Group Signature with Constant Revocation Costs for Signers and Verifiers

Chun-I Fan[1], Ruei-Hau Hsu[1], and Mark Manulis[2]

[1] Computer Science Engineering National Sun Yat-sen University
Kaohsiung, Taiwan
cifan@faculty.cse.nsysu.edu.tw,
xyzhsu@gmail.com
[2] Cryptographic Protocols Group, Department of Computer Science
TU Darmstadt & CASED, Germany
mark@manulis.eu

Abstract. Membership revocation, being an important property for applications of group signatures, represents a bottleneck in today's schemes. Most revocation methods require linear amount of work to be performed by unrevoked signers or verifiers, who usually have to obtain fresh update information (sometimes of linear size) published by the group manager. We overcome these disadvantages by proposing a novel group signature scheme, where computation costs for unrevoked signers and potential verifiers remain constant, and so is the length of the update information that must be fetched by these parties from the data published by the group manager. We achieve this complexity by increasing the amount of work at the group manager's side, which growths quadratic with the total number of members. This increase is acceptable since algorithms of the group manager are typically executed on resourceful devices. Our scheme uses a slightly modified version of the pairing-based dynamic accumulator, introduced by Camenisch, Kohlweiss, and Soriente (PKC 2009), which we implicitly combine with the short (non-revocable) group signature scheme by Boneh, Boyen, and Shacham (CRYPTO 2004). We prove that our revocable scheme satisfies the desired security properties of anonymity, traceability, and non-frameability in the random oracle model, although for better efficiency we resort to a somewhat stronger hardness assumption.

1 Introduction

Revocable Group Signatures. Group Signatures (GS) [17] protect anonymity of signers, who are considered as members of the group, managed by a Group Manager (GM), and who can sign on behalf of the group, while remaining traceable (identifiable) only by the group manager. The tracing ability of the group manager is often used in case of dispute, e.g. if the signer misused his signing rights. In many situations, identification of the misbehaving signer should also lead to the revocation of his signing abilities. Group signatures, allowing the group manager to additionally revoke the signing rights of group members are called *revocable*. A revoked group member should no longer be able to produce valid group signatures. In traditional public key infrastructures revocation is typically handled by certification authorities that publish unique information

D. Lin, G. Tsudik, and X. Wang (Eds.): CANS 2011, LNCS 7092, pp. 214–233, 2011.

about the revoked certificates and which is then used by verifiers for checking the validity of certificates. In group signature schemes, however, revocation process must take into account the anonymity requirements offered by these schemes. Currently, there exist two main approaches for revocation: The first approach, originated by Camenisch and Lysyanskaya [16], uses so-called *dynamic accumulators*, where each secret signing key of an unrevoked group member contains a *witness* associated to the public accumulator value; upon revocation of some group member GM updates the accumulator value and publishes some update information, which in turn allows remaining group members to update their witnesses. The accumulator value is also used as input to the verification procedure. The second approach, termed *verifier-local revocation (VLR)*, originated by Boneh and Shacham [12], requires from the group manager to release a revocation token associated with the revoked member; all published revocation tokens are then used as input to the verification procedure, whereas unrevoked signers need not to update their secret signing keys.

Revocation and Security. Revocation of signing rights should not compromise the basic security properties of GS schemes. Modern GS schemes are proven secure in (variants of) the security model, introduced by Bellare, Micciancio, and Warinschi [4], that defined two main requirements, namely *full-anonymity* and *full-traceability*, capturing many previously stated (sometimes informally described) security properties, with regard to the anonymity of signers, unlinkability of their signatures, unforgeability of signatures, protection against framing attacks, in particular in the presence of malicious coalitions and possibly corrupted group managers. Security definitions from [4] were designed for static schemes and later refined in [5] to address caveats with full-traceability in dynamic schemes; in particular, full-traceability was relaxed to *traceability* and an additional requirement of *non-frameability* was used to address possible corruptions of the group manager in case of framing attacks. We observe that support for revocation introduces dynamic behavior, even for schemes that do not provide support for the dynamic admission of new group members. In schemes with VLR property an additional concern arises due to the implicit opening mechanism that is inherent to all these schemes, namely published revocation tokens also invalidate signatures that were produced by the revoked signer before the revocation took place, and by this introduce linkability amongst all signatures of that signer. More recent VLR schemes were enriched with the additional anonymity protection in form of *BU-anonymity* [25], where BU stands for "backward unlinkability", aiming to prevent linkability of signatures that were produced by the revoked signer while he was a legitimate member of the group. Note that BU-anonymity in VLR schemes is typically achieved by splitting the lifetime of the GS scheme in distinct time intervals and revoking a particular signer in all subsequent time intervals, starting with the interval in which his revocation took place for the first time.

Revocation Costs and Their Impact. Ideally, support for revocation should not introduce significant overhead with respect to the computational complexity, for at least the most frequent operations on the side of members and verifiers, namely signature generation and verification. Support for revocation should not significantly increase the size of main parameters, such as the length of group public keys and signatures. Note

that many modern group signature schemes (without revocation) offer constant complexity for these parameters. However, all existing revocable GS schemes that either use dynamic accumulators or utilize VLR introduce linear costs in one or another way. For example, the length of (public) update information used by signers and/or verifiers is often linear in at least the number of revoked members: In schemes with dynamic accumulators public information is used by each (unrevoked) signer to update his secret signing key, resulting in the linear amount of computations on the signer's side. In schemes with VLR property public information contains revocation tokens of revoked signers, which are only used in the verification procedure, resulting in the linear amount of computations on the verifier's side (to perform the revocation check). Designing a revocable group signature scheme that would offer constant costs for unrevoked members to update their secret signing keys, constant costs for verifiers to perform the revocation check, and constant length of the update information, published by the group manager remains an open problem[1] so far.

2 Prior Work on Revocable Group Signatures

Revocation in group signature schemes was identified as a desirable property by Ateniese and Tsudik [1], who suggested that revoked signers should no longer be able to generate valid group signatures, while their earlier group signatures must remain anonymous. Thereafter, many revocable group signature schemes were built, e.g. [3, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 23, 24, 25, 26, 31, 32]. The first popular approach for handling revocation in group signature schemes is based on dynamic accumulators, originally applied by Camenisch and Lysyanskaya [16], and later adopted to further constructions [20, 15, 30]. With this approach the group manager publishes an updated accumulator value together with some update information, which is processed by unrevoked signers prior or during the computation of subsequent group signatures.

An alternative revocation method is used in group signatures with VLR property, e.g. [12, 13, 19, 25, 26, 31, 32]. These schemes require from the group manager to publish a revocation list containing partial information about the secret keys of revoked signers. This list is then used as input to the verification algorithm, which performs the revocation check by processing all of its entries in the worst case. Unrevoked signers no longer need to update their signing keys. Many earlier VLR constructions were not able to offer anonymity with regard to group signatures, output in the past by the meanwhile revoked signers. This property, known as BU-anonymity, was introduced in [13] and further considered in [12, 19, 25, 26, 31, 32]. Many of existing revocable group signature schemes [3, 11, 12, 13, 14, 15, 16, 19, 25, 26, 31, 32] have linear computation complexity for the generation and/or verification of group signatures, either $O(N)$ with N being the number of group members, or $O(R)$ with R being the number of revoked members.

[1] Jin et al. [18] claim that their revocable group signature scheme has constant costs with regard to signing/verifying and lengths of signatures, group public key, and individual secret signing keys. A closer inspection of their scheme (which was also not proven secure) reveals, however, that one of the components published as revocation information is linear in the number of group members and that all these components must be fetched by unrevoked signers to perform the signing operation, resulting in its linear computation costs.

There exist, however, several more efficient constructions: The scheme by Nakanischi et al. [23], which improves upon [24], partitions the entire group into several subgroups, offering constant costs for signature generation and verification, yet requiring the signers to fetch public update information of size $O(N)$. The scheme by Nakanishi and Funabiki [20] offers revocation for larger groups by reducing this length to $O(\sqrt{N})$ and a more recent scheme by Nakanishi et al. [21] succeeded in reducing this size further to $O(R)$. Camenisch, Kohlweiss, and Soriente [15] introduced an accumulator-based revocation mechanism that can also be applied to achieve revocation for group signatures. Applying the specified version of their accumulator would increase the signing costs by $O(R)$ modular multiplications (for signers to update their keys), which is still more efficient than previous accumulator-based constructions, where the linear amount of work was dedicated to costlier operations, e.g. modular exponentiations. Furthermore, the length of the group public key would be increased to $O(N)$ as unrevoked members would have to download their witnesses to perform the update of secret signing keys. There is an informal discussion in [15] according to which the update procedure for witnesses can also be offloaded to GM (or some third party). This tweak would lead to the constant costs for signers to perform the update procedure and possibly result in constant-size group public keys.

3 Our Results and Organization

Revocable Group Signature. Our work aims at further improving revocation costs in group signature schemes. The main idea is to consider the accumulator-based approach and let the group manager, who is typically responsible for the update of publicly available revocation information, to invest more computational resources (in comparison to other schemes), and by this minimize the costs of other parties (signers and verifiers). In particular, our revocable group signature (RGS) scheme achieves constant computation costs for signature generation, verification, and update of individual secret signing keys. It offers constant lengths for the the group public key, the output group signatures, and the amount of public information that each unrevoked signer must fetch in order to update own secret signing key. Note that algorithms of the group manager are typically executed on devices with rich computational resources so that increasing the computation costs for those algorithms is a rather minor issue.

In Table 1 we emphasize our improvement through the comparison with revocable schemes from [11, 14, 15, 20, 21, 23, 24]. The table does not include VLR schemes, e.g. [12, 19, 22, 25, 26, 31, 32], which all have an intrinsic limitation of $O(R)$ work in the verification procedure, typically dedicated to pairing evaluations or modular exponentiations. We compare sizes of the group public key, signatures, and update information, which is fetched by unrevoked signers and verifiers to keep an up-to-date view over the current composition of the group. We further indicate computational costs for unrevoked signers to perform signature generation, for verifiers to perform signature verification with revocation check, and for group managers to compute the update information, which is newly published after any revocation event. Table 1 uses the following timing notations: T_e denotes the amount of time to perform one modular exponentiation (in a suitable group), T_p is the time of one pairing evaluation, T_m is the time for one

modular multiplication, T_a measures one modular addition. Additionally, by N_s we denote the number of available subgroups (applies to [23, 20]) and l_n indicates the length of the RSA modulus (applies to [24], which is the predecessor of [23]). From the comparison we observe that, in general, our scheme performs by a magnitude better than the schemes from [11, 14, 15, 24]. Although computation costs for signature generation and verification of the schemes in [20, 21, 23] are comparable to ours, the size of the group public key in [21] and the length of the update information fetched by signers and verifiers in [20, 21, 23] are worse. In contrast to previous schemes, the group manager in our construction has quadratic costs of $O(NR)$. However, these costs refer to modular additions, which are known to be more efficient than modular multiplications.

Table 1. Comparison of Lengths and Computation Costs in Revocable Group Signature Schemes

	lengths			computation costs		
Schemes	GPK	GS	UI	Sign	Verify	GM Costs
[11]	$O(1)$	$O(1)$	$O(R)$	$O(R) \cdot T_e$	$O(1) \cdot (T_p + T_e)$	$O(R) \cdot T_e$
[14]	$O(1)$	$O(1)$	$O(R)$	$O(R) \cdot (T_e)$	$O(1) \cdot T_e$	$O(R) \cdot T_e$
[15]	$O(N)$	$O(1)$	$O(R)$	$O(R) \cdot T_m$	$O(1) \cdot (T_p + T_e)$	$O(R) \cdot T_m$
[20]	$O(1)$	$O(1)$	$O(N_s)$	$O(1) \cdot T_e$	$O(1) \cdot T_e$	$O(N_s)$
[21]	$O(\sqrt{N})$	$O(1)$	$O(R)$	$O(1) \cdot (T_p + T_e)$	$O(1) \cdot (T_p + T_e)$	$O(R) \cdot (T_p + T_e)$
[23]	$O(1)$	$O(1)$	$O(N_s)$	$O(1) \cdot T_e$	$O(1) \cdot T_e$	$O(N_s) \cdot T_e$
[24]	$O(1)$	$O(1)$	$O(N)$	$O(N/l_n) \cdot T_e$	$O(N/l_n) \cdot T_e$	$O(R) \cdot T_a$
Our RGS	$O(1)$	$O(1)$	$O(1)$	$O(1) \cdot (T_p + T_e)$	$O(1) \cdot (T_p + T_e)$	$O(NR) \cdot T_a$

GPK: group public key GS: group signature UI: update information

Our Techniques. Our RGS scheme implicitly applies a variant of the pairing-based dynamic accumulator by Camenisch, Kohlweiss, and Soriente [15] to the short pairing-based group signature scheme by Boneh, Boyen, and Shacham [11]. We stress that our revocable BBS scheme is different from the revocation mechanism that was discussed for the BBS scheme in [11] based on the ideas underlying the accumulator from [16], which is much less efficient than [15]. In our construction we slightly modify the process by which witnesses in the accumulator from [15] are updated and resort to a stronger hardness assumption for this purpose. Our modifications shift the computation costs for all updates from signers to the group manager and the new assumption, which we call Power Diffie-Hellman Exponent (PDHE) is stronger than the (non-standard) n-DHE assumption used in [15]. Under this assumption update of individual witnesses (which we call membership tokens in the scheme) requires quadratic amount $O(NR)$ of modular additions, performed by the group manager. At the same time we can obtain constant size for the group public key and for the amount of public update information, which an unrevoked signer must fetch prior to the generation of new group signatures. Upon revocation of some group member, the group manager will publish updated membership tokens of all unrevoked signers. In the signing phase an unrevoked signer will only use personal membership token (together with the secret signing key). In contrast, any verifier can check the validity of the group signature using the group public key and the up-to-date accumulator value. Taking into account the discussion in [15] we show how to achieve constant revocation costs for signers and constant lengths for the

group public key. Additionally, we show how to reduce the amount of public update information that unrevoked signers must fetch prior to updating their secret signing keys from linear to constant. We observe that our work extends the initial ideas from [15] for offloading the computation of updated witnesses to GM towards a concrete realization and with some optimizations: In particular, we show that quadratic computation costs on the group manager's side to update witnesses for all members can further be optimized, i.e. we can replace quadratic amount of multiplications that would be necessary for the scheme in [15] with the quadratic amount of (considerably more efficient) additions, by slightly changing the computation of updated witnesses on the group manager's side. This modification, however, requires a slightly stronger hardness assumption to prove the non-frameability property of our scheme.

Organization. We proceed as follows. In Section 4, we recall the setting of bilinear groups and discuss several number-theoretic assumptions used in our work, including our PDHE assumption, which we introduce as a stronger variant of n-DHE from [15]. In Section 5, we describe definitions and security model for revocable group signature schemes. Section 6 provides high-level overview and full specification of our RGS scheme, whose security we prove in Section 7.

4 Preliminaries

4.1 Bilinear Groups

We will work in the pairing-based setting and thus recall the notion of bilinear groups:

1. $\mathbb{G}$, $\mathbb{G}'$, and $\mathbb{G}_{\mathbb{T}}$ are cyclic groups, all of prime order p;
2. g_1 is a generator of $\mathbb{G}$; g_2 is a generator of $\mathbb{G}'$;
3. $e : \mathbb{G}\times\mathbb{G}' \to \mathbb{G}_T$ is an efficiently computable map with the following two properties:
   - Bilinear: for all $u \in \mathbb{G}, v \in \mathbb{G}'$, $a, b \in Z_p^*$: $e(u^a, v^b) = e(u, v)^{ab}$.
   - Non-degenerate: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$.

4.2 Hardness Assumptions

Here we first recall the two well-known hardness assumptions — q-SDH [7, 8] and DLIN [11]. We then introduce our Power Diffie-Hellman Exponent (PDHE) assumption as a stronger variant of the n-DHE assumption from [15].

Definition 1 (q-SDH Assumption). *For all probabilistic polynomial-time algorithms $\mathcal{A}$, the following success probability of $\mathcal{A}$ is assumed to be negligible:*

$$\Pr\left[\mathcal{A}(g_1, g_2, g_2^{\gamma}, ..., g_2^{(\gamma^q)}) = (g_1^{\frac{1}{\gamma+x}}, x) : g_1 \in \mathbb{G}, g_2 \in \mathbb{G}', (\gamma, x) \in \mathbb{Z}_p^2\right].$$

Definition 2 (Decision Linear (DLIN) Assumption). *For all probabilistic polynomial-time algorithms $\mathcal{A}$, the following advantage probability of $\mathcal{A}$ is assumed to be negligible:*

$$\left|\begin{array}{l}\Pr\left[\mathcal{A}(u, v, h, u^a, v^b, h^{a+b}) = 1 : (u, v, h) \in_R \mathbb{G}^3, (a, b) \in_R \mathbb{Z}_p^2\right] - \\ \qquad\qquad \Pr\left[\mathcal{A}(u, v, h, u^a, v^b, \eta) = 1 : (u, v, h, \eta) \in_R \mathbb{G}^4, (a, b) \in_R \mathbb{Z}_p^2\right]\end{array}\right|.$$

Note that DLIN assumption serves as a basis for the well-known Linear Encryption scheme [11]. The following n-DHE assumption was introduced in [15].

Definition 3 (n-DHE Assumption [15]). *For all probabilistic polynomial-time algorithms $\mathcal{A}$, the following success probability of $\mathcal{A}$ is assumed to be negligible:*

$$\Pr\left[\mathcal{A}(g, g_1, g_2, ..., g_n, g_{n+2}, ..., g_{2n}) = g_{n+1} : g \in \mathbb{G}', g_i = g^{\alpha^i}, \alpha \in_R \mathbb{Z}_p, \quad i = 1, ..., n, n+2, ..., 2n, n \in \mathbb{N}\right].$$

We will rely on a stronger assumption, which we call PDHE and which can be seen as a variant of n-DHE. Note that in n-DHE assumption, the adversary must compute $g^{\alpha^{n+1}}$ and is only given a set of group elements. In contrast in PDHE assumption the adversary receives roughly twice as many group elements (from both groups) and an additional set of integers, each denoted by η_j.

Definition 4 (PDHE Assumption). *For all probabilistic polynomial-time algorithms $\mathcal{A}$, the following success probability of $\mathcal{A}$ is assumed to be negligible:*

$$\Pr\left[\begin{array}{l}\mathcal{A}\Big(g, \hat{g}, (\{g^{\alpha^{\psi+i}}, \hat{g}^{\alpha^{\psi+i}}, g^{\beta^{\psi+i}}, \hat{g}^{\beta^{\psi+i}}\}_i, \{g^{\alpha^{2\psi+j}} g^{\beta^{2\psi+j}}, \hat{g}^{\alpha^{2\psi+j}} \hat{g}^{\beta^{2\psi+j}}\}_j), \hat{g}^{\beta^{2\psi+n+1}}, \eta_j\Big) = \hat{g}^{\alpha^{2\psi+n+1}} \\ \quad : \ \eta_j = \alpha^{2\psi+j} + \beta^{2\psi+j} \in \mathbb{Z}, (\alpha, \beta) \in \mathbb{Z}_p^{*2}, \psi \in \mathbb{Z}_q, \hat{g} \in \mathbb{G}, g \in \mathbb{G}', \\ \quad i = 1, ..., n, j = 1, ..., n, n+2, ..., 2n, n \in \mathbb{N}\end{array}\right],$$

where α and β generate a subgroup of $\mathbb{Z}_p^*$ of prime order q, and p is a large prime such that $p = 2q + 1$.

Note that values α, β, and ψ remain unknown to the adversary. The main difference to n-DHE is that $\mathcal{A}$ learns integers $\eta_j = \alpha^{2\psi+j} + \beta^{2\psi+j}$ and must output $\hat{g}^{\alpha^{2\psi+n+1}}$. In the generic group model [29] security of PDHE could be argued as follows: A separate analysis can be performed to prove that the probability of $\mathcal{A}$ breaking the PDHE assumption using only set of elements $(g, \hat{g}, g^{\alpha^{\psi+i}}, \hat{g}^{\alpha^{\psi+i}}, g^{\beta^{\psi+i}}, \hat{g}^{\beta^{\psi+i}}, g^{\alpha^{2\psi+j}} g^{\beta^{2\psi+j}}, \hat{g}^{\alpha^{2\psi+j}} \hat{g}^{\beta^{2\psi+j}}, \hat{g}^{\beta^{2\psi+n+1}})$ from $\mathbb{G}$ and $\mathbb{G}'$ remains negligible. This proof is similar to that of the n-DHE assumption. One can then argue that integers η_j perfectly hide the additional values $\alpha^{2\psi+j}$ and $\beta^{2\psi+j}$ used in the PDHE assumption (note that ψ is unknown to the adversary).

5 Security Model and Definitions for Revocable Group Signatures

The security model of a revocable group signature scheme (RGS) defined in this section resembles the standard security model for static group signatures from [4], where we additionally consider the revocation algorithm and augment signing and verification operations of the group signature with revocation-relevant information. Therefore, this model has also partial connection to the security model from [12], where revocation information was handled within the verification procedure only.

Definition 5. *A Revocable Group Signature (RGS) scheme consists of the following algorithms:*

- **KeyGen**(λ, n): *This randomized algorithm takes as input a security parameter* $\lambda \in \mathbb{N}$*, and an integer* $n \in \mathbb{N}$ *(total number of group members). It outputs a* group public key gpk*, a* group manager's secret key gsk*, a* public membership information pmi*, an n-element vector of* membership tokens $\mathsf{mt} = \{\mathsf{mt}_1, ..., \mathsf{mt}_n\}$*, an n-vector of* secret signing keys $\mathsf{sk} = \{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}$*, and a set* S *of indices of unrevoked group members (initially set to contain all indices* $i \in [1, n]$*).*
- **Sign**$(\mathsf{gpk}, \mathsf{mt}_i, \mathsf{sk}_i, M)$: *On input* gpk*,* mt_i*, a secret signing key* sk_i *of user i, and a message* $M \in \{0,1\}^*$*, this randomized algorithm outputs a* group signature σ.
- **Verify**$(\mathsf{gpk}, \mathsf{pmi}, \sigma, M)$: *On input* gpk*,* pmi*, a candidate group signature* σ*, and a message M, this deterministic algorithms outputs either "**true**" or "**false**". (The output of "**true**" indicates that* σ *is a valid signature on M, meaning also that its signer is not revoked.)*
- **Open**$(\mathsf{gpk}, \mathsf{gsk}, \sigma, M)$: *On input* gpk*,* gsk*, a candidate signature* σ*, and a message M, this algorithm outputs index i (meaning that i belongs to the signer of* σ*) or* $\perp$ *(meaning that* σ *is untraceable for* $i \notin [1, n]$*).*
- **Revoke**$(\mathsf{gsk}, \mathsf{S}, \mathsf{pmi}, \mathsf{mt}, i)$**:** *This deterministic algorithm takes as input* gsk*, the set* S *containing indices of unrevoked group member, the up-to-date* pmi*, the up-to-date n-element vector* mt*, and an index i (of the signer to be revoked). The algorithm updates* $\mathsf{S} = \mathsf{S} \backslash \{i\}$*,* pmi*, and* mt *(from which only* pmi *and* mt *will be published, see below for the explanation).*

An RGS scheme is correct *if: (1) for all* $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{sk}, \mathsf{pmi}, \mathsf{mt}, \mathsf{S}) = \mathbf{KeyGen}(\lambda, n)$*, all* $i \in \mathsf{S}$*, and any message* $M \in \{0,1\}^*$*:*

$$\mathbf{Verify}(\mathsf{gpk}, \mathsf{pmi}, \mathbf{Sign}(\mathsf{gpk}, \mathsf{mt}_i, \mathsf{sk}_i, M), M) = \text{“}\mathbf{true}\text{”}$$

and (2) for all $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{sk}, \mathsf{pmi}, \mathsf{S}) = \mathbf{KeyGen}(\lambda, n)$*, all* $i \in S$*, and any message* $M \in \{0,1\}^*$*:*

$$\mathbf{Open}(\mathsf{gpk}, \mathsf{gsk}, \mathbf{Sign}(\mathsf{gpk}, \mathsf{mt}_i, \mathsf{sk}_i, M), M) = i.$$

In our description of the **Revoke** algorithm we implicitly assume that revocation is performed by the group manager (not necessarily the same party that also issues secret signing keys), who in order to revoke some group member $i \in [1, n]$ removes the corresponding index i from S, and updates pmi and mt according to the new set S. Note that the up-to-date set S is used by the group manager only to keep track of unrevoked members and to update pmi. In contrast, the public membership information pmi is distributed by the group manager and is used as input to the verification procedure. In our scheme pmi will correspond to the updated value of the accumulator, while individual membership tokens mt_i will correspond to the updated witnesses of unrevoked signers.

An RGS scheme should satisfy three main security requirements, discussed in the following. We start with the notion of full-traceability, which we define similar to [4], except that in order to account for the introduced dynamic behavior through revocation support, several modifications must be applied. This makes our definition somewhat related to the traceability definition from [5] for dynamic groups. In particular, the adversary must come up with a group signature which verifies successfully but for which the opening algorithm fails to output an index in $[1, n]$. The adversary is allowed to corrupt all members of the group.

Definition 6 (Full-traceability). *An RGS scheme is full-traceable if no probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ can win the following game with non-negligible advantage by interacting with a challenger C.*

1. **Setup:** *C runs the algorithm* **KeyGen**(λ, n) *to generate a group public key* gpk*, a group master secret* gsk*, a public membership information* pmi*, an n-element vector of membership tokens* mt*, an n-element vector of secret signing keys* sk*, and a set* $\mathsf{S} = [1, n]$*. C also defines an initially empty set* U *of corrupted members. Then C invokes $\mathcal{A}$ on input* (gpk, gsk, pmi, mt, S)*, while keeping* sk *private.*
2. **Oracles:** *$\mathcal{A}$ can make a polynomial number of queries to the following oracles (which are answered by C):*
   - **Signing oracle:** *On input a message M and $i \in \mathsf{S}$, this oracle outputs $\sigma =$* **Sign**(gpk, mt_i, sk, M).
   - **UCorruption oracle:** *On input $i \in \mathsf{S}$, this oracle outputs sk_i and updates $\mathsf{U} = \mathsf{U} \cup \{i\}$.*
   - **Revocation oracle:** *On input $i \in [1, n]$, this oracle responds with the output of* **Revoke**(gsk, S, pmi, mt, i).
   - **Opening oracle:** *On input a signature σ and a message M, this oracle responds with the output of* **Open**(gpk, gsk, σ, M).
3. **Output:** *Eventually, $\mathcal{A}$ stops and outputs a signature σ^* and a message M^*.*

$\mathcal{A}$ wins if all of the following holds:

- **Verify**(gpk, pmi, σ^*, M^*) = **true**.
- **Open**(gpk, gsk, σ^*, M^*) = $\perp$.

The advantage of $\mathcal{A}$ in breaking full-traceability is defined as:

$$\mathbf{Adv}_{\mathcal{A}}^{trace}(\lambda) = \Pr[\mathcal{A} \text{ wins in the full-traceability game}],$$

where the probability is taken over the coin tosses of $\mathcal{A}$ and C.

Our next security requirement for RGS schemes is CPA-anonymity. Unlike [4, 5], by dealing with revocability we have to address anonymity of revoked signers. Our definition of CPA-anonymity comes close to the anonymity definition, that was used in the context of the BBS scheme in [11], where the adversary is not given access to the opening oracle.

Definition 7 (CPA-anonymity). *An RGS scheme is CPA-anonymous if no PPT adversary $\mathcal{A}$ can win the following game with non-negligible advantage by interacting with a challenger C.*

1. **Setup:** *C runs* **KeyGen**(λ, n) to generate a group public key gpk, a group master secret gsk, a public membership information pmi, an n-vector of membership token mt, group member's signing keys sk, and a set S of members' indices. *C* defines a set U of corrupt members' being empty initially. Then, *C* gives gpk, pmi, mt, and S to $\mathcal{A}$, while keeping gsk and sk private.
2. **Oracles:** *$\mathcal{A}$ can make a polynomial number of queries to the following oracles (which are answered by C):*

- **Signing oracle:** *On input a message M and $i \in \mathsf{S}$, this oracle outputs* $\sigma = \mathbf{Sign}(\mathsf{gpk}, \mathsf{mt}_i, \mathsf{sk}_i, M)$.
- **UCorruption oracle:** *On input $i \in \mathsf{S}$, this oracle outputs* sk_i *and updates* $\mathsf{U} = \mathsf{U} \cup \{i\}$.
- **Revocation oracle:** *On input $i \in [1, n]$, this oracle responds with the output of* $\mathbf{Revoke}(\mathsf{gsk}, \mathsf{S}, \mathsf{pmi}, i)$.

3. **Challenge:** *$\mathcal{A}$ selects a message M and two indices i_0 and i_1 with $i_0, i_1 \in \mathsf{S}$. C picks random bit $b \leftarrow \{0, 1\}$ and computes $\sigma^* = \mathbf{Sign}(\mathsf{gpk}, \mathsf{mt}_i, \mathsf{sk}_{i_b}, M)$. Then, C sends σ^* to $\mathcal{A}$.*
4. **Output:** *$\mathcal{A}$ continues querying the oracles as above until it eventually stops and outputs a bit b' as its answer to the challenge.*

$\mathcal{A}$ wins the game if $b' = b$ and neither i_0 nor i_1 are revoked. The advantage of $\mathcal{A}$ in breaking anonymity is defined as:

$$\mathbf{Adv}_{\mathcal{A}}^{anon}(\lambda) = \Pr[\mathcal{A} \text{ wins in the CPA-anonymity game}],$$

where the probability is taken over the coin tosses of $\mathcal{A}$ and C.

The final security requirement is non-frameability [5], which accounts for framing attacks executed by a possibly corrupted group manager. Note that traceability only guaranties that any group signature remains traceable. However, it does not take into account potential attacks mounted by the group manager. As motivated in [5], such attacks cannot be captured in a meaningful way in the traceability definition for the dynamic setting (which we have here due to revocability) if the adversary learns the group manager's secret key. That is why non-frameability in the presence of corrupted group managers has to be defined separately.

Definition 8 (Non-frameability). *An RGS is non-frameable if no PPT adversary $\mathcal{A}$ can win the following game with non-negligible advantage by interacting with a challenger C.*

1. **Setup:** *C runs* $\mathbf{KeyGen}(\lambda, n)$ *to obtain a group public key* gpk*, a group master secret* gsk*, a public membership information* pmi*, an n-vector of membership token* mt*, group members' signing keys* sk*, and a set* S *of members' indices. C also defines an empty set* U *as the set of corrupted members' indices. C then gives* gpk*,* gsk*,* pmi*,* mt*, and* S *to $\mathcal{A}$ while keeping* sk *private.*
2. **Oracles:** *$\mathcal{A}$ can make a polynomial number of queries to the following oracles (which are answered by C):*
   - **Signing oracle:** *On input a message M and $i \in \mathsf{S}$, this oracle outputs* $\sigma = \mathbf{Sign}(\mathsf{gpk}, \mathsf{mt}_i, \mathsf{sk}_i, M)$.
   - **UCorruption oracle:** *On input $i \in \mathsf{S}$, this oracle responds with* sk_i *and updates* $\mathsf{U} = \mathsf{U} \cup \{i\}$.
   - **Revocation oracle:** *On input $i \in [1, n]$, this oracle responds with the output of* $\mathbf{Revoke}(\mathsf{gsk}, \mathsf{S}, \mathsf{pmi}, i)$.
   - **Opening oracle:** *On input a signature σ and a message M, this oracle responds with the output of* $\mathbf{Open}(\mathsf{gpk}, \mathsf{gsk}, \sigma, M)$.
3. **Output:** *Eventually, $\mathcal{A}$ stops and outputs a signature σ^* and a message M^*.*

$\mathcal{A}$ wins if all of the following holds

- **Verify**(gpk, pmi, σ^*, M^*) = **true**.
- **Open**(gpk, gsk, σ^*, M^*) = i^*, where $i^* \in [1, n]$, and **Sign**(gpk, mt_i, sk_{i^*}, M^*) has never been queried by $\mathcal{A}$.

The advantage of $\mathcal{A}$ in breaking non-frameability is defined as:

$$\mathbf{Adv}_{\mathcal{A}}^{non-frame}(\lambda) = \Pr[\mathcal{A} \text{ wins the non-frameability game}],$$

where the probability is taken over the coin tosses of $\mathcal{A}$ and C.

6 Our RGS Scheme with Constant Costs for Signers and Verifiers

In this section we provide specification of our RGS scheme. Prior to detailing its algorithms we give a high-level intuition for its construction.

6.1 High-Level Intuition

Our RGS scheme is mainly based on two previous techniques: The (non-revokable) group signature scheme by Boneh, Boyen, and Shacham (BBS) [11] and the dynamic accumulator by Camenisch, Kohlweiss, and Soriente (CKS) [15] (with slight modifications). The use of the BBS scheme in our RGS constructions helps to achieve constant size for group public keys and group signatures, while the efficient revocation is achieved due to the deployed CKS accumulator. The main technical problem in combining BBS scheme with CKS accumulator is as follows: The use of the CKS accumulator results in the linear length of the group public key and in the linear increase of computation costs for signers to update their witnesses with each revoked group member. Our main modification is to change the computation of witnesses in the CKS accumulator by shifting the significant amount of computation costs from the signers over to the group manager.

Combining Modified CKS Accumulator with BBS Group Signature Scheme. As our construction builds on the BBS scheme, we require that key generation is performed by a trusted key issuer, akin to [11]. The key issuer is responsible for the generation of: all secret signing keys, the group manager's secret key (which includes secrets to open signatures and to revoke members), the group public key, and the initial public membership information. In particular, the issuer picks a secret exponent x_i for each member i and computes a secret value $\alpha^{\psi+i}$ and a corresponding group element $\hat{g}^{x_i}\hat{g}^{\alpha^{\psi+i}}\tilde{g}$ in $\mathbb{G}$. It also computes the secret membership certificate $A_i = (\hat{g}^{x_i}\hat{g}^{\alpha^{\psi+i}}\tilde{g})^{\frac{1}{\mu_i+\omega}}$, which becomes part of the secret signing key sk_i. A group member i receives further a personal witness, containing $L_i = \sum_{j\in S \wedge j\neq i} \eta_{n+1-j+i}$, $g^{\alpha^{\psi+i}}$, and $g^{\beta^{\psi+i}}$, where L_i is the initial membership token mt_i, which will be publicly updated by the group manager on each revocation event, as long as i remains unrevoked. The public membership information pmi will contain up-to-date $L = \hat{g}^{\sum_{j\in S} \alpha^{\psi+n+1-j}}$ and $L' = \hat{g}^{\sum_{j\in S} \beta^{\psi+n+1-j}}$, which play the role of the public accumulator value. An unrevoked group member i can thus produce a signature to prove

that it actually possesses a secret signing key, containing a witness accumulated in pmi. When a member i' is revoked, GM updates set S, values L and L' in pmi, and L_i of each unrevoked signer i. In order to generate a new signature i must first obtain up-to-date L, L', and its personal L_i (all of constant length). Note that major computation costs are in the update of corresponding L_i, which is performed by the group manager.

Reducing the Computation Cost of Witnesses of Accumulator. As soon as some member gets revoked remaining group members must implicitly update their secret signing keys upon the execution of the signing operation. For this purpose members use information, which is prepared for them by the group manager, who performs the revocation procedure. In our scheme for every signer there exists an individual public information (of constant length), which we call membership token, and which is updated by the group manager for all unrevoked signers. This information is represented by elements of bilinear groups that correspond to the group elements given to the adversary in the definition of the PDHE assumption (cf. Section 4.2). Using the CKS-like approach for witness updates, when applied in construction, each unrevoked signer would have to compute $\sum_{j\in \mathsf{S}\wedge j\neq i} \alpha^{2\psi+n+1-j+i} + \beta^{2\psi+n+1-j+i}$ in the exponent, where i is the index of the revoked member. This computation can be done by first requiring from the group manager to update j group elements for $j \in \mathsf{S} \wedge j \neq i$, whose discrete logarithms would correspond to each of the $\sum_{j\in \mathsf{S}\wedge j\neq i} \alpha^{2\psi+n+1-j+i} + \beta^{2\psi+n+1-j+i}$. Unrevoked signers would then compute a product of j different group elements. Instead, we let the group manager, who knows $(\alpha, \beta, \psi, i, j)$ anyway, compute and publish $\eta_j = \alpha^{2\psi+j} + \beta^{2\psi+j}$ for each j. In this case computation costs for witnesses would become constant on the signer's side. Note that each member would have to fetch only his updated witness.

6.2 Specification of RGS Algorithms

– **KeyGen(λ, n)** The key generation algorithm is executed by a trusted issuer (akin to [11, 12]), according to the following steps:

1. Select bilinear groups $\mathbb{G}$, $\mathbb{G}'$, $\mathbb{G}_T$ of prime order $p > 2^\lambda$ such that $q = (p-1)/2$ is a prime, and the bilinear map e. Pick a cryptographic hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$.
2. Select $(\tilde{g}, \hat{g}, h, u, v) \in \mathbb{G}^5$, $g \in \mathbb{G}'$, $(\xi_1, \xi_2) \in_R \mathbb{Z}_p^{*2}$, $\omega \in \mathbb{Z}_p^*$, and $\psi \in \mathbb{Z}_q$, where $\hat{g} = u^{\xi_1} = v^{\xi_2}$. Note that $(\hat{g}, u, v)$ represents a public key of the Linear Encryption scheme. Selects further two generators $(\alpha, \beta) \in \mathbb{Z}_p^{*2}$ of prime order q from $\mathbb{Z}_p$, and computes $\alpha^{\psi+i}$ and $\beta^{\psi+i}$, where $i = 1, ..., 2n$.
3. Compute $z = e(\hat{g}^{\alpha^{2\psi+n+1}}, g)$, $z' = e(\hat{g}^{\beta^{2\psi+n+1}}, g)$, $g_{\alpha,i} = g^{\alpha^{\psi+i}}$, $g_{\beta,i} = g^{\beta^{\psi+i}}$, $\eta_j = \alpha^{2\psi+j} + \beta^{2\psi+j}$, where $i = 1, ..., n$, and $j = 1, ..., n, n+2, ..., 2n$.
4. Define the group master secret key $\mathsf{gsk} = (\xi_1, \xi_2, \alpha, \beta, \psi)$ and the group public key $\mathsf{gpk} = (H, p, \mathbb{G}, \mathbb{G}', \mathbb{G}_T, e, \hat{g}, \tilde{g}, g, \Omega, z, z', h, u, v)$, where $\Omega = g^\omega$.
5. Define $\mathsf{S} = \{1, ..., n\}$ as the index set of group members. Compute $L = \hat{g}^{\sum_{j\in \mathsf{S}} \alpha^{\psi+n+1-j}}$, $L' = \hat{g}^{\sum_{j\in \mathsf{S}} \beta^{\psi+n+1-j}}$, and $L_i = \sum_{j\in \mathsf{S}\wedge j\neq i} (\alpha^{2\psi+n+1-j+i} + \beta^{2\psi+n+1-j+i})$ for all $i \in \mathsf{S}$. Define $\mathsf{pmi} = \{L, L'\}$ to be the public membership information, which will be updated by the group manager upon revocation of members, and set $\mathsf{mt} = \{L_1, ..., L_n\}$ to be a vector of membership tokens; each unrevoked member will fetch its own membership token from this vector.

6. Compute secret signing keys $\mathsf{sk}_i = (A_i, g_{\alpha,i}, g_{\beta,i}, \mu_i, x_i)$ for each member $i \in \mathsf{S}$, where $A_i = (\hat{g}^{x_i}\hat{g}^{\alpha^{\psi+i}}\tilde{g})^{\frac{1}{\mu_i+\omega}}$.
7. Publish gpk, pmi, and mt. Output privately gsk, $\{A_i\}_{i\in\mathsf{S}}$, and S to the group manager (GM). Output privately sk_i to the corresponding member i. (Note that ω remains secret and should be ideally erased by the issuer.)

– **Sign**(gpk, mt_i, sk_i, M) Let $\mathsf{gpk} = (H, p, \mathbb{G}, \mathbb{G}', \mathbb{G}_T, e, \hat{g}, \tilde{g}, g, \Omega, z, z', h, u, v)$, $\mathsf{mt}_i = L_i$, and $\mathsf{sk}_i = (A_i, g_{\alpha,i}, g_{\beta,i}, \mu_i, x_i)$. In order to sign some message $M \in \{0,1\}^*$ the signer i proceeds as follows:
1. Select $\pi, \theta, \rho, \delta, r_\pi, r_\theta, r_{\mu_i}, r_\rho, r_{x_i}, r_{\pi\mu_i}, r_{\theta\mu_i}$, and r_δ at random from $\mathbb{Z}_p^*$.
2. Compute $T_1 = A_i\hat{g}^{\pi+\theta}$, $T_2 = u^\pi$, $T_3 = v^\theta$, $T_4 = g^\rho$, $T_5 = \hat{g}^\rho$, $T_6 = g_{\alpha,i}^\rho$, $T_7 = (zz')^\rho$, $T_8 = e(h,T_4)^\delta$, $T_9 = \hat{g}^{L_i}h^\delta$, $T_{10} = g_{\beta,i}^\rho$, $R_1 = e(T_1,T_4)^{r_{\mu_i}} \cdot e(T_1,\Omega)^{r_\rho} / e(\hat{g},T_4)^{r_{x_i}} \cdot e(\tilde{g},g)^{r_\rho} \cdot e(\hat{g},T_4)^{r_{\pi\mu_i}+r_{\theta\mu_i}} \cdot e(T_5,\Omega)^{r_\pi+r_\theta}$, $R_2 = u^{r_\pi}$, $R_3 = v^{r_\theta}$, $R_4 = g^{r_\rho}$, $R_5 = \hat{g}^{r_\rho}$, $R_6 = T_2^{r_{\mu_i}} \cdot u^{-r_{\pi\mu_i}}$, $R_7 = (zz')^{r_\rho}$, $R_8 = e(h,T_4)^{r_\delta}$, $R_9 = e(T_9,g)^{r_\rho}$, and $R_{10} = T_3^{r_{\mu_i}} \cdot v^{-r_{\theta\mu_i}}$, where (T_1, T_2, T_3) is a Linear Encryption ciphertext that encrypts A_i, and T_4 through T_{10} and R_1 through R_{10} are required to prove the knowledge of $\pi, \theta, \mu_i, \rho, x_i, \pi\mu_i, \theta\mu_i$ and of the accumulator $\{L, L'\}$.
3. Compute challenge $c = H(\mathsf{gpk}, M, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10})$.
4. Compute $s_\pi = r_\pi + c\pi$, $s_\theta = r_\theta + c\theta$, $s_\rho = r_\rho + c\rho$, $s_{x_i} = r_{x_i} + cx_i$, $s_{\pi\mu_i} = r_{\pi\mu_i} + c\pi\mu_i$, $s_{\theta\mu_i} = r_{\theta\mu_i} + c\theta\mu_i$, and $s_\delta = r_\delta + c\delta$.
5. Output $\sigma = (c, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, s_\pi, s_\theta, s_\rho, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_\delta)$ on M.

– **Verify**(gpk, pmi, σ, M) Let $\mathsf{gpk} = (H, p, \mathbb{G}, \mathbb{G}', \mathbb{G}_T, e, \hat{g}, \tilde{g}, g, \Omega, z, z', h, u, v)$, $\mathsf{pmi} = (L, L')$, and $\sigma = (c, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, s_\pi, s_\theta, s_\rho, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_\delta)$. The validity of a candidate group signature σ on some message M can be checked according to the following procedure:
1. Compute $R_1' = e(T_1,T_4)^{s_{\mu_i}} \cdot e(T_1,\Omega)^{s_\rho} / e(\hat{g},T_4)^{s_{x_i}} \cdot e(\hat{g},T_6)^c \cdot e(\tilde{g},g)^{s_\rho} \cdot e(\hat{g},T_4)^{s_{\mu_i\pi}+s_{\mu_i\theta}} \cdot e(T_5,\Omega)^{s_\pi+s_\theta}$, $R_2' = u^{s_\pi} \cdot (T_2)^{-c}$, $R_3' = v^{s_\theta} \cdot (T_3)^{-c}$, $R_4' = g^{s_\rho} \cdot (T_4)^{-c}$, $R_5' = \hat{g}^{s_\rho} \cdot (T_5)^{-c}$, $R_6' = T_2^{s_{\mu_i}} \cdot u^{-s_{\pi\mu_i}}$, $R_7' = (zz')^{s_\rho} \cdot (T_7)^{-c}$, $R_8' = e(h,T_4)^{s_\delta} \cdot T_8^{-c}$, and $R_9' = (T_7^c \cdot e(T_9,g)^{s_\rho}) / (T_8 \cdot e(L,T_6) \cdot e(L',T_{10}))^c$, $R_{10}' = T_3^{s_{\mu_i}} \cdot v^{-s_{\theta\mu_i}}$
2. Compute $c' = H(\mathsf{gpk}, M, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1', R_2', R_3', R_4', R_5', R_6', R_7', R_8', R_9', R_{10}')$.
3. If $c' = c$ then output **true**; else output **false**.

If the output is **true**, it means that σ on M is signed by valid signing key sk_i for $i \in \mathsf{S}$. Otherwise, sk_i is invalid or revoked (i.e., $i \notin \mathsf{S}$).

– **Open**(gpk, gsk, σ, M) Let $\mathsf{gpk} = (H, p, \mathbb{G}, \mathbb{G}', \mathbb{G}_T, e, \hat{g}, \tilde{g}, g, \Omega, z, z', h, u, v)$, $\mathsf{gsk} = (\xi_1, \xi_2, \alpha, \beta, \psi)$, and $\sigma = (c, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, s_\pi, s_\theta, s_\rho, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_\delta)$. In order to open some candidate group signature σ the group manager proceeds as follows:
1. If **Verify**(gpk, pmi, σ, M)≠**true** then return $\perp$.
2. Otherwise, compute $A_i = T_1/(T_2^{\xi_1}T_3^{\xi_2})$ with ξ_1, ξ_2 extracted from gsk and T_2, T_3 extracted from σ. Returns i associated to A_i.

– **Revoke**(gsk, S, pmi, mt, i') In order to revoke some member i' the group manager proceeds as follows:

1. Update $\mathsf{S} = \mathsf{S} \backslash \{i'\}$, $L = L/(g^{\alpha^{\psi+n+1-i'}})$, and $L' = L'/(g^{\psi+\beta^{n+1-i'}})$.
2. Compute $L_i = L_i - \alpha^{2\psi+n+1-i'+i} - \beta^{2\psi+n+1-i'+i}$ in mt for each $i \in \mathsf{S}$.
3. Output updated S, pmi, and mt.

7 Security Analysis

Security of our RGS scheme with respect to definitions from Section 5 is established through the following theorems.

Theorem 1 (Full-traceability). *The proposed RGS scheme is fully-traceable in the random oracle model, based on the q-SDH assumption in bilinear groups $\mathbb{G}$ and $\mathbb{G}'$.*

Proof. The proof is given in Appendix A and follows some ideas from [11, 15].

Theorem 2 (CPA-anonymity). *The proposed RGS scheme is CPA-anonymous in the random oracle model, based on the DLIN assumption in $\mathbb{G}$.*

Proof. The proof is given in Appendix B.

Theorem 3 (Non-frameability). *The proposed RGS scheme is non-frameable in the random oracle model, based on the PDHE assumption and the hardness of computing discrete logarithms in $\mathbb{G}$.*

Proof. The proof is given in Appendix C.

8 Conclusion

In this work we made another step towards better efficiency in revocable group signatures. Our proposed RGS scheme achieves constant costs for signers and verifiers at the price of a higher amount of work for the group manager and a rather strong Power Diffie-Hellman Exponent (PDHE) assumption. In addition to constant computation costs, our scheme keeps group public keys, group signature, and the amount of public update information, to be fetched by either an unrevoked signer or a verifier, also constant. Our scheme, which is based on the combination of a modified CKS dynamic accumulator from [15] with the BBS group signature scheme from [11] preserves the original security properties of the BBS scheme, while also offering support for the revocation of the signing rights. An open problem would be to find a solution that achieves similar complexity under somewhat more standard (pairing-based) assumptions.

Acknowledgements. Ruei-Hau Hsu acknowledges the NSC-DAAD Sandwich Program of 2011 (Spring Season) sponsored by National Science Council (NSC) in Taiwan and German Academic Exchange Service (DAAD) for the received financial support during his research stay at TU Darmstadt. Mark Manulis was supported by grant MA 4957 of the German Research Foundation (DFG).

References

1. Ateniese, G., Tsudik, G.: Some Open Issues and New Directions in Group Signatures. In: Franklin, M.K. (ed.) FC 1999. LNCS, vol. 1648, pp. 196–211. Springer, Heidelberg (1999)
2. Ateniese, G., Camenisch, J.L., Joye, M., Tsudik, G.: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
3. Ateniese, G., Song, D., Tsudik, G.: Quasi-Efficient Revocation of Group Signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction based on General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
5. Bellare, M., Shi, H., Zhang, C.: Foundations of Group Signatures: The Case of Dynamic Groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
6. Biham, E., Shamir, A.: Differential Cryptanalysis of the Full 16-Round DES. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 487–496. Springer, Heidelberg (1993)
7. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
8. Boneh, D., Boyen, X.: Short Signatures without Random Oracles and the SDH Assumption in Bilinear Groups. Journal of Cryptology 21(2), 149–177 (2008)
9. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
10. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
11. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
12. Boneh, D., Shacham, H.: Group Signatures with Verifier-Local Revocation. In: Proceedings of 11th ACM Conference on Computer and Communication Security: ACM-CCS 2004, pp. 168–177 (2004)
13. Bresson, E., Stern, J.: Efficient Revocation in Group Signatures. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 190–206. Springer, Heidelberg (2001)
14. Camenisch, J.L., Groth, J.: Group Signatures: Better Efficiency and New Theoretical Aspects. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 120–133. Springer, Heidelberg (2005)
15. Camenisch, J., Kohlweiss, M., Soriente, C.: An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
16. Camenisch, J.L., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
17. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
18. Jin, H., Wong, D.S., Xu, Y.: Efficient Group Signature with Forward Secure Revocation. In: Ślęzak, D., Kim, T.-h., Fang, W.-C., Arnett, K.P. (eds.) SecTech 2009. CCIS, vol. 58, pp. 124–131. Springer, Heidelberg (2009); Proceedings of ANTS IV. LNCS 1838, pp.385–394. Springer (2000)

19. Libert, B., Vergnaud, D.: Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 498–517. Springer, Heidelberg (2009)
20. Nakanishi, T., Funabiki, N.: Efficient Revocable Group Signature Schemes Using Primes. Journal of Information Processing 16, 110–121 (2008)
21. Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 463–480. Springer, Heidelberg (2009)
22. Nakanishi, T., Funabiki, N.: "Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E90-A(1), 65–74 (2007)
23. Nakanishi, T., Kubooka, F., Hamada, N., Funabiki, N.: Group Signature Schemes with Membership Revocation for Large Groups. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 443–454. Springer, Heidelberg (2005)
24. Nakanishi, T., Sugiyama, Y.: A Group Signature Scheme with Efficient Membership Revocation for Reasonable Groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 336–347. Springer, Heidelberg (2004)
25. Nakanishi, T., Funabiki, N.: Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 533–548. Springer, Heidelberg (2005)
26. Nakanishi, T., Funabiki, N.: A Short Verifier-Local Revocation Group Signature Scheme with Backward Unlinkability. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 17–32. Springer, Heidelberg (2006)
27. Pointcheval, D., Stern, J.: Security Proofs for Signature Schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
28. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology 13(3), 361–396 (2000)
29. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
30. Tsudik, G., Xu, S.: Accumulating Composites and Improved Group Signing. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 269–286. Springer, Heidelberg (2003)
31. Zhou, S., Lin, D.: A Shorter Group Signature with Verifier-Local Revocation and Backward Unlinkability, Cryptology ePrint Archive: Report 2006/100 (2006), `http://eprint.iacr.org/2006/100`
32. Zhou, S., Lin, D.: Shorter Verifier-Local Revocation Group Signatures from Bilinear Maps. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 126–143. Springer, Heidelberg (2006)

A Proof of Theorem 1 (Full-Traceability)

Let $\mathcal{A}$ be an adversary that breaks the full-traceability of the proposed protocol by returning an untraceable signature σ_{i^*} with probability at least ϵ. We construct a PPT algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ and breaks the q-SDH assumption with probability at least ϵ'. The interaction of $\mathcal{B}$ with $\mathcal{A}$ proceeds as follows.

– **Setup:** $\mathcal{B}$ is given an n-SDH instance $(\tilde{g}, g, g^{\omega}, g^{\omega^2}, ..., g^{\omega^n})$ by a challenger C_s of n-SDH assumption, where $\tilde{g} \in \mathbb{G}$, $g \in \mathbb{G}'$, and n is the number of group members. $\mathcal{B}$ defines set $\mathsf{S} = [1, n]$ and an initially empty set U, and randomly selects an index

$i^* \in \mathsf{S}$ of a member to be attacked by $\mathcal{A}$. $\mathcal{B}$ then randomly selects $(\alpha, \beta, \tau, \xi_1, \xi_2) \in \mathbb{Z}_p^{*5}$, $\psi \in \mathbb{Z}_q$, and computes $\hat{g} = \tilde{g}^{\tau}$, $\{g_{\alpha,i} = g^{\alpha^{\psi+i}}, g_{\beta,i} = g^{\beta^{\psi+i}}\}_{1 \le i \le n}$. $\mathcal{B}$ uses g^{ω} as Ω and computes gpk, gsk, pmi, and mt given to $\mathcal{A}$. After that, $\mathcal{B}$ turns an n-SDH instance into values $(\tilde{g}, g, \Omega = g^{\omega})$ and $n-1$ SDH pairs $(\tilde{g}^{\frac{1}{\mu_i+\omega}}, \mu_i)$ by Lemma 3.2 from [7] such that $e(g^{\frac{1}{\mu_i+\omega}}, \Omega g^{\mu_i}) = e(\tilde{g}, g)$. $\mathcal{B}$ then transforms the $n-1$ SDH pairs to $n-1$ members' signing key $\mathsf{sk}_i = (A_i (= (\tilde{g}^{\frac{1}{\mu_i+\omega}})^{\tau(x_i+\alpha^{\psi+i})+1} = (\hat{g}^{x_i}\hat{g}^{\alpha^{\psi+i}}\tilde{g})^{\frac{1}{\mu_i+\omega}}), g_{\alpha,i}, g_{\beta,i}, \mu_i, x_i)$, where $i \neq i^*$.

– **Oracles**: $\mathcal{B}$ simulates RGS by answering the following oracle queries.
  - **Hash oracle:** The hash oracle as a random oracle is simulated by $\mathcal{B}$. $\mathcal{B}$ randomly selects element in $\mathbb{Z}_p$ as the output of hash query and makes sure the responses are identical to the same queries by maintaining a hash list H-**list**.
  - **Signing oracle:** On input a pair (i, M), if $i \in [1, n]$, $\mathcal{B}$ can successfully responds with the corresponding σ by sk_i. If $i \in \mathsf{U}$, reject this request. If $i = i^*$, the simulation fails.
  - **UCorruption oracle:** On input i, if $i \in \mathsf{S} \wedge i \neq i^*$, $\mathcal{B}$ responds with $\mathsf{sk}_i = (A_i, g_{\alpha,i}, g_{\beta,i}, \mu_i, x_i)$ to $\mathcal{A}$ and appends i to U. If $i \in \mathsf{U}$, $\mathcal{B}$ returns sk_i without changing S and U. If $i = i^*$, the simulation fails.
  - **Revocation oracle:** On input$i$, if $i \in \mathsf{S}$, $\mathcal{B}$ updates $\mathsf{S} = \mathsf{S} \backslash \{i\}$, $\mathsf{pmi} = \{L, L'\}$, and $\mathsf{mt} = \{L_i\}_{i \in \mathsf{S}}$. $\mathcal{B}$ outputs S, pmi, and mt to $\mathcal{A}$.
  - **Opening oracle:** On inputσ, $\mathcal{A}$ decrypt (T_1, T_2, T_3) of σ to obtain A_i and returns the corresponding i of A_i to $\mathcal{A}$. Otherwise, return $\perp$.
– **Output:** Finally, $\mathcal{A}$ outputs a signature-message pair (σ^*, M^*).

$\mathcal{A}$ is the adversary to break the proposed scheme if σ^* is correct and belongs to some member $i \notin [1, n]$ with probability at least ϵ. Then $\mathcal{A}$ outputs a forged signature σ_{i^*} of the member i^* with probability at least ϵ/n. By Forking Lemma [27, 28], if $\mathcal{A}$ outputs a valid message-signature tuple $(M, \sigma_0 = (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, T_{10}), c, \sigma_1 = (s_\pi, s_\theta, s_\rho, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_\delta))$. We rewind the framework and $\mathcal{A}$ with the same random tape and the different random oracle H'. Then $\mathcal{A}$ still can output a forgery $(M, \sigma_0, c', \sigma_1')$ with probability at least $\epsilon/4$. Consequently, we obtain two valid tuples (M,σ_0,c,σ_1) and $(M,\sigma_0, c',\sigma_1')$ of the same member i^* with probability at least $\epsilon/4n$. $\mathcal{B}$ can extract the secrets $x_{i^*} = (s'_{x_{i^*}} - s_{x_{i^*}})/(c' - c)$, $\pi = (s_\pi - s'_\pi)/(c - c')$, and $\theta = (s_\theta - s'_\theta)/(c - c')$ from the above two valid signature tuples of the member i^*. After that, $\mathcal{B}$ computes $\tilde{g}^{\frac{1}{\mu_{i^*}+\omega}} = ((\hat{g}^{x_{i^*}}\hat{g}^{\alpha^{\psi}+i}\tilde{g})^{\frac{1}{\mu_{i^*}+\omega}})^{(\tau(x_{i^*}+\alpha^{\psi+i})+1)^{-1}}$ to obtain an SDH pair $(\tilde{g}^{\frac{1}{\mu_{i^*}+\omega}}, \mu_{i^*})$.

B Proof of Theorem 2 (CPA-Anonymity)

Suppose $\mathcal{A}$ is an adversary that breaks the CPA-anonymity of our RGS scheme with the advantage at least ϵ. We construct a PPT algorithm $\mathcal{B}$ that breaks Linear encryption (and by this the DLIN assumption) with the advantage at least ϵ by playing the anonymity game from Definition 7. The interaction between $\mathcal{B}$ and $\mathcal{A}$ proceeds as follows.

– **Setup:** First, $\mathcal{B}$ selects the groups $\mathbb{G}$, $\mathbb{G}'$, and $\mathbb{G}_T$ of prime order p. $\mathcal{B}$ is in possession of a public key $(u, v, \hat{g}) \in \mathbb{G}^3$ for the Linear encryption scheme (which it received

from the Linear encryption challenger C_{le}). Recall that $\hat{g} = u^{\xi_1} = v^{\xi_2}$ for some unknown $(\xi_1, \xi_2) \in \mathbb{Z}_p^2$. $\mathcal{B}$ randomly picks $\tilde{g} \in \mathbb{G}$, $g \in \mathbb{G}'$, $\omega \in \mathbb{Z}_p^*$, $(\alpha, \beta) \in \mathbb{Z}_p^*$ of prime order q, and $\psi \in \mathbb{Z}_q$. $\mathcal{B}$ then initializes the sets $\mathsf{S} = \{1, ..., n\}$ and U (set of corrupted signers). Then, $\mathcal{B}$ generates the remaining parts of gpk, pmi, and mt according to the key generation procedure of the RGS scheme. $\mathcal{B}$ computes secret signing keys $\mathsf{sk}_i = \{A_i, g_{\alpha,j}, g_{\beta,j}, \mu_i, x_i\}$ for all $i \in \mathsf{S}$. $\mathcal{B}$ then provides $\mathcal{A}$ with $\mathsf{gpk} = (p, \mathbb{G}, \mathbb{G}', \mathbb{G}_T, e, \hat{g}, \tilde{g}, g, \Omega, z, z', h, u, v)$, $\mathsf{pmi} = \{L, L'\}$, $\mathsf{mt} = \{L_i\}_{i \in \mathsf{S}}$, S, U, and stores ω, α, β, and ψ for later use.

– **Oracles:** $\mathcal{B}$ answers oracle queries of $\mathcal{A}$ as follows.
  - **Hash oracle:** $\mathcal{B}$ simulates the random oracle H by maintaining a hash list H-**list**, and responds on new queries with random elements from $\mathbb{Z}_p$, while making sure that previous queries when asked again are answered consistently with H-**list**.
  - **Signing oracle:** Before the simulation, $\mathcal{B}$ also maintains a list E-**list** for storing the corresponding identity information of signatures. In the simulation of signing queries, $\mathcal{B}$ responds with σ_i for the oracle query with the corresponding input (i, M) of $\mathcal{A}$ as follows. $\mathcal{B}$ selects π and $\theta \in_R \mathbb{Z}_p^*$, and encrypts A_i as a ciphertext $(T_1 = A_i \hat{g}^{\pi+\theta}, T_2 = u^{\pi}, T_3 = v^{\theta})$. $\mathcal{B}$ then randomly selects ρ, δ, r_{π}, r_{θ}, r_{μ_i}, r_{ρ}, r_{x_i}, $r_{\pi\mu_i}$, $r_{\theta\mu_i}$, r_{δ}, and r_{δ} to generates T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, and R_{10}. $\mathcal{B}$ also updates the output of hash list at $(\mathsf{gpk}, M, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10})$ is equal to $c \in \mathbb{Z}_p$ and outputs $\sigma = (c, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, s_{\pi}, s_{\theta}, s_{\rho}, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_{\delta})$. $\mathcal{B}$ also adds (σ, i) into E-**list**.
  - **UCorruption oracle:** On input $i \in \mathsf{S}$, $\mathcal{B}$ responds with the corresponding sk_i to $\mathcal{A}$ and appends i to U.
  - **Revocation oracle:** On input $i \in \mathsf{S}$, $\mathcal{B}$ removes i from S, re-computes $\mathsf{pmi} = \{L, L'\}$ and $\mathsf{mt} = \{L_i\}_{i \in \mathsf{S}}$, and responds with the updated S, pmi, and mt to $\mathcal{A}$.
  - **Signing oracle:** $\mathcal{B}$ also maintains a list E-**list** for keeping track of output signatures. In the simulation of signing queries, $\mathcal{B}$ responds with σ_i computed on input (i, M) from $\mathcal{A}$ as follows. $\mathcal{B}$ selects π and $\theta \in_R \mathbb{Z}_p^*$, and compute Linear encryption of A_i, i.e. ciphertext $(T_1 = A_i \hat{g}^{\pi+\theta}, T_2 = u^{\pi}, T_3 = v^{\theta})$. $\mathcal{B}$ then randomly selects ρ, δ, r_{π}, r_{θ}, r_{μ_i}, r_{ρ}, r_{x_i}, $r_{\pi\mu_i}$, $r_{\theta\mu_i}$, r_{δ}, and r_{δ} to generates T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, and R_{10}. $\mathcal{B}$ defines the output of $H(\mathsf{gpk}, M, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10})$ to be equal to $c \in \mathbb{Z}_p$ and outputs $\sigma = (c, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, s_{\pi}, s_{\theta}, s_{\rho}, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_{\delta})$. Finally, $\mathcal{B}$ adds (σ, i) into E-**list**.
  - **UCorruption oracle:** If $\mathcal{A}$ corrupts $i \in \mathsf{S}$ then $\mathcal{B}$ hands the corresponding secret signing key sk_i over to $\mathcal{A}$ and includes i into U.
  - **Revocation oracle:** If $\mathcal{A}$ wishes to revoke some member $i \in \mathsf{S}$ then $\mathcal{B}$ removes i from S and updates $\mathsf{pmi} = \{L, L'\}$ and $\mathsf{mt} = \{L_i\}_{i \in \mathsf{S}}$ as specified in the RGS scheme. $\mathcal{B}$ then hand updated pmi and mt over to $\mathcal{A}$.

– **Challenge:** In the challenge phase, $\mathcal{A}$ selects a message M, two indices i_0 and i_1, and sends them to $\mathcal{B}$. If $(i_0, i_1) \in \mathsf{S}^2 \wedge (i_0, i_1) \notin \mathsf{U}^2$, $\mathcal{B}$ returns A_{i_0} and A_{i_1} as its challenge to C_{le}. C_{le} replies with Linear encryption ciphertext (T_1, T_2, T_3) of A_{i_b} according to some random (unknown) bit $b \in \{0, 1\}$. Then $\mathcal{B}$ randomly selects ρ and δ to compute T_4, T_5, T_6, T_7, T_8, T_9, and T_{10}. It further selects random s_{π}, s_{θ}, s_{ρ}, s_{x_i},

$s_{\pi\mu_i}$, $s_{\theta\mu_i}$, s_δ, and c from $\mathbb{Z}_p^*$, and computes $R_1 = e(T_1, T_4)^{s_{\mu_i}} \cdot e(T_1, \Omega)^{s_\rho} / e(\hat{g}, T_4)^{s_{x_i}} \cdot e(T_6, g)^c \cdot e(\tilde{g}, g)^{s_\rho} \cdot e(\hat{g}, T_4)^{s_{\mu_i\pi} + s_{\mu_i\theta}} \cdot e(T_5, \Omega)^{s_\pi + s_\theta}$, $R_2 = u^{s_\pi} \cdot (T_2)^{-c}$, $R_3' = v^{s_\theta} \cdot (T_3)^{-c}$, $R_4 = g^{s_\rho} \cdot (T_4)^{-c}$, $R_5 = \hat{g}^{s_\rho} \cdot (T_5)^{-c}$, $R_6 = T_2^{s_{\mu_i}} \cdot u^{-s_{\pi\mu_i}}$, $R_7 = (zz')^{s_\rho} \cdot (T_7)^{-c}$, $R_8 = e(h, T_4)^{s_\delta} \cdot T_8^{-c}$, $R_9 = (T_7^c \cdot e(T_9, g)^{s_\rho}) / (T_8 \cdot e(L, T_6) \cdot e(L', T_{10}))^c$, and $R_{10} = T_3^{s_{\mu_i}} \cdot v^{-s_{\theta\mu_i}}$. $\mathcal{B}$ then sends $\sigma_{i_b} = (c, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, s_\pi, s_\theta, s_\rho, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_\delta)$ to $\mathcal{A}$.

– $\mathcal{B}$ continues answering oracle queries of $\mathcal{A}$ as specified above, until $\mathcal{A}$ outputs a bit b'.

$\mathcal{B}$ forwards b' as its answer on the challenge of C_{le} in the Linear encryption game. Clearly, if $\mathcal{A}$ wins then $\mathcal{B}$ breaks the IND-CPA security of the Linear encryption scheme.

C Proof of Theorem 3 (Non-frameability)

We consider two types of adversaries to break non-frameability in the proposed scheme. Type I adversary can forge group signatures of the member i for $i \in \mathsf{S}$ and type II adversary for $i \in [1, n] \wedge i \notin S$.

Type I Adversary. Let $\mathcal{A}$ be the type I adversary of non-frameability of our RGS scheme with the probability at least ϵ by forging a group signature of member i for $i \in \mathsf{S}$. We then construct an algorithm $\mathcal{B}$ that can break the classical discrete logarithm (DL) assumption with probability at least ϵ' in polynomial time by playing the game in Definition 8. The algorithm $\mathcal{B}$ proceeds as follows.

– **Setup:** $\mathcal{B}$ is given a DL instance $(\hat{g}, \mathcal{U} = \hat{g}^x)$ by a challenger C_{DL} of DL assumption, where $\hat{g} \in \mathbb{G}$ and $x \in_R \mathbb{Z}_p$. $\mathcal{B}$ then prepares the sets S and U, and an index $i^* \in \mathsf{S}$ of the target member to be attacked by $\mathcal{A}$. After that, $\mathcal{B}$ generates gsk, gpk, pmi, mt, and $\{\mathsf{sk}_i\}_{i \in \mathsf{S}}$ as well as the proposed scheme except sk_{i^*}. Here $\mathcal{B}$ computes $A_{i^*} = (\mathcal{U}\tilde{g})^{\frac{1}{\mu_i + \omega}}$ and sets $x_{i^*} = x$, which is unknown.
– **Oracles:**
  - **Hash oracle:** The simulation of hash queries is the same as in the proof of Theorems 2 and 1.
  - **Signing oracle:** Here $\mathcal{B}$ also maintains a list E**-list** for storing the corresponding identity and signature pairs. On input a pair (i, M), if $i \in \mathsf{S}$ and $i \neq i^*$, $\mathcal{B}$ can successfully generate any signature of the member i. If $i = i^*$, $\mathcal{B}$ generates $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_6, R_7, R_8, R_9$, and T_{10} as in the RGS specification by using random s_π, s_θ, s_ρ, s_{x_i}, $s_{\pi\mu_i}$, $s_{\theta\mu_i}$, and s_δ from $\mathbb{Z}_p^*$. After that, $\mathcal{B}$ responds with $\sigma = (c, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, s_\pi, s_\theta, s_\rho, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_\delta)$.
  - **UCorruption oracle:**On input $i \in \mathsf{S}$, if $i \neq i^*$ then $\mathcal{B}$ returns sk_i. Otherwise, $\mathcal{B}$ aborts.
  - **Revocation and opening oracles:** The simulations of the revocation and opening oracles are the same as that of Theorem 2 and 1.
– **Output:** Finally, $\mathcal{B}$ outputs a signature-message pair (σ^*, M^*).

$\mathcal{A}$ breaks the proposed scheme if **Verify**$(\mathsf{gpk}, \mathsf{pmi}, \sigma^*, M^*) = \mathbf{true}$. In addition, if $\mathcal{A}$ outputs a forged σ_i with probability at least ϵ, then $\mathcal{A}$ outputs a forged σ_{i^*} of the target member i^* with probability at least ϵ/n. $\mathcal{B}$ then can successfully obtain two forged signatures $\sigma_{i^*} = (\sigma_{i_0^*} = (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10}), c, \sigma_{i_1^*} = (s_\pi, s_\theta, s_\rho, s_{x_i}, s_{\pi\mu_i}, s_{\theta\mu_i}, s_\delta)$ and $\sigma'_{i^*} = (\sigma_{i_0^*} = (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10}), c', \sigma'_{i_1^*} = (s'_\pi, s'_\theta, s'_\rho, s'_{x_i}, s'_{\pi\mu_i}, s'_{\theta\mu_i}, s'_\delta))$ by applying Forking Lemma [27] and extract $x_{i^*} = (s'_{x_{i^*}} - s_{x_{i^*}})/(c' - c)$ as the solution of DL problem with probability at least $\epsilon/4n$. Hence, we have that $\epsilon/4n \leq \epsilon'$.

Type II Adversary. Let $\mathcal{A}$ be an adversary that breaks the non-frameability of the proposed scheme with the probability at least ϵ by forging a group signature of member i for $i \in [1, n] \wedge i \notin \mathsf{S}$. We then construct an algorithm $\mathcal{B}$ to break PDHE assumption with probability at least ϵ''. The construction of $\mathcal{B}$ and its interaction with $\mathcal{A}$ proceed as follows.

- **Setup:** $\mathcal{B}$ is given a PDHE instance $(g, \hat{g}, g^{\alpha^{\psi+i}}, \hat{g}^{\alpha^{\psi+i}}, g^{\beta^{\psi+i}}, \hat{g}^{\beta^{\psi+i}}, \eta_j\}_{1\leq i,j\leq n\wedge i\neq j})$ by a challenger C_{PDHE} of PDHE assumption, where $i = 1, ..., n$ and $j = 1, ..., n, n+2, ..., 2n$. $\mathcal{B}$ then prepares sets S and U being the same as that of the simulation for type I adversary and prepares gsk and gpk as well as that of the proposed scheme except that α, β, and ψ are unknown. Then $\mathcal{B}$ simulates $\mathsf{sk}_i = (A_i, g^{\alpha,i}, g^{\beta,i}, \mu_i, x_i)$ for the member i by using PDHE instance, gsk, and gpk.
- **Oracles:** $\mathcal{B}$ answers the following oracles to simulates RGS interacting with $\mathcal{A}$.
  - **Hash oracle:** The simulation of hash queries are the same as that of the simulation for type I adversary.
  - **Signing oracle:**On inputting a pair (i, M), if $i \in [1, n]$, $\mathcal{B}$ generates the corresponding σ by sk_i. Otherwise, reject this request.
  - **UCorruption, revocation, and opening oracles:** The simulations of ucorruption revocation and opening oracles are the same as that for type I adversary.
- **Output:** Finally, $\mathcal{A}$ output a signature-message pair (σ^*, M^*).

$\mathcal{A}$ is a type II adversary of non-frameability to break the proposed scheme if σ^* is correct and belongs to some $i \in [1, n] \wedge i \notin \mathsf{S}$ with probability at least ϵ. Then $\mathcal{B}$ can successfully break PDHE assumption as follows. $\mathcal{B}$ can apply Forking Lemma as well as the proof for type I adversary to extract secrets ρ and δ such that $(zz')^\rho = e(L, T_6) \cdot e(L', T_{11}) \cdot T_8/e(T_9, g)^\rho = (e(\hat{g}^{\alpha^{2\psi+n+1}}, g) \cdot e(\hat{g}^{\beta^{2\psi+n+1}}, g))^\rho$. This means that $\hat{g}^{\alpha^{2\psi+n+1}}\ \hat{g}^{\beta^{2\psi+n+1}} = \frac{\prod_{j\in\mathsf{S}} \hat{g}^{\alpha^{2\psi+n+1-j+i}} \prod_{j\in\mathsf{S}} \hat{g}^{\beta^{2\psi+n+1-j+i}}}{T_9 \cdot (h^\delta)^{-1}}$. $\mathcal{B}$ can directly compute $\prod_{j\in\mathsf{S}} \hat{g}^{\alpha^{2\psi+n+1-j+i}}$ and $\prod_{j\in\mathsf{S}} \hat{g}^{\beta^{2\psi+n+1-j+i}}$ since $i \notin \mathsf{S}$ such that $i \neq j$. Therefore, $\mathcal{B}$ can successfully break PDHE assumption by extracting $\hat{g}^{\alpha^{2\psi+n+1}}$ from the forged signature σ^* by using $\hat{g}^{\beta^{2\psi+n+1}}$.