrupt some group members by running Join with IA (through oracles *CrptU* and *SndToI*); it might observe the communication of some honest members joining in (through *SndToU* if IA is corrupted, not available otherwise); it might want to write to *Reg* (through oracles *WReg*); it might want to revoke some honest group members except i_0, i_1. Obviously $\mathcal{A}$ should not be allowed to corrupt OA and IA and request to *RReg*, and it is also forbidden from requesting revocation token of i_0, i_1 before the challenged time period J (including J).

A VLR group signature GS is *selfless-anonymous* if the probability for any PPT adversary to win is negligible, i.e., the value of $\mathrm{Adv}^{anon}_{GS,\mathcal{A}}$ defined below is negligible.

$$\mathrm{Adv}^{anon}_{GS,\mathcal{A}}(k) = \Pr[Exp^{anon-1}_{GS,\mathcal{A}}(k) = 1] - \Pr[Exp^{anon-0}_{GS,\mathcal{A}}(k) = 1],$$

where experiments $Exp^{anon-b}_{GS,\mathcal{A}}(k)$ are defined as in Table 1.

Traceability. Imagine a PPT adversary $\mathcal{A}$, whose goal is to produce a valid group signature (m, σ) at time period j and a corresponding revocation list

Experiment $Exp^{anon-b}_{GS,A}(k)$, $b \in \{0,1\}$
$(gpk, ik) \xleftarrow{\$} \text{Setup}(1^k)$; $CU \leftarrow \varnothing$, $HU \leftarrow \varnothing$, $Chlist \leftarrow \varnothing$;
$d \xleftarrow{\$} A(gpk : Open, CrptU, SndToI, USK, Ch(b,.,.,.), GSig, WReg, Revoke)$,
Return d.

Table 1. Selfless-anonymity.

RL_j, the output of Open points to a non-existent and unrevoked member or an existing corrupted member but can not pass Judge.

Naturally the adversary $\mathcal{A}$ might corrupt some group members by running Join with IA (through oracles *CrptU* and *SndToI*); it might want to see some outputs of OA (through oracle *Open*); it might want to read from (through oracles *RReg*); or $\mathcal{A}$ might corrupt OA directly (through oracle $CrptOA$). Obviously $\mathcal{A}$ should not be allowed to corrupt IA and query *WReg*. Note that $\mathcal{A}$ might not bother to query about honest group members for they are of little help for it.

A VLR group signature GS is *traceable* if the probability for any PPT adversary to win is negligible, i.e., the value of $\text{Adv}^{trace}_{GS,\mathcal{A}}$ defined below is negligible.

$$\text{Adv}^{trace}_{GS,\mathcal{A}}(k) = \Pr[Exp^{trace}_{GS,\mathcal{A}}(k) = 1],$$

where experiment $Exp^{trace}_{GS,\mathcal{A}}(k)$ is defined as in Table 2.

Experiment $Exp^{trace}_{GS,A}(k)$
$(gpk, ik) \xleftarrow{\$} \text{Setup}(1^k)$; $CU \leftarrow \varnothing$, $HU \leftarrow \varnothing$;
$(m, \sigma, j, RL_j) \xleftarrow{\$} A(gpk : CrptOA, CrptU, SndToI)$.
If $\text{GVer}(gpk, RL_j, j, m, \sigma) = 0$, return 0, else $(i, \pi) \leftarrow \text{Open}(Reg, j, m, \sigma)$.
If $i = 0$ or $(\text{Judge}(gpk, RL_j, j, m, \sigma, i, \pi, pk_i) = 0$ and $i \in CU)$ then return 1, else return 0.

Table 2. Traceability.

Non-frameability. Imagine a PPT adversary $\mathcal{A}$, whose goal is to produce a valid group signature (m, σ) at time period j and a corresponding revocation list RL_j, the output of Open points to an existing unrevoked honest member i_h and the result passes Judge.

Naturally the adversary $\mathcal{A}$ might want to get the group signing keys of some group members (through oracle USK); it might want to obtain some group signatures signed by some honest group members (through oracle *GSig*); it might want to see some outputs of OA (through oracle *Open*); it might also try to corrupt some group members by running Join with IA (through oracles *CrptU* and *SndToI*); it might observe the communication of some honest members joining in (through *SndToU* if *CrptIA* is queried, not available otherwise); it might wait until more group members has joined in (through *AddU*); it might want to write to or read from *Reg* (through oracles *WReg*, *RReg*); or $\mathcal{A}$ might corrupt OA or IA directly (through oracle *CrptOA* and *CrptIA*). Obviously $\mathcal{A}$ should not be allowed to query $CrptU(i_h)$, $SndToI(i_h,.)$, $USK(i_h)$.

A VLR group signature GS is *non-frameable* if the probability for any PPT adversary to win is negligible, i.e., the value of $\mathrm{Adv}^{nf}_{GS,\mathcal{A}}$ defined below is negligible.

$$\mathrm{Adv}^{nf}_{GS,\mathcal{A}}(k) = \Pr[Exp^{nf}_{GS,\mathcal{A}}(k) = 1],$$

where experiment $Exp^{nf}_{GS,\mathcal{A}}(k)$ is defined as in Table 3 after taking consideration of “over” relationship between oralces.

```
Experiment $Exp^{nf}_{GS,A}(k)$
$(gpk, ik) \xleftarrow{\$} \mathrm{Setup}(1^k)$; $CU \leftarrow \varnothing$, $HU \leftarrow \varnothing$, $GSet \leftarrow \varnothing$;
$(m, \sigma, j, RL_j, i, \pi) \xleftarrow{\$} A(gpk : CrptIA, CrptOA, SndToU, GSig, USK, WReg)$.
If $\mathrm{GVer}(gpk, RL_j, j, m, \sigma) = 0$, return 0.
Else if $i \in HU$ and Judge $(gpk, RL_j, j, m, \sigma, i, \pi, pk_i) = 1$
and $(i, j, m, .) \notin GSet$, return 1, else return 0.
```

Table 3. Non-frameability.

Definition 4 *A VLR group signature scheme is secure if it is selfless-anonymous, traceable and non-frameable.*

4.1 VLR Group Signature with Backward Unlinkability

The following model and definitions conform to [9].

Definition 5 (BU-VLR group signature) A BU-VLR group signature, i.e., a group signature scheme with verifier-local revocation and backward unlinkability simultaneously consists of the following algorithms. We suppose the maximum number of group members is n and the total time period is T.

- KGen(n, T): A probabilistic algorithm to generate group public key gpk, secret key gsk_i for each group member $i \in [1, n]$, and revocation tokens grt_{ij} for each member i at time period j.
- GSig(gpk, j, gsk_i, m): A probabilistic algorithm that produces a signature σ on message $m \in \{0,1\}^*$ at time period j by group member i who possesses the secret key gsk_i.
- Revoke(RL_j, grt_{ij}): If i is to be revoked for the time period j, the group manager adds grt_{ij} to the revocation list of time period j, i.e., $RL_j \leftarrow RL_j \cup \{grt_{ij}\}$.
- GVer$(gpk, j, RL_j, \sigma, m)$: A deterministic algorithm executable by anyone to generate a one bit b. If $b = 1$, it means σ is a valid group signature on m by some valid member (whose revocation token does not exist in RL_j); if $b = 0$, it means otherwise.

KGen corresponds to algorithms Setup and Join. Open is omitted since GM can run GVer against unpublished revocation tokens to find a group member match.

Definition 6 (Correctness) A BU-VLR group signature is correct if for all $(gpk, gsk, grt) \leftarrow \mathrm{KGen}(n, T)$, all $j \in [1, T]$, all $i \in [1, n]$, and all $m \in \{0, 1\}^*$, $\mathrm{GVer}(gpk, j, RL_j, \mathrm{GSig}(gpk, j, gsk_i, m), m) = 1 \leftrightarrow grt_{ij} \notin RL_j$.

Definition 7 (BU-Anonymity) A BU-VLR group signature has BU-anonymity if any PPT bounded probabilistic adversary $\mathcal{A}$ only has probability of $\frac{1}{2} + \epsilon$ (ϵ is negligible), i.e., with advantage of ϵ, to win in the following game.

- Setup: An instance of the BU-VLR group signature is established and gpk, gsk, grt are generated by a challenger, $\mathcal{A}$ is given only gpk.
- Queries:
  - Signing queries: $\mathcal{A}$ is allowed to request a signature on any message m for any group member i at time period j.
  - Corruption: $\mathcal{A}$ is allowed to request the secret key of any group member i, i.e., gsk_i.
  - Revocation: $\mathcal{A}$ is allowed to request the revocation token of any group member i at any time period j, i.e., grt_{ij}.
- Challenge: $\mathcal{A}$ outputs some (m, i_0,i_1, J) on the conditions that group members i_0 and i_1 have not been corrupted, and their revocation tokens have not been requested before time period J (including J). The challenger randomly selects $\phi \in \{0, 1\}$ and responds with a group signature on m by group member i_ϕ at time period J.
- Restricted queries: $\mathcal{A}$ is allowed to continue queries of Signing, Corruption and Revocation, except that i_0 and i_1 are forbidden in Corruption queries, and their Revocation queries are not allowed before time period J and current time period that $\mathcal{A}$ is to generate output (including J and current time)
- Output: $\mathcal{A}$ has to output a one bit value ϕ', and wins if $\phi' = \phi$.

Definition 8 (Traceability) A BU-VLR group signature has traceability if any PPT bounded probabilistic adversary $\mathcal{A}$ only has negligible probability ϵ to win in the following game.

- Setup: An instance of the BU-VLR group signature is established and gpk, gsk, grt are generated by a challenger, $\mathcal{A}$ is given gpk, grt. A set U is initialized empty.
- Queries:
  - Signing queries: $\mathcal{A}$ is allowed to request a signature on any message m for any group member i at time period j.
  - Corruption: $\mathcal{A}$ is allowed to request the secret key of any group member i, i.e., gsk_i, i is added into U.
- Output: $\mathcal{A}$ has to output (m^*, j^*, $RL^*_{j^*}$, σ^*), and it wins if (1) GVer(gpk, j^*, $RL^*_{j^*}$, σ^*, m^*)= 1, and (2) σ^* is traced to a group member outside of $U \setminus RL_{j^*}$ or failure, and (3) $\mathcal{A}$ has not obtained σ^* in signing queries on message m^* for this group member at time period j^*.

5 Proposed VLR Group Signature

Brief Idea. Actually a group signature can be viewed as a proof of knowledge of a standard signature signed by an authority ([11], [16], [17], [18], [19], [4], [10], [20]), so every standard signature can be employed to construct a group signature scheme [21], the point is how to obtain an efficient group signature scheme, and not every standard signature will result in efficient construction.

Scheme 1, however, has two features that make it a suitable candidate for group signature.

- For any signature (U,V) of m, any one can derive a new signature (U',V') of the same message: $U' = Ug^{r'}$, $V' = VX^{r'}Y^{mr'}$, where $r' \overset{R}{\longleftarrow} Z_p^*$. Every random derivation is independent from each other.
- The generation of (U,V) can be done even when m is not revealed: $U = g^r$, $V = C^r g^{rx+xy}$, where $C = Y^m$.

The brief idea of Scheme 2 is: let group member i choose his secret key s_i, commit it (without information theoretic hiding) to $C_i = Y^{s_i}$. IA, as the signer of Scheme 1, signs blindly on s_i, i.e., outputs (U_i, V_i) as a member certificate of i. Group member i firstly generates a proof of knowledge of (s_i, U_i, V_i), when he is asked to produce a group signature of a message, then randomize his member certificate according to the first feature, now what left is to prove his knowledge of s_i, which has standard and efficient resolution already [22].

The brief idea of Scheme 3 follows the above idea. Additionally a revocation tag $e(g^{s_i}, h_j)^\delta$, g^δ ($\delta \overset{R}{\longleftarrow} Z_p^*$) as well as a proof of knowledge of (s_i, δ) is appended to the end of a group signature, where h_j is chosen at the beginning of time period j by the revocation authority (IA or OA), and published along with the revocation list at that time RL_j. The method is the same of [9], but our resulted scheme is about 23% shorter in signature length.

5.1 The Scheme without Backward Unlinkability

Scheme 2 utilizes a trusted third party TP in case OA might be corrupted. If OA is fully trusted, then the scheme can be simplified by eliminating TP and signature of i on A_i, without invalidating corresponding proofs. Note that we omit the index of time period in the following description because it is a VLR scheme without backward unlinkability.

Scheme 2 (Our Proposal w/o BU) Let G_1, G_2, G_3 and bilinear map $e : G_1 \times G_2 \rightarrow G_3$ be defined as in Section 2.

- Setup: Group secret key is $(x, y) \overset{R}{\longleftarrow} Z_p^* \times Z_p^*$, $x \neq y$, public key is (g, $\tilde{g}$, e, $X = g^x$, $Y = g^y$, $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$, p, $h \overset{R}{\longleftarrow} G_2$). A hash function $H : \{0,1\}^* \rightarrow Z_p^*$.
  A registration table *Reg* is maintained and initialized empty.
  A signature scheme $\mathcal{S}$ is selected, which is similar to the scheme in [23]:

  $\mathcal{S}$.KeyGen: Select $z \overset{R}{\longleftarrow} Z_p^*$ as a secret key, set $pk = g^z \in G_1$ as a public key.
  $\mathcal{S}$.Sign: To sign a message $m \in G_2$, calculate $\sigma = m^z$ as the signature.
  $\mathcal{S}$.Verify: To verify a given message-signature pair (m, σ), check if $e(g, \sigma) = e(pk, m)$.

A third trusted party TP is also selected. Each user i has to generate his public key pk_i and secret key $sk_i = z_i$ of $\mathcal{S}$, and register them to TP before joining in the group. TP will publish the users' public key and corresponding identity i.

– Join: A user i interacts with IA to obtain his certificate in a private channel as follows:

User $\rightarrow$ IA:	User i selects $\tilde{s}_i \overset{R}{\longleftarrow} Z_p^*$, sends $\widetilde{C}_i = Y^{\tilde{s}_i}$ along with a proof of knowledge of $\tilde{s}_i$ to IA.
User $\leftarrow$ IA:	IA verifies that the proof of knowledge is correct, then selects $r_1 \overset{R}{\longleftarrow} Z_p^*$ so that $C_i = \widetilde{C}_i Y^{r_1}$ is different from all values already stored in *Reg*. It also selects $\alpha_i \overset{R}{\longleftarrow} Z_p^*$, computes $U_i = g^{\alpha_i}$, $V_i = (XC_i)^{\alpha_i} g^{xy}$ and returns (r_1, U_i, V_i) to i.
User $\rightarrow$ IA:	User i computes $s_i = \tilde{s}_i + r_1$, checks if $e(V_i, \tilde{g}) = e(U_i, \widetilde{X}\widetilde{Y}^{s_i})e(X, \widetilde{Y})$, accepts (U_i, V_i) as his member certificate if the above equation holds. User i also generates a $\mathcal{S}$ signature on $A_i = \widetilde{Y}^{s_i}$, i.e., $\sigma_i = A_i{}^{z_i}$, and a proof of equality [24, 25] of $\log_{\widetilde{Y}} A_i$ and $\log_Y C_i$, sends (A_i, σ_i) to IA.
IA:	IA checks if (A_i, σ_i) is valid under pk_i, stores it in *Reg* if that is the case.

– GSig: Member i (in possession of member certificate (U_i, V_i) and secret key s_i) generates a group signature σ on message m as follows.

Firstly, calculate $U' = U_i g^{r'}$, $V' = V_i X^{r'} Y^{s_i r'}$, where $r' \overset{R}{\longleftarrow} Z_p^*$.
Secondly, generate a signature of knowledge of s_i, i.e.,
$\tau = \mathrm{SK}_1\{s_i\colon\ e(U', \widetilde{Y})^{s_i} = e(V', \tilde{g})e(U'Y, \widetilde{X})^{-1}\}\{m\}$, which is standard, i.e., $\tau = (s, c)$, where $c = H(e(U', \widetilde{Y})^{k_1}, V', X, Y, \widetilde{X}, \widetilde{Y}, m)$, $s = k_1 + cs_i$, $k_1 \overset{R}{\longleftarrow} Z_p^*$. It can be proved sound and of honest verifier zero-knowledge exactly as in [22, 20, 26] etc.

The group signature of m signed by i is $\sigma = (U', V', \tau)$.

– GVer: A verifier does the following checks, given a group signature $\sigma = (U', V', \tau)$ of m:

Firstly, check the validity of $\tau = (s, c)$ by running $\mathrm{VSK}_1(\tau)$, which is also standard, i.e., verify if $c = H(e(U', \widetilde{Y})^s [e(U'Y, \widetilde{X})^{-1} e(V', \tilde{g})]^{-c}, V', X, Y, \widetilde{X}, \widetilde{Y}, m)$, return 1 if that is the case, 0 otherwise.
Secondly, check if there is a $A \in RL$ that $e(V', \tilde{g}) = e(U', \widetilde{X}A)e(X, \widetilde{Y})$, return 0 if that is the case, 1 otherwise.
σ is valid if both checks return 1.

– Open: The identity of the signer of a given group signature (m, U', V', τ) can be opened as follows.

Check if $e(V', \tilde{g}) = e(U', \widetilde{X}A_i)e(X, \widetilde{Y})$ for some A_i stored in *Reg*.
If A_i satisfies the above equation, generates π, a proof of knowledge of (A_i, σ_i), i.e., let $W_\alpha = \sigma_i h^\alpha$, $W_\beta = A_i^\beta$, $(\alpha, \beta) \overset{R}{\longleftarrow} Z_p^* \times Z_p^*$, $\pi = \mathrm{PK}_2\{(\alpha, \beta) : [e(V', \tilde{g})e(U'Y, \widetilde{X})^{-1}]^\beta = e(U', W_\beta) \wedge e(g, W_\alpha)^\beta e(g, h)^{-\alpha\beta} = e(pk_i, W_\beta)\}$.
If no record in *Reg* satisfy the above equation, set $i = 0$, $\pi = NULL$.

Here pk_i is the $\mathcal{S}$ public key of the revealed member i, which are bound together with i by TP. The detail of PK_2 is provided in Appendix B. The output of Open is (i, π).

Note that IA can also open a group signature. IA checks if there exists a A_i stored in *Reg* that $e(U', \widetilde{X}A_i) = e(V'/g^{xy}, \tilde{g})$, it retrieves the matching A_i and generates π, a proof of knowledge of (xy, A_i, σ_i) if that is the case. This method can only be executed by IA knowing x, y, the former method can be done by any opening authority assigned by IA, if only the OA has access to *Reg*. Thus our scheme has a kind of flexibility.

– Judge: This algorithm judges the correctness of output of Open by checking π.
– Revoke: To revoke a group member i, IA or OA just publishes the corresponding A_i in *RL*.

The security results of Scheme 2 are as follows.

Theorem 2 (Traceability). *Scheme 2 is traceable in random oracle model under Assumption 1.*

The proof is in Appendix C.

Theorem 3 (Non-frameability). *Scheme 2 is non-frameable in random oracle model under Discrete Logarithm assumption in group G_3 which implies Discrete Logarithm is hard in G_1, G_2 too.*

The proof is standard and similar to those of [26] etc., because a valid group signature of Scheme 2 is in fact a zero-knowledge proof of knowledge of s_i that $e(V', \tilde{g}) = e(U', \widetilde{X}\widetilde{Y}^{s_i})e(X, \widetilde{Y})$, and s_i is never exposed to others including IA and OA.

Theorem 4 (Selfless-anonymity). *Scheme 2 is selfless-anonymous in random oracle model under DDH assumption in G_1.*

The theorem is implied by by the following lemma with proof in Appendix D.

Lemma 2. *Suppose an adversary $\mathcal{A}$ breaks the selfless-anonymity of Scheme 2 with advantage ϵ after q_H hash queries, q_{sig} signature queries, then there exists an algorithm $\mathcal{B}$ breaking DDH assumption with advantage $\frac{\epsilon}{2}(\frac{1}{n} - \frac{q_H q_{sig}}{p})$, where n is the total number of group members.*

Open with Complexity $O(1)$. An alternative construction of obtaining Open with complexity independent with size of revocation list is to encrypt Y^{s_i} using the linear encryption scheme based on Linear Diffie-Hellman Assumption [4], i.e., select $(\alpha, \beta) \xleftarrow{R} Z_p^* \times Z_p^*$ and compute $T_1 = u^{\alpha}$, $T_2 = v^{\beta}$, $T_3 = Y^{s_i}w^{\alpha+\beta}$, where $u, v, w \in G_1$ are among the group public keys, and OA owns x_1, x_2 that $w = u^{x_1} = v^{x_2}$ as secret keys.

The group signature by member i is $(U_i', {V_i'}^r, T_1, T_2, T_3)$ plus a proof of knowledge of (α, β, s_i, r) accordingly. The resulted signature length is 1704 bits, equal to [4].

5.2 The Scheme with Backward Unlinkability

Our proposal Scheme 2 can be extended to include backward unlinkability in the same method as in [9], see the following description.

Scheme 3 (Our Proposal w/ BU) Let G_1, G_2, G_3 and bilinear map $e : G_1 \times G_2 \rightarrow G_3$ be defined as in Section 2.

- Setup: Same as Scheme 2, except that h is missing from the group public key.
- Join: Same as Scheme 2.
- GSig: Member i (in possession of certificate (U_i, V_i) and secret key s_i) generates a group signature σ of message m at time period j as follows.

  Calculate $U' = U_i g^{r'}$, $V' = V_i X^{r'} Y^{s_i r'}$, where $r' \overset{R}{\longleftarrow} Z_p^*$.
  Select $\delta \overset{R}{\longleftarrow} Z_p^*$, calculate $S = e(g^{s_i}, h_j)^{\delta}$, $T = g^{\delta}$.
  Generate a signature of knowledge of $(s_i, \delta, s_i\delta)$, i.e.,
  $\tau = \mathrm{SK}_2\{\alpha, \beta, \gamma : e(U', \widetilde{Y})^{\alpha} = e(V', \tilde{g})e(U'Y, \widetilde{X})^{-1} \wedge T = g^{\beta} \wedge S = e(g, h_j)^{\gamma} \wedge T^{\alpha} = g^{\gamma}\}\{m\}$,
  which is standard, i.e., $\tau = (s_\alpha, s_\beta, s_\gamma, c)$, where
  $c = H(e(U', \widetilde{Y})^{k_\alpha}, g^{k_\beta}, e(g, h_j)^{k_\gamma}, T^{k_\alpha}/g^{k_\gamma}, U', V', S, T, X, Y, \widetilde{X}, \widetilde{Y}, m)$, $(k_\alpha, k_\beta, k_\gamma) \overset{R}{\longleftarrow} Z_p^{*3}$,
  and $s_\alpha = k_\alpha + cs_i$, $s_\beta = k_\beta + c\delta$, $s_\gamma = k_\gamma + cs_i\delta$.

  The group signature of m signed by i at time period j is $\sigma = (U', V', S, T, \tau)$.
- GVer: A verifier does the following checks, given a group signature $\sigma = (U', V', S, T, \tau)$ on m at time period j:

  Firstly, check the validity of $\tau{=}(s_\alpha, s_\beta, s_\gamma, c)$ by running $\mathrm{VSK}_2(\tau)$, which is also standard, i.e., verify that if $c{=}H(e(U', \widetilde{Y})^{s_\alpha}[e(U'Y, \widetilde{X})e(V', \tilde{g})^{-1}]^c, g/T^c, e(g, h_j)^{s_\gamma}/S^c, T^{s_\alpha}/g^{s_\gamma}, U', V', S, T, X, Y, \widetilde{X}, \widetilde{Y}, m)$, return 1 if that is the case, 0 otherwise.
  Secondly, check if there is a $B \in RL$ that $S = e(T, B)$, return 0 if that is the case, 1 otherwise.
  σ is valid if both checks return 1.
- Revoke: To revoke a group member i at time period $j \in [1, t]$, IA or OA selects and publishes a unique $h_j = \widetilde{Y}^{r_j}$ ($r_j \overset{R}{\longleftarrow} Z_p^*$) for each time period j, and publishes the corresponding $B_{ij} = A_i^{r_j} \triangleq h_j^{s_i}$ in RL, where A_i is obtained from member i during algorithm Join, and stored in *Reg*.
- Open and Judge: Same as Scheme 2.

The correctness of Scheme 3 is easy to verify. The traceability and non-frameability follow from that of Scheme 2. What remains to analyze is selfless-anonymity in the case of backward unlinkability, i.e., BU-anonymity [9].

Theorem 5. *Scheme 3 is selfless-anonymous in random oracle model under DDH assumption in G_1.*

The theorem is implied by the following lemma (proved in Appendix E).

Lemma 3. *Suppose an adversary $\mathcal{A}$ breaks the selfless-anonymity of Scheme 3 with advantage ϵ, after q_H hash queries, q_S signature queries, then there exists an algorithm $\mathcal{B}$ breaking DDH assumption in G_1 with advantage $\frac{\epsilon}{2}(\frac{1}{n} - \frac{q_H q_S}{p})$.*

Our scheme has the feature of being BU-enabled and non-frameable at the same time. [9] can also be extended to satisfy the two requirements simultaneously just as how the basic scheme is enhanced with strong exculpability in [4], at the cost of longer signature length because knowledge of an extra exponent has to be proved.

5.3 The Scheme without Random Oracles

Our new assumption 1 can be used to construct an efficient group signature without Random Oracles (RO) following [13].

Scheme 4 (Our Proposal w/o RO) Let G_1, G_2, G_3 and bilinear map $e : G_1 \times G_2 \to G_3$ be defined as in Section 2.

– Setup:

GroupSetup. Group secret key is $(x, y) \xleftarrow{R} Z_p^* \times Z_p^*$, $x \neq y$, public key is $(g, \tilde{g}, e, X = g^x, Y = g^y, \widetilde{X} = \tilde{g}^x, \widetilde{Y} = \tilde{g}^y, p)$. The group manager or IA also maintain a database *Reg*.

UserKeyGen. Each user selects random $sk \in Z_p^*$ and random $h \in G_1$, and outputs its public key $pk = (h, e(h, \tilde{g})^{sk})$.

– Join: User i holding its secret key sk_i and public key $pk_i = (h_i, e(h_i, \tilde{g})^{sk_i})$ run interactively with IA.

User $\to$ IA:	User i submits its public key pk_i, the user provides a zero-knowledge proof of knowledge of the corresponding sk_i using any extractable proof technique (see [13] for a discussion of such proof techniques).
User $\to$ IA:	User i submits its tracing information $Q_i = \tilde{g}^{sk_i}$ to IA. Let $pk_i = (p_1, p_2)$, if $e(p_1, Q_i) = p_2$, and Q_i is new in *Reg*, IA stores Q_i in the database *Reg*; otherwise IA aborts.
User $\to$ IA:	User i sends $A = g^{sk_i}$ to IA.
User $\leftarrow$ IA:	IA computes $f_1 = g^r$, $f_2 = g^{r(x+my)+xy}$ and sends them to the user. User accepts them if $e(f_2, \tilde{g}) = e(f_1, \widetilde{X}\widetilde{Y}^{sk_i})e(X, \widetilde{Y})$. At the end of the protocol, the user obtains the following member certificate (f_1, f_2).

– GSig: Member i (in possession of certificate (f_1, f_2) and secret key sk_i) generates a group signature σ of message m as follows.

The user re-random its certificate by computing $a_1 = f_1 g^r$, $a_2 = f_2 X^r Y^{sk_i r}$, where $r \xleftarrow{R} Z_p^*$.
Next, the user chooses a random $v \in Z_p^*$ and sets $a_3 = a_1^{sk_i}$, $a_4 = a_1^v$.
The user generates a BB signature [14] on v using its secret key sk_i, i.e., $a_5 = \tilde{g}^{\frac{1}{sk_i+v}}$.
The user treats the value v as its one-time signing key and computes a BB signature on m using its secret key v, i.e., $a_6 = \tilde{g}^{\frac{1}{v+m}}$.

The group signature of m signed by i is $\sigma = (a_1, a_2, a_3, a_4, a_5, a_6)$.

– GVer: A verifier does the following checks, given a group signature $\sigma = (a_1, a_2, a_3, a_4, a_5, a_6)$ on m:

Firstly, check that (a_1, a_2) is a valid signature on $\log_{a_1} a_3$ under Scheme 1, i.e., if $e(a_2, \tilde{g}) = e(a_3, \tilde{X}\tilde{Y})e(X, \tilde{Y})$ holds.
Secondly, check if a_5 is a valid BB signature on $\log_{a_1} a_4$ for public key $(a_1, \tilde{g}, a_3)$, i.e., if $e(a_3a_4, a_5) = e(a_1, \tilde{g})$ holds.
In the end, check if a_6 is a valid BB signature on m for public key $(a_1, \tilde{g}, a_4)$, i.e., if $e(a_4a_1^m, a_6) = e(a_1, \tilde{g})$ holds.

– Open: IA checks whether $e(a_3, \tilde{g}) = e(a_1, Q_i)$ for each Q_i in *Reg*.
– VerifyOpen: Firstly, IA checks whether δ is a valid group signature by running *GVer*. Next, IA checks if there exists a $Q_i \in Reg$ that $e(p_1, Q_i) = p_2$. If both conditions are satisfied, then IA proceeds to convince a verifier that user i, whose public key is (p_1, p_2), is really the signer by either one of the following ways.
  - Total Anonymity Revocation: IA simply publish Q_i, so that verification of $e(p_1, Q_i) = p_2$ and $e(a_3, \tilde{g}) = e(a_1, Q_i)$ are available to anyone, with user i losing anonymity on any group signature it could generate thereafter.
  - Partial Anonymity Revocation: IA engage a zero-knowledge proof of knowledge of $Q \in G_2$ that $e(p_1, Q) = p_2$ and $e(a_3, \tilde{g}) = e(a_1, Q)$.

5.4 Efficiency Comparison

To implement our schemes, a group where DDH is hard and an efficient bilinear map is defined is required. A natural selection is non-supersingular elliptic curves defined on finite field, with MOV degree, i.e., embedding degree, larger than one, because distortion map which is the only tool solving DDH on an elliptic curve nowadays does not exist in these curves according to [27], and MNT curves happen to satisfy the requirements and can be constructed systematically [28]. So our schemes are realizable on MNT curves. Scheme of [13] has the same requirement as ours, while schemes of [9, 8] are also realizable on supersingular elliptic curves besides MNT curves.

The following table is a performance comparison of known VLR schemes in signature size, i.e., length of σ in bits, and computations required in algorithms GSig and GVer, i.e., multi-exponentiations (denoted as ME) number in G_1 and bilinear map (denoted as BM) number. Note that computations that permit preprocessing are not counted.

Note that the computation estimations are made according to [8], i.e., p is about 170 bits, elements of G_1 are 171 bits, and elements of G_3 are 1020 bits, achieving a security level similar to 1024 bits RSA. In case [13] and Scheme 4, elements of G_2 are chosen about 3 times that of G_1.

Acknowledgments

The authors are grateful to Jing Xu, Zhenfeng Zhang for the helpful discussions, and to Lingbo Wei for the final proofreading.

	$\vert\sigma\vert$ (bits)	GSig Comp.	GVer Comp.	Back.-Unlink.	Non-Frame.
[9]	2893	10 ME+1 BM	6 ME+$(2 + \vert RL\vert)$ BM	**Yes**	No
[13]	2052	8 ME	1 ME+$(9 + 2\vert RL\vert)$ BM	No	**Yes**
[8]	1192	8 ME+2 BM	6 ME+$(3 + \vert RL\vert)$ BM	No	No
[29]	2044	10 ME+ 1 BM	6 ME+$(2 + \vert RL\vert)$ BM	**Yes**	No
Scheme 2	682	3 ME	2 ME+$(3 + \vert RL\vert)$ BM	No	**Yes**
Scheme 3	2213	8 ME	5 ME+$(2 + \vert RL\vert)$ BM	**Yes**	**Yes**
Scheme 4	1710	6 ME	1 ME+$(6 + 2\vert RL\vert)$ BM	No	**Yes**

Table 4. A Comparison of Some VLR Group Signature Schemes.

References

1. Chaum and E. van Heyst, "Group signatures," in *EUROCRYPT'91*, LNCS 547, pp. 257–265, Springer-Verlag, 1991.
2. G. Ateniese and G. Tsudik, "Some open issues and new directions in group signature schemes," in *Financial Cryptography'99*, LNCS 1648, Springer-Verlag, 1999.
3. J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *CRYPTO'02*, LNCS 2442, pp. 61–76, Springer-Verlag, 2002.
4. D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *CRYPTO'04*, LNCS 3152, pp. 45–55, Springer-Verlag, 2004.
5. L. Nguyen, "Accumulators from bilinear pairings and applications," in *CT-RSA'05*, LNCS 3376, pp. 275–292, Springer-Verlag, 2005. A modified version is available at Cryptology ePrint Archive: Report 2005/123.
6. E. Bresson and J. Stern, "Efficient revocation in group signatures," in *PKC'01*, LNCS 1992, Springer-Verlag, 2001.
7. G. Ateniese, D. Song, and G. Tsudik, "Quasi-efficient revocation in group signatures," in *Financial Cryptography'02*, LNCS 2357, Springer-Verlag, 2002.
8. D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *CCS'04*, LNCS 3108, pp. 168–177, ACM Press, 2004.
9. T. Nakanishi and N. Funabiki, "Verifer-local revocation group signature schemes with backward unlinkability from bilinear maps," in *ASIACRYPT'05*, LNCS 3788, pp. 533–548, Springer-Verlag GmbH, 2005.
10. J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *CRYPTO'04*, LNCS 3152, pp. 56–72, Springer-Verlag, 2004.
11. J. Camenisch and M. Stadler, "Efficient group signatures schemes for large groups," in *CRYPTO'97*, LNCS 1296, pp. 410–424, Springer-Verlag, 1997.
12. A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in *CRYPTO'86*, LNCS 263, pp. 186–194, Springer, 1987.
13. G. Ateniese, J. Camenisch, B. de Medeiros, and S. Hohenberger, "Practical group signatures without random oracles," in *EUROCRYPT'06*.
14. D. Boneh and X. Boyen, "Short signatures without random oracles," in *EUROCRYPT'04*, LNCS 3027, pp. 56–73, Springer-Verlag, 2004.
15. M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *CT-RSA'05*, LNCS 3376, pp. 136–153, Springer-Verlag, 2005. Full Paper at http://www-cse.ucsd.edu/ mihir/papers/dgs.html.

16. J. Camenish and M. Michels, "A group signature scheme with improved efficiency," in *ASIACRYPT'98*, LNCS 1514, pp. 160–174, Springer, 1998.
17. J. Camenisch and M. Michels, "A group signature scheme based on an RSA-variant," in *Technical Report RS-98-27. BRICS, University of Aarhus*, Primary version of this paper appeared at ASIACRYPT'98, Springer-Verlag, November 1998.
18. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *CRYPTO'00*, LNCS 1880, pp. 255–270, Springer-Verlag, 2000.
19. G. Ateniese and B. de Medeiros, "Efficient group signatures without trapdoors," in *ASIACRYPT'03*, LNCS 2894, pp. 246–268, Springer-Verlag, 2003.
20. L. Nguyen and R. Safavi-Naini, "Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings," in *ASIACRYPT'04*, LNCS 3329, pp. 372–386, Springer-Verlag, 2004.
21. H. Peterson, "How to convert any digital signature scheme into a group signature scheme," in *Proc. Security Protocols Workshop, Paris*, LNCS 1361, pp. 177–190, Springer-Verlag, 1997.
22. C. P. Schnorr, "Effcient signature generation by smart cards," in *Journal of Cryptology*, vol. 4, pp. 161–174, 1991.
23. D. Boneh, B. Lynn, and H. Shacha, "Short signatures from the weil pairing," in *ASIACRYPT'01*, LNCS 2248, pp. 514–532, 2001.
24. D. Chaum and T. P. Pedersen, "Wallet database with observers," in *CRYPTO'92*, LNCS 740, pp. 89–105, Springer-Verlag Berlin Heidelberg, 1993.
25. J. Camenisch and M. Michels, "Separability and efficiency for generic group signature schemes," in *CRYPTO'99*, LNCS 1666, pp. 413–430, Springer-Verlag, 1999.
26. A. Kiayias, Y. Tsiounis, and M. Yung, "Traceable signatures," in *EUROCRYPT'04*, LNCS 3027, pp. 571–589, Springer, 2004.
27. E. R. Verheul, "Evidence that xtr is more secure than supersingular elliptic curve cryptosystems," *Journal of Cryptology*, vol. 17, pp. 277–296, 2004.
28. A. Miyaji, M. Nakabayashi, and S. Takano, "New explicit conditions of elliptic curve traces for fr-reduction," *IEICE Trans. Fundamentals*, vol. E84 A, no. 5, 2001.
29. S. Zhou and D. Lin, "A shorter group signature with verifier-location revocation and backward unlinkability." Cryptology ePrint Archive, Report 2006/100, 2006.
30. A. Kiayias and M. Yung, "Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders," in *Cryptology ePrint Archive, Report 2004/076*, 2004.

A Proof of Theorem 1

Proof. Consider an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ in the following game. $\mathcal{B}$ maintains two lists of pairs $L_1 = \{(F_{1,i}, \xi_{1,i}) : i = 0, ..., \tau_1 - 1\}$, $L_2 = \{(F_{2,i}, \xi_{2,i}) : i = 0, ..., \tau_2 - 1\}$, $L_3 = \{(F_{3,i}, \xi_{3,i}) : i = 0, ..., \tau_3 - 1\}$, such that at step τ in the game, we have $\tau_1 + \tau_2 + \tau_3 = \tau + 6$. The $F_{1,i}$, $F_{2,i}$, $F_{3,i}$ are polynomials in $Z_p[x]$, the $\xi_{1,i}$, $\xi_{2,i}$, $\xi_{3,i}$ are set to unique random strings in $\{0,1\}^*$.

We start the game at step $\tau = 0$ with $\tau_1 = 3$, $\tau_2 = 3$, $\tau_3 = 0$, they corresponds to $F_{1,0} = 1$, $F_{1,1} = x$, $F_{1,2} = y$, $F_{2,0} = 1$, $F_{2,1} = x$, $F_{2,2} = y$, and the random strings $\xi_{1,0}, \xi_{1,1}, \xi_{1,2}$, $\xi_{2,0}, \xi_{2,1}, \xi_{2,2}$.

$\mathcal{B}$ simulates the following oracles that $\mathcal{A}$ may query. Let τ_v denote the number of queries to oracle $O_{x,y}$ by $\mathcal{A}$, and initialize $\tau_v = 1$.

Group action: $\mathcal{A}$ inputs two group elements $\xi_{1,i}$, $\xi_{1,j}$ where $0 \le i, j \le \tau_1$, and a request to multiply/divide. $\mathcal{B}$ sets $F_{1,\tau_1} \leftarrow F_{1,i} \pm F_{1,j}$ accordingly. If $F_{1,\tau_1} = F_{1,u}$ for some $u \in \{0, ..., \tau_1 - 1\}$, then $\mathcal{B}$ sets $\xi_{1,\tau_1} = \xi_{1,u}$; otherwise it sets ξ_{1,τ_1} to a random string in $\{0,1\}^* \setminus \{\xi_{1,0}, ..., \xi_{1,\tau_1-1}\}$. Finally $\mathcal{B}$ returns ξ_{1,τ_1} to $\mathcal{A}$, adds $(F_{1,\tau_1}, \xi_{1,\tau_1})$to L_1 and increments τ_1. Group actions for G_2, G_3 is handled similarly.

Pairing: $\mathcal{A}$ inputs two group elements $\xi_{1,i}$ and $\xi_{2,j}$, where $0 \le i \le \tau_1$, $0 \le j \le \tau_2$. $\mathcal{B}$ sets $F_{3,\tau_3} \leftarrow F_{1,i} \cdot F_{2,j}$. If $F_{3,\tau_3} = F_{3,u}$ for some $u \in \{0, ..., \tau_3 - 1\}$, then $\mathcal{B}$ sets $\xi_{3,\tau_3} = \xi_{3,u}$; otherwise it sets ξ_{3,τ_3} to a random string in $\{0,1\}^* \setminus \{\xi_{3,0}, ..., \xi_{3,\tau_3-1}\}$. Finally $\mathcal{B}$ returns ξ_{3,τ_3} to $\mathcal{A}$, adds $(F_{3,\tau_3}, \xi_{3,\tau_3})$to L_3 and increments τ_3.

Oracle $O_{x,y}$: $\mathcal{A}$ inputs $m_{\tau_v} \in Z_p^*$. $\mathcal{B}$ chooses a new variable v_{τ_v} and sets $F_{1,\tau_1} \leftarrow v_{\tau_v}$, $F_{1,\tau_1+1} \leftarrow v_{\tau_v}(x + m_{\tau_v} y) + xy$. For $t \in \{0,1\}$, if $F_{1,\tau_1+t} = F_{1,u}$ for some $u \in \{0, ..., \tau_1 - 1 + t\}$, then $\mathcal{B}$ sets $\xi_{1,\tau_1+t} = \xi_{1,u}$; otherwise it sets ξ_{1,τ_1+t} to a random string in $\{0,1\}^* \setminus \{\xi_{1,0}, ..., \xi_{1,\tau_1-1+t}\}$. Finally $\mathcal{B}$ returns $(\xi_{1,\tau_1},\ \xi_{1,\tau_1+1})$ to $\mathcal{A}$ and adds $(F_{1,\tau_1},\ \xi_{1,\tau_1})$, $(F_{1,\tau_1+1},\ \xi_{1,\tau_1+1})$ to L_1, τ_1 is incremented 2, τ_v is incremented 1.

Eventually $\mathcal{A}$ stops and outputs $(m, \xi_{1,a}, \xi_{1,b})$, where $m \in Z_p^* \setminus \{m_1, ..., m_{\tau_v}\}$ and $0 \le a, b \le \tau_1$.

Analysis of $\mathcal{A}$'s Output. For $\mathcal{A}$'s output to be always correct, then $F_{1,b} - F_{1,a} \cdot (x + my) - xy = 0$ for any $(x, y, v_1, ..., v_{\tau_v})$, where $F_{1,a}$ $(F_{1,b})$ corresponds to $\xi_{1,a}$ $(\xi_{1,b})$. We now argue that it is impossible for $\mathcal{A}$ to achieve this.

$F_{1,i}$ has the following form according to the description above:

$F_{1,i} = c_{0,i} + c_{1,i}x + c_{2,i}y + \sum_k f_{k,i}v_k + \sum_k d_{k,i}[v_k(x + m_k y) + xy]$, where $\sum_k$ denotes $\sum_{1 \le k \le \tau_v}$ for simplicity.

It follows that

$F_{1,b} - F_{1,a}(x+my) - xy = c_{0,b} + (c_{1,b} - c_{o,a})x + (c_{2,b} - mc_{0,a})y + \sum_k f_{k,b}v_k + \sum_k (d_{k,b} - f_{k,a})v_k x + \sum_k (d_{k,b}m_k - f_{k,a}m)v_k y + (\sum_k d_{k,b} - c_{2,a} - mc_{1,a} - 1)xy - c_{1,a}x^2 - \sum_k d_{k,a}v_k x^2 - \sum_k (m_k - m)d_{k,a}v_k xy - (\sum_k d_{k,a})x^2 y - (\sum_k d_{k,a}m)xy^2 - mc_{2,a}y^2 - \sum_k d_{k,a}m_k m v_k y^2$.

For the above function to be zero for any $(x, y, v_1, ..., v_{\tau_v})$, all the coefficients are to be zero, then

$$d_{k,a} = 0, f_{k,b} = 0, d_{k,b} = f_{k,a}, d_{k,b}m_k = f_{k,a}m \tag{1}$$

$$c_{0,b} = 0, c_{1,b} = c_{0,a}, c_{2,b} = mc_{0,a}, c_{1,a} = 0, c_{2,a} = 0, \sum_k d_{k,b} = c_{2,a} + mc_{1,a} + 1. \tag{2}$$

We have $d_{k,b} = f_{k,a} = 0$ from (1) because $m \ne m_k$ for any k. We also have $\sum_k d_{k,b} = 1$ from (2), which is a contradiction. Thus we conclude that $\mathcal{A}$'s success depends solely on his luck when $(x, y, v_1, ..., v_{\tau_v})$ is instantiated.

Analysis of $\mathcal{B}$'s Simulation. At this point $\mathcal{B}$ chooses random (x^*, y^*, v_1^*, ..., $v_{\tau_v}^*$). $\mathcal{B}$ now tests if its simulation was perfect by checking (3) and (5), i.e., if the instantiation $(x^*, y^*, v_1^*, ..., v_{\tau_v}^*)$ does not create any equality relation among the polynomials that was not revealed by the random strings provided to $\mathcal{A}$. $\mathcal{B}$ also tests whether or not $\mathcal{A}$'s output was correct by checking (6).

$$F_{1,i}(x^*, y^*, \{v_k^*\}) - F_{1,j}(x^*, y^*, \{v_k^*\}) = 0, \text{ for some } i, j \text{ s.t. } F_{1,i} \neq F_{1,j} \quad (3)$$

$$F_{2,i}(x^*, y^*) - F_{2,j}(x^*, y^*) = 0, \text{ for some } i, j \text{ s.t. } F_{2,i} \neq F_{2,j} \quad (4)$$

$$F_{3,i}(x^*, y^*, \{v_k^*\}) - F_{3,j}(x^*, y^*, \{v_k^*\}) = 0, \text{ for some } i, j \text{ s.t. } F_{3,i} \neq F_{3,j} \quad (5)$$

$$F_{1,b}(x^*, y^*, \{v_k^*\}) - F_{1,a}(x^*, y^*, \{v_k^*\})(x^* + my^*) - x^*y^* = 0 \quad (6)$$

Thus $\mathcal{A}$'s overall success is bounded by the probability that any of the above equation holds.

We observe that $F_{1,i}$ is non-trivial polynomial of degree at most 2, $F_{2,i}$ at most 1, $F_{3,i}$ at most 4, the function of (6) at most 3.

For fixed i, j, the first case occur with probability $\leq 2/p$, the second case $\leq 1/p$, the third case $\leq 4/p$. The fourth case happens with probability $\leq 3/p$. Summing over all (i, j) pairs in each case, we bound $\mathcal{A}$'s overall success probability $\varepsilon \leq \binom{\tau_1}{2}\frac{2}{p} + \binom{\tau_2}{2}\frac{1}{p} + \binom{\tau_3}{2}\frac{4}{p} + \frac{3}{p}$, i.e. $\varepsilon \leq O(Q_G^2/p)$, since $\tau_1 + \tau_2 + \tau_3 \leq Q_G + 6$.

B Detail of PK_2 in Scheme 2

The detail of the proof of knowledge
$\pi = \mathrm{PK}_2\{(\alpha, \beta) : [e(V', \tilde{g})e(U'Y, \widetilde{X})^{-1}]^\beta = e(U', W_\beta) \wedge e(g, W_\alpha)^\beta e(g, h)^{-\alpha\beta} = e(pk_i, W_\beta)\}$:

- Prover selects $(k_1, k_2) \overset{R}{\longleftarrow} Z_p^* \times Z_p^*$, calculates $R_1 = [e(V', \tilde{g})e(U'Y, \widetilde{X})^{-1}]^{k_1}$, $R_2 = e(g, W_\alpha)^{k_1} e(g, h)^{-k_2}$, and sends R_1, R_2 to Challenger. Challenger replies with a random $c \overset{R}{\longleftarrow} Z_p^*$.
- Prover calculates $s_1 = k_1 + c\beta$, $s_2 = k_2 + c\alpha\beta$, and sends them to Challenger.
- Challenger checks whether the following equations are satisfied:
  $R_1 = [e(V', \tilde{g})e(U'Y, \widetilde{X})^{-1}]^{s_1} e(U', W_\beta)^{-c}$,
  $R_2 = e(g, W_\alpha)^{s_1} e(g, h)^{-s_2} e(pk_i, W_\beta)^{-c}$.
  Challenger accepts (s_1, s_2) if the above check passes, rejects it otherwise.

The following Lemma can be proved similarly to corresponding Lemmas or Theorems in [10, 20, 5].

Lemma 4. *The above interactive protocol π is statistical honest verifier zero-knowledge and sound, under Discrete Logarithm assumption in group G_1, G_2 and G_3.*

Proof. Zero-knowledge is easy to see. The soundness is as follows:

Soundness: By resetting Prover under the same random inputs, an honest verifier can get (s_1, s_2, c) and (s_1', s_2', c') where $s_j' \neq s_j$, $j = 1, 2$, $c' \neq c$.

Let $\Delta s_j = s_j - s_j'$, $j = 1, 2$, $\Delta c = c' - c$, then
$[e(V', \tilde{g})e(U'Y, \widetilde{X})^{-1}]^{\Delta s_1} = e(U', W_\beta)^{\Delta c}$,
$e(g, W_\alpha)^{\Delta s_1} e(g, h)^{-\Delta s_2} = e(pk_i, W_\beta)^{\Delta c}$.

Set $\beta' = \Delta s_1/\Delta c \bmod p$, $\alpha' = \Delta s_2/\Delta s_1 \bmod p$, then $W_\alpha = \sigma_i h^{\alpha'}$, $W_\beta = A_i^{\beta'}$, it follows that (A_i, σ_i) is easy to decide.

C Proof of Theorem 2

C.1 Preliminaries

Lemma 5 (Generalized Forking Lemma [30]). *Consider a PPT algorithm $\mathcal{P}$, a PPT predicate Q, and a hash function $\mathcal{H}$ with range $\{0,1\}^k$ thought of as a random oracle. The predicate Q satisfies that $Q(x) = \top \Rightarrow \{x = (\rho_1, c, \rho_2) \wedge c = \mathcal{H}(\rho_1)\}$. $\mathcal{P}$ is allowed to ask queries on $\mathcal{H}$ and $\mathcal{R}$, where $\mathcal{R}$ is a process that given (t, c) reprograms $\mathcal{H}$ so that $\mathcal{H}(t) = c$, and it is assumed that $\mathcal{P}$ behaves in such a way that queries (t, c) to $\mathcal{R}$ adhere to the following conditions:*
- *c is uniformly distributed over $\{0,1\}^k$.*
- *The probability of the occurrence of a specific $t = t_0$ is upper bounded by $2/2^k$.*

Suppose that $\mathcal{P}^{\mathcal{H},\mathcal{R}}(\mathsf{param})$ returns a x such that $Q(x) = \top$ with non-negligible probability $\epsilon \geq 10(q_R + 1)(q_R + q_H)/2^k$, where q_R, q_H are numbers of queries to $\mathcal{R}$ and $\mathcal{H}$ respectively. Then there exists a PPT $\mathcal{P}'$ so that if $y \leftarrow \mathcal{P}'(\mathsf{param})$ it holds with probability 1/9 that (1) $y = (\rho_1, c, \rho_2, c', \rho_2')$, (2) $Q(\rho_1, c, \rho_2) = \top$ and $Q(\rho_1, c', \rho_2') = \top$, (3) $c \neq c'$. The probabilities are taken over the choices for $\mathcal{H}$, the random coin tosses of $\mathcal{P}$ and the random choice of the public parameters param.

C.2 Proof of Theorem 2

Proof. Suppose $\mathcal{A}$ is an adversary breaking the traceability of Scheme 2, we can construct an adversary $\mathcal{B}$ breaking Assumption 1 as follows:

$\mathcal{B}$ is given $X, Y \in G_1$, $\widetilde{X}, \widetilde{Y} \in G_2$, bilinear map e, and G_3, as well as an oracle $O_{x,y}$. The task of $\mathcal{B}$ is to figure out a triple (m, U, V), where $U = g^r$, $V = g^{r(x+my)+xy}$ and m is never queried to $O_{x,y}$. $\mathcal{B}$ transfers (g, $\tilde{g}$, e, X, Y, $\widetilde{X}$, $\widetilde{Y}$, p, $h \overset{R}{\longleftarrow} G_2$) to $\mathcal{A}$ as public keys, sets $CU \leftarrow \varnothing$, $HU \leftarrow \varnothing$, $Reg \leftarrow \varnothing$, and answers queries from $\mathcal{A}$ as follows.
- When $\mathcal{A}$ queries $CrptU(i, pk)$, $\mathcal{B}$ sets the user public key $pk_i = pk$, $CU \leftarrow CU \cup \{i\}$.
- When $\mathcal{A}$ queries $SndToI(i, M_{in})$, where $M_{in} = \widetilde{C}_i$, $\mathcal{A}$ should run an interactive proof of knowledge of $\tilde{s}_i$ that $\widetilde{C}_i = Y^{\tilde{s}_i}$ with $\mathcal{B}$, in which $\mathcal{B}$ provides a random challenge c_1. Now $\mathcal{B}$ rewinds $\mathcal{A}$ and provides a second random challenge $c_2 \neq c_1$. $\tilde{s}_i$ can be extracted due to the soundness of the above proof of knowledge. The detail of the rewind technique is referred to Section 6 of [26].

  $\mathcal{B}$ then chooses $s_i \overset{R}{\longleftarrow} Z_p^*$, sets $r_1 = s_i - \tilde{s}_i$, sends s_i to $O_{x,y}$, which outputs a pair (U_i, V_i). $\mathcal{B}$ returns (r_1, U_i, V_i) to $\mathcal{A}$ as a reply.

  $\mathcal{A}$ can obtain $s_i = \tilde{s}_i + r_1$, and should send a $A_i = \widetilde{Y}^{s_i}$ and σ_i (a $\mathcal{S}$ signature on A_i), as well as a proof of equality of $\log_{\widetilde{Y}} A_i$ and $\log_Y \widetilde{C}_i Y^{r_1}$ to $\mathcal{B}$. $\mathcal{B}$ checks that (A_i, σ_i) is valid under public key pk_i, and stores (A_i, σ_i) in Reg if that is the case.
- When $\mathcal{A}$ queries $CrptOA$, $\mathcal{B}$ returns Reg=$\{(A_i, \sigma_i)\}$.

– When $\mathcal{A}$ queries random oracle $\mathcal{H}$, $\mathcal{B}$ chooses a random $c \in Z_p^*$ and replies consistently, i.e., if it is a duplicate query, $\mathcal{B}$ always replies with the same random value chosen for the first query.

In the end, if $\mathcal{A}$ wins with non-negligible probability, which means $\mathcal{A}$ outputs a (m, U, V, c, s, RL) that OA can not find the identity of the signer (Case 1), or OA can find the identity of the signer, but can not prove that to a judger (Case 2), i.e.,

Case 1	Case 2
$c = \mathcal{H}(e(U, \widetilde{Y})^s[e(UY, \widetilde{X})e(V, \tilde{g})^{-1}]^c,$ $V, X, Y, \widetilde{X}, \widetilde{Y}, m),$ $e(V, \tilde{g}) \neq e(U, \widetilde{X}A_j)e(X, \widetilde{Y}), \forall A_j \in RL.$ $e(V, \tilde{g}) \neq e(U, \widetilde{X}A_i)e(X, \widetilde{Y}), \forall A_i \in Reg.$	$c = \mathcal{H}(e(U, \widetilde{Y})^s[e(UY, \widetilde{X})e(V, \tilde{g})^{-1}]^c, V, X, Y, \widetilde{X}, \widetilde{Y}, m),$ $e(V, \tilde{g}) \neq e(U, \widetilde{X}A_j)e(X, \widetilde{Y}), \forall A_j \in RL,$ $e(V, \tilde{g}) = e(U, \widetilde{X}A_i)e(X, \widetilde{Y}), \exists A_i \in Reg,$ i has been queried to $CrptU$, and $(i, \pi) \leftarrow$ Open (Reg, m, U, V, s, c), Judge$(RL, i, \pi) = 0$.

The latter case is negligible because the soundness and correctness of Open, and existential unforgeability of the standard signature $\mathcal{S}$. So with a non-negligible probability, $\mathcal{A}$ will output a (m, U, V, c, s, RL) satisfying the former condition.

Apply Lemma 5 to $\mathcal{A}$, where GVer is the predicate, $\mathcal{B}$ will get a $\mathcal{A}'$ that outputs (m, U, V, c, s, RL) and (m, U, V, $c' \neq c$, s', RL). Thus $\mathcal{B}$ can get $s_i = \frac{s-s'}{c-c'}$ that $e(V, \tilde{g}) = e(U, \widetilde{X}\widetilde{Y}^{s_i})e(X, \widetilde{Y})$, where $\widetilde{Y}^{s_i} \notin Reg$, i.e., s_i is never queried to oracle $O_{x,y}$, and (U, V) must has the form of $(g^r, g^{r(x+s_i y)+xy})$ for some $r \in Z_p^*$, thus Assumption 1 is broken.

D Proof of Lemma 2

Proof. $\mathcal{B}$ is given (A, B, Z), where $A = g^a$, $B = g^b$, $Z = g^{ab}$ or $Z = g^c$, $(a, b, c) \xleftarrow{R} Z_p^{*3}$. The task of $\mathcal{B}$ is to distinguish which is the case for Z, i.e., output a guess $\omega' \in \{0, 1\}$ of ω, where $\omega = 1$ denotes $Z = g^{ab}$ and $\omega = 0$ denotes $Z = g^c$. $\mathcal{B}$ solves the challenge by interacting with $\mathcal{A}$ as follows.

$\mathcal{B}$ simulates Setup as follows:

1. $\mathcal{B}$ sets group public key: $X = g^x$, $Y = g^y$, $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$, $h \xleftarrow{R} G_2$, where $(x, y) \xleftarrow{R} Z_p^* \times Z_p^*$.

2. Suppose the total number of group members is n, $\mathcal{B}$ picks $i^* \xleftarrow{R} [1, n]$, sets $s_{i^*} = a$, $U_{i^*} = g^r$, $V_{i^*} = XA^{yr}g^{xy}$, $A_{i^*} = A$, $\sigma_{i^*} = A_{i^*}^z$, $pk_{i^*} = g^z$, where $(r, z) \xleftarrow{R} Z_p^* \times Z_p^*$, $HU \leftarrow \{i^*\}$, $Reg \leftarrow \{(A_{i^*}, \sigma_{i^*})\}$.

• When $\mathcal{A}$ queries $CrptU(i, pk)$, if $i \notin HU \cup CU$, $\mathcal{B}$ sets the user public key $pk_i = pk$, $CU \leftarrow CU \cup \{i\}$.

• When $\mathcal{A}$ queries $SndToI(i, M_{in})$, where $M_{in} = \widetilde{C}_i$, $\mathcal{A}$ should run an interactive proof of knowledge of $\tilde{s}_i$ that $\widetilde{C}_i = Y^{\tilde{s}_i}$ with $\mathcal{B}$, in which $\mathcal{B}$ provides a random challenge c_1. Now $\mathcal{B}$ rewinds $\mathcal{A}$ and provides a second random challenge $c_2 \neq c_1$. $\tilde{s}_i$ can be extracted due to the soundness of the above proof of knowledge. The detail of the rewind technique is referred to Section 6 of [26].

$\mathcal{B}$ then chooses $s_i \xleftarrow{R} Z_p^*$, sets $r_1 = s_i - \tilde{s}_i$, sends s_i to $O_{x,y}$, which outputs a pair (U_i, V_i). $\mathcal{B}$ returns (r_1, U_i, V_i) to $\mathcal{A}$ as a reply.

$\mathcal{A}$ can obtain $s_i = \tilde{s}_i + r_1$, and should send a $A_i = \widetilde{Y}^{s_i}$ and σ_i (a $\mathcal{S}$ signature on A_i), as well as a proof of equality of $\log_{\widetilde{Y}} A_i$ and $\log_Y \widetilde{C}_i Y^{r_1}$ to $\mathcal{B}$. $\mathcal{B}$ checks that (A_i, σ_i) is valid under public key pk_i, and stores (A_i, σ_i) in *Reg* if that is the case.

• When $\mathcal{A}$ queries random oracle $\mathcal{H}$, $\mathcal{B}$ answers randomly and consistently.

• When $\mathcal{A}$ queries $USK(i)$, if $i \neq i^*$, $\mathcal{B}$ replies with (s_i, U_i, V_i) and $CU \leftarrow CU \cup \{i\}$; if $i = i^*$, $\mathcal{B}$ aborts and outputs a random guess $\omega' \in_R \{0, 1\}$.

• When $\mathcal{A}$ queries $Revoke(i)$ for revocation token of i, if $i \neq i^*$, $\mathcal{B}$ responds with A_i; if $i = i^*$, $\mathcal{B}$ aborts and outputs a random guess $\omega' \in_R \{0, 1\}$.

• When $\mathcal{A}$ queries $GSig(i, m)$ for a signature on m signed by member $i \neq i^*$, $\mathcal{B}$ generates the signature exactly as algorithm GSig since the secret key (s_i, U_i, V_i) is known; if $i = i^*$, $\mathcal{B}$ calculates $U \leftarrow U_{i^*} g^{r'}$, $V \leftarrow V_{i^*} (A^y g^x)^{r'}$, where $r' \xleftarrow{R} Z_p^*$, and simulates a proof of knowledge of a. In case duplicate hash queries haven been made, $\mathcal{B}$ aborts and outputs a random guess $\omega' \in_R \{0, 1\}$.

• When $\mathcal{A}$ queries $Open(m, \sigma)$, $\mathcal{B}$ does exactly as in algorithm Open, since it knows the content of *Reg*.

• When $\mathcal{A}$ queries $WReg(i, .)$, $\mathcal{B}$ sets $Reg_i = s$ if i has not been recorded in *Reg*.

• When $\mathcal{A}$ queries $Ch(., i_0, i_1, m)$, i_0, i_1 should never have been queried to *USK* and *Revoke*, $\mathcal{B}$ picks $\phi \xleftarrow{R} \{0, 1\}$ with uniform probability, if $i_\phi \neq i^*$ or $i^* \notin \{i_0, i_1\}$, $\mathcal{B}$ aborts and outputs a random guess $\omega' \in_R \{0, 1\}$, otherwise generates the following challenge

$$U = B, V = B^x Z^y g^{xy}, \tau, \tag{7}$$

where τ is a simulation of $\mathrm{SK}_1\{s_{i_\phi} : e(U, \widetilde{Y})^{s_{i_\phi}} = e(V, \tilde{g}) e(UY, \widetilde{X})^{-1}\}\{m\}$.

If $Z = g^{ab}$, then $U = g^b$, $V = g^{b(x+a)y+xy}$, the challenge is a real group signature on m by i_ϕ.

If $Z = g^c$, then V is independent with i_0 or i_1, there is no better method for $\mathcal{A}$ to win than guessing ϕ totally.

$\mathcal{A}$ outputs $\phi' \in \{0, 1\}$ after it is given the challenge (7). $\mathcal{B}$ outputs $\omega' = 1$ if $\phi' = \phi$ (implying $Z = g^{ab}$), outputs $\omega' = 0$ otherwise (implying $Z = g^c$).

The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_{\mathcal{A}}^{anon} = \Pr\{\phi' = 1 | \phi = 1\} - \Pr\{\phi' = 1 | \phi = 0\}$. The advantage of $\mathcal{B}$ is defined as $\mathrm{Adv}_{\mathcal{B}}^{ddh} = \Pr\{\omega' = 1 | \omega = 1\} - \Pr\{\omega' = 1 | \omega = 0\}$. It follows that

$$\begin{aligned}\mathrm{Adv}_{\mathcal{B}}^{ddh} &= (\Pr\{\omega' = 1 | \overline{\mathsf{abort}}, \omega = 1\} - \Pr\{\omega' = 1 | \overline{\mathsf{abort}}, \omega = 0\}) \Pr\{\overline{\mathsf{abort}}\} \\ &= (\Pr\{\phi' = \phi | \omega = 1\} - \Pr\{\phi' = \phi | \omega = 0\}) \Pr\{\overline{\mathsf{abort}}\} \\ &= \frac{1}{2} \mathrm{Adv}_{\mathcal{A}}^{anon} \Pr\{\overline{\mathsf{abort}}\} \geq \frac{\epsilon}{2} (\frac{1}{n} - \frac{q_H q_{sig}}{p}),\end{aligned}$$

where $\Pr\{\overline{\mathsf{abort}}\} \geq \frac{1}{n} - \frac{q_H q_{sig}}{p}$ because $\overline{\mathsf{abort}}$ happens if only $\mathcal{B}$ has chosen the right random i^* from $[1, n]$, ϕ from $\{0, 1\}$ and duplicate hash requests have not occurred.

E Proof of Lemma 3

Proof. $\mathcal{B}$ is given $(A, B, Z) \in G_1^3$, where $A = g^a$, $B = g^b$, $Z = g^{ab}$ or $Z = g^c$, $(a, b, c) \overset{R}{\longleftarrow} Z_p^{*3}$. The task of $\mathcal{B}$ is to distinguish which is the case for Z, i.e., output a guess $\omega' \in \{0, 1\}$ of ω, where $\omega = 1$ denotes $Z = g^{ab}$ and $\omega = 0$ denotes $Z = g^c$. $\mathcal{B}$ solves the challenge by interacting with $\mathcal{A}$ as follows.

$\mathcal{B}$ simulates Setup as follows: $\mathcal{B}$ sets group public key: $X = g^x$, $Y = g^y$, $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$, where $(x, y) \overset{R}{\longleftarrow} Z_p^* \times Z_p^*$. Suppose the total number of group members is n, total time period is t, $\mathcal{B}$ picks $i^* \overset{R}{\longleftarrow} [1, n]$, and sets $s_{i^*} = a$, $U_{i^*} = g^r$, $V_{i^*} = XA^{yr}g^{xy}$, $A_{i^*} = A$, $\sigma_{i^*} = A_{i^*}^z$, $pk_{i^*} = g^z$, where $(r, z) \overset{R}{\longleftarrow} Z_p^* \times Z_p^*$, $HU \leftarrow \{i^*\}$, $Reg \leftarrow \{(A_{i^*}, \sigma_{i^*})\}$.

At the beginning of time period $j \in [1, t]$, $\mathcal{B}$ calculates $h_j = \widetilde{Y}^{r_j}$, $r_j \overset{R}{\longleftarrow} Z_p^*$.

•When $\mathcal{A}$ queries $CrptU(i, pk)$, if $i \notin HU \cup CU$, $\mathcal{B}$ sets the user public key $pk_i = pk$, $CU \leftarrow CU \cup \{i\}$.

• When $\mathcal{A}$ queries $SndToI(i, M_{in})$, where $M_{in} = \widetilde{C}_i$, $\mathcal{A}$ should run an interactive proof of knowledge of $\tilde{s}_i$ that $\widetilde{C}_i = Y^{\tilde{s}_i}$ with $\mathcal{B}$, in which $\mathcal{B}$ provides a random challenge c_1. Now $\mathcal{B}$ rewinds $\mathcal{A}$ and provides a second random challenge $c_2 \neq c_1$. $\tilde{s}_i$ can be extracted due to the soundness of the above proof of knowledge. The detail of the rewind technique is referred to Section 6 of [26].

$\mathcal{B}$ then chooses $s_i \overset{R}{\longleftarrow} Z_p^*$, sets $r_1 = s_i - \tilde{s}_i$, sends s_i to $O_{x,y}$, which outputs a pair (U_i, V_i). $\mathcal{B}$ returns (r_1, U_i, V_i) to $\mathcal{A}$ as a reply.

$\mathcal{A}$ can obtain $s_i = \tilde{s}_i + r_1$, and should send a $A_i = \widetilde{Y}^{s_i}$ and σ_i (a $\mathcal{S}$ signature on A_i), as well as a proof of equality of $\log_{\widetilde{Y}} A_i$ and $\log_Y \widetilde{C}_i Y^{r_1}$ to $\mathcal{B}$. $\mathcal{B}$ checks that (A_i, σ_i) is valid under public key pk_i, and stores (A_i, σ_i) in Reg if that is the case.

• When $\mathcal{A}$ queries random oracle $\mathcal{H}$, $\mathcal{B}$ answers $\mathcal{A}$'s hash queries randomly and consistently.

•When $\mathcal{A}$ queries $GSig(m, i, j)$, where $i \neq i^*$, $\mathcal{B}$ can generate the signature because it knows the member secret key of i.

When $i = i^*$, $\mathcal{B}$ can generate the signature correctly except not knowing s_{i^*}, so it can simulate a proof of knowledge of s_{i^*}.

In case duplicate hash queries haven been made, $\mathcal{B}$ aborts and outputs a random guess $\omega' \in_R \{0, 1\}$.

•When $\mathcal{A}$ queries $USK(i)$, $\mathcal{B}$ replies (s_i, U_i, V_i) when $i \neq i^*$, $\mathcal{B}$ aborts and outputs a random guess $\omega' \overset{R}{\longleftarrow} \{0, 1\}$ when $i = i^*$.

•When $\mathcal{A}$ queries $Revoke(i, j)$, $\mathcal{B}$ computes and returns the revocation token of i at time period j according to the following formula

$$B_{ij} \triangleq h_j^{s_i} = \begin{cases} g^{s_i r_j y}, & i \neq i^* \\ A^{r_j y}, & i = i^* \end{cases}$$

•When $\mathcal{A}$ queries $Ch(m, i_0, i_1, j)$, i_0, i_1 should never have been queried to *USK* and *Revoke* at time period j. $\mathcal{B}$ picks $\phi \xleftarrow{R} \{0,1\}$ randomly, if $i_\phi \neq i^*$, aborts and outputs a random guess $\omega' \xleftarrow{R} \{0,1\}$. Otherwise, $\mathcal{B}$ generates the following challenge

$$U = B^{r'}, V = B^{r'x} Z^{r'y} g^{xy}, S = e(A, \tilde{g})^{yr'' r_j}, T = B^{r''}, \tau,$$

where $(r', r'') \xleftarrow{R} Z_p^* \times Z_p^*$, τ is a simulated proof of (a, br'', abr'').

If $Z = g^{ab}$, the challenge is a valid group signature signed by i^* at time j in random oracle model.

If $Z = g^c$, the challenge is a valid group signature signed by a group member $\hat{i}$ whose secret key $s_{\hat{i}} = c/b$ at time period j, which is non-preferable for guessing i_0 or i_1, there is no better method for $\mathcal{A}$ to win than guessing ϕ totally.

$\mathcal{B}$ outputs $\omega' = 1$ if $\phi' = \phi$ (implying $Z = g^{ab}$), outputs $\omega' = 0$ otherwise (implying $Z = g^c$). Thus $\mathrm{Adv}_{\mathcal{B}}^{ddh} = \frac{1}{2}\mathrm{Adv}_{\mathcal{A}}^{anon}\Pr\{\overline{\mathsf{abort}}\} \geq \frac{\epsilon}{2}(\frac{1}{n} - \frac{q_H q_{sig}}{p})$, because $\overline{\mathsf{abort}}$ happens if only $\mathcal{B}$ has chosen the right random i^* from $[1, n]$, ϕ from $\{0,1\}$ and duplicate hash requests have not occurred.