Efficient and Provably Secure Trapdoor-free Group Signature Schemes from Bilinear Pairings

Lan Nguyen and Rei Safavi-Naini

School of Information Technology and Computer Science
University of Wollongong, Wollongong 2522, Australia
{ldn01,rei}@uow.edu.au

[1]

Abstract. Group signature schemes are cryptographic systems that provide revocable anonymity for signers. We propose a group signature scheme with constant-size public key and signature length that does not require trapdoor. So system parameters can be shared by multiple groups belonging to different organizations. The scheme is provably secure in the formal model recently proposed by Bellare, Shi and Zhang (BSZ04), using random oracle model, Decisional Bilinear Diffie-Hellman and Strong Diffie-Hellman assumptions. We give a more efficient variant scheme and prove its security in a formal model which is a modification of BSZ04 model and has a weaker anonymity requirement. Both schemes are very efficient and the sizes of signatures are approximately one half and one third, respectively, of the sizes of the well-known ACJT00 scheme. We will show that the schemes can be used to construct a traceable signature scheme and identity escrow schemes. They can also be extended to provide membership revocation.
Keywords: Group signatures, traceable signatures, membership revocation, identity escrow, privacy and anonymity, cryptographic protocols.

1 Introduction

Group signature schemes, introduced by Chaum and Van Heyst [15], allow a group member to sign a message on behalf of the group without revealing his identity and without allowing the message to be linkable to other signed messages that are verifiable with the same public key. Participants in a group signature scheme are a set of *group members* and a *group manager*. The role of the group manager is to register new users by issuing membership certificates that contain registration details, and in case of dispute about a signed message, revoking anonymity of the signed message by 'opening' the signature. In some schemes the functions of the group manager can be split between two managers: an *issuer* and an *opener*. This is a desirable property that allows distribution of trust. It is required that no collusion of the issuer and the opener can frame a group member.

[1] An extended abstract of this paper is in Advances in Cryptology - Asiacrypt 2004, Springer-Verlag.

Group signatures are among the most important cryptographic primitives for providing privacy and have been used for applications such as anonymous credentials [2], identity escrow [24], voting and bidding [1], and electronic cash [26]. Group signature schemes are the non-interactive counterpart of identity escrow systems [23].

In early group signature schemes [10, 15, 16] the size of the public key and the signature grew with the size of the group and so the schemes were impractical for large groups. Schemes with fixed size group public key and signature length have been first proposed in [14] and later extended in [13, 1, 2]. In Crypto 2000, Ateniese et al. (ACJT00) [1] proposed an efficient group signature scheme with very short length and low computation cost. This scheme is also the only scheme that has been proved to satisfy the informal list of security requirements of group signature schemes.

Ateniese and de Medeiros (AdM03) proposed an efficient group signature scheme [2] that is 'without trapdoor' in the sense that none of parties in the system including the group manager need to know the trapdoor. That is the system trapdoor is only used during the initialisation and to generate system parameters. The advantage of this property is that the same trapdoor information can be used to initiate different groups. The importance and usefulness of this property in real-world applications, for example when the group signature scheme is used as a building block of an anonymous credential system among a number of organizations that need to communicate and transfer information about users while protecting their privacy, have been outlined in [2]. A drawback of AdM03 scheme is that it has a single group manager who is responsible for registration of users and opening of signatures, and it is not possible to separate the two functionalities. In AdM03 scheme, the group manager stores the certificate (r, s) of each member. The signature of a group member contains elements χ and E_1 satisfying the equation $E_1 = \chi^r$, and so, to revoke a signature, the group manager (or any party with the knowledge of the certificates) can try all certificates to find the one satisfying the equation. This is an computationally expensive process. The security proof (corrected version) is for the informal list of security requirements, and is given in the generic model [3].

Security of a group signature scheme has been traditionally proved by showing that it satisfies a list of informally defined requirements. Bellare et al. [4] gave a formal security model for group signature schemes for static groups and reduced the number of requirements to three, correctness, full anonymity and full traceability, hence simplifying security goals and analysis. This model (referred to as BMW03 model) was later extended [5] to (partially) dynamic groups with four security requirements (Correctness, Anonymity, Traceability and Non-frameability). Kiayias et al. [22] independently proposed a second formal model (KY04 model) for group signature with four requirements, Correctness, Anonymity, Misidentification and Framing, that shares many features of BSZ04 model. Both models use various oracles including an Open oracle that takes a signed message and reveals the identity of the signer. The ACJT00 scheme although satisfies the conventional list of requirements but cannot be proved secure in either of the two formal models mainly because of the inclusion of the Open oracle in these models. Kiayias et al. [22] proposed an extension (KY04 scheme) of ACJT00 scheme that is proved secure in their formal model.

A new direction in constructing group signature schemes is to use bilinear pairings to shorten the lengths of the signature and key. Boneh et al. [8] proposed a short group signature scheme (BBS04) based on the Strong Diffie-Hellman assumption and a new assumption called the Decisional Linear assumption. The scheme is provably secure in a variant of BMW03 model where the Opening oracle is not available and the Non-frameability property is not required, in comparison with the BSZ04 model. They also showed how to construct an extension, which provides Non-frameability (exculpability). Based on the LRSW assumption [25], Camenisch and Lysyanskaya [12] proposed a group signature scheme (CL04) derived from a signature scheme which allows an efficient zero-knowledge proof of the knowledge of a signature on a committed message, and used it to construct an efficient anonymous credential system. Our group signature schemes belong to this direction and are proposed independently from the BBS04 and CL04 schemes.

Our contribution

In this paper, we first propose a new efficient group signature scheme with a number of attractive properties and prove its security in the BSZ04 model under the Decisional Bilinear Diffie-Hellman and Strong Diffie-Hellman assumptions, using random oracle model. We then give an efficient variant of this scheme and prove its security in the reduced version of BSZ04 model. The only difference between the original BSZ04 model and the reduced version is in modelling anonymity property in the reduced version, the adversary does not have access to the Open oracle. This is a plausible model for all cases that the opener is a highly trusted entity and cannot be accessed by the adversary.

The main difference between our two schemes is that in the first scheme the opener uses an encryption scheme that is indistinguishably secure against adaptively chosen ciphertext attack, whereas in the variant scheme the encryption scheme is indistinguishably secure against adaptively chosen plaintext attack. The difference between the anonymity requirement and the weak anonymity requirement is similar to the difference in modelling chosen ciphertext attack and chosen plaintext attack in encryption schemes. That is in the anonymity and weak-anonymity security requirement games, the identity is encrypted and access to the open oracle is similar to access to the decryption oracle. As the open oracle is not used in the informal list of security requirements, using the same arguments as in [4, 5], we can conclude that the weak anonymity, traceability and non-frameability properties are sufficient to capture the conventional list of requirements for group signature schemes. We also show that the ACJT00 scheme provides weak anonymity and under Strong RSA and Decisional Diffie-Hellman assumptions in the random oracle model. We note that the relationship between ACJT00 and KY04 is the same as the one between our two schemes.

In the following we outline attractive features of our schemes in comparison with previous schemes and point out the relationship between them. Both proposed schemes have fixed lengths for group public key and signature, and so can be used for large size groups. The schemes are trapdoor-free. All previous efficient constant-size group signature schemes, except for the BBS04 and CL04 schemes, are based on the Strong RSA assumption which allows many user keys be issued using the same composite modulus. Mitsunari, Sakai and Kasahara [27] introduced a new computational assumption that was later strengthened by Boneh and Boyen [6] and referred to as q-Strong Diffie-Hellman (q-SDH) assumption. Our proposed group signature schemes are based on the q-SDH assumption and are without a trap-door.

The only other trap-door free scheme is the AdM03 scheme, which uses a trapdoor in the initialisation of the system and assumes that the initialising party "safely forgets" the trapdoor. An advantage of our schemes over AdM03 scheme is that they allow separation of issuer and the opener, hence distribution of trust.

Using elliptic curve cryptography in our schemes results in shorter lengths for signatures and keys. For example, for a comparable level of security as the ACJT00 scheme with 1024 bit composite modulus, our schemes require elliptic curve groups of order 170 bit prime, resulting in the sizes of signatures in our two schemes to be one third and one half, respectively, of the size in ACJT00 scheme. For higher security levels this ratio will be smaller.

Finally in our schemes, the interactive protocol underlying the signature scheme achieves perfect zero-knowledge whereas in ACJT00 and KY04, the corresponding protocols achieve statistical zero-knowledge. We note that all these zero-knowledge proofs including ours, are in honest verifier model. Also, our schemes achieve higher level of unconditional security. That is, given a signature of our schemes, an adversary with unlimited power but without access to the registration table of group members can compute only one part of the signer's private signing key. However in ACJT00 and KY04 schemes, an unlimited adversary can construct the whole private signing key of the signer.

Related Primitives

Group signature schemes are closely related to a number of other cryptographic primitives. They are known to be the non-interactive counterpart of identity escrow systems. In an identity escrow system a user can prove his membership of a group without revealing his identity and anonymity is revocable if a dispute occurs. Most identity escrow systems can be converted into a group signature scheme using the Fiat-Shamir heuristic [17]. Recently, it was shown [21] that traitor tracing schemes can be converted into a group signature scheme [21]. Kiayias et. al. [20] also introduced the Traceable Signature primitive, which is basically the Group Signature system with added properties allowing a variety of levels for protecting user privacy.

The paper is organized as follows. The BSZ04 model of group signature is given in section 2 and other related background is given in section 3. Section 4 describes our group signature scheme and its security proofs. Section 5 gives a modification of BSZ04 formal model and a variant group signature scheme and proves that the variant scheme and the ACJT00 scheme are secure in the modified model. Section 6 provides extensions of the proposed schemes to traceable signatures, schemes with membership revocation and identity escrow and section 7 provides efficiency comparison with ACJT00 scheme. Section 8 concludes the paper.

2 The Model of Group Signature Schemes

We use the BSZ04 formal model. We first describe participants and procedures in this model, then describe oracles accessible to the adversaries and finally define formal security requirements.

2.1 Participants and Procedures

A group signature scheme consists of a trusted party for initial set-up, two group managers (the issuer and the opener), and users with unique identities $i \in \mathbb{N}$ (the set of positive integers). Each user can join the group and become a group member. The scheme is specified as a tuple $\mathcal{GS} = (\mathsf{GKg}, \mathsf{UKg}, \mathsf{Join}, \mathsf{Iss}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge})$ of polynomial-time algorithms described as follows.

- GKg: In the setup phase where the trusted party runs the group-key generation algorithm GKg that takes as input a security parameter 1^l and outputs a triple of keys (gpk, ik, ok), where ik is given to the issuer, and ok is given to the opener. The group public key gpk for signature verification is published.
- UKg: A user i runs the user-key generation algorithm UKg that takes as input a security parameter 1^l and outputs a personal public and private key pair $(upk[i], usk[i])$. The table upk is published.
- Join, Iss: These interactive algorithms are performed by a user, who has a personal public and private key pair, and the issuer as two sides of a group-joining protocol. Each party takes as input an incoming message (unless the party is initiating the protocol) and a current state, and outputs an outgoing message, an updated state, and a decision which is one of accept, reject, cont. The communication is assumed to be secure (i.e., private and authenticated), and the user i is assumed to send the first message. If the issuer accepts, it makes an entry $reg[i]$ for i, in a registration table reg, and fills this entry with a new membership certificate, which is the final state output by Iss. If i accepts, it stores the final state output by Join as its membership secret key $gsk[i]$.
- GSig: A group member i runs the group signing algorithm GSig that takes as input the user's signing key $gsk[i]$ and a message $m \in \{0,1\}^*$ and returns a signature on m.
- GVf: Anyone can run the deterministic group signature verification algorithm GVf on inputs gpk, a message m, and a candidate signature ω for m, to obtain a bit. The signature ω is valid for m with respect to gpk if this bit is 1 (accept).

– Open: The opener, has read-access to the registration table reg, and can run the deterministic opening algorithm Open that takes as input the opening key ok, the registration table reg, a message m, and a valid signature ω of m under gpk and returns a pair (i, τ), where i is a non-negative integer. If $i \geq 1$, the algorithm is claiming that the group member i produced ω and τ is a proof of this claim, and if $i = 0$, it is claiming that no group member produced ω.
– Judge: Anyone can run the deterministic judge algorithm Judge that takes as input the group public key gpk, an integer $j \geq 1$, the public key $upk[j]$ of the user j (this is an empty string if this user has no public key), a message m, a valid signature ω of m, and a proof-string τ. It aims to check that τ is a proof that j produced ω. Note that the judge will base its verification on the public key of j.

2.2 The Oracles

The security requirements are formulated via experiments in which an adversary capabilities are modelled by providing it access to certain oracles. It is assumed that each experiment has run GKg on input 1^l to obtain keys gpk, ik, ok that are used by the oracles, and all entries of the tables upk, reg are assumed initially to be empty strings. It is also assumed that the experiment maintains the following sets which are initially empty and manipulated by the oracles: a set HU of honest users; a set CU of corrupted users; a set GSet of message-signature pairs. Different experiments will provide the adversary with different subsets of the following set of oracles. The oracles are: add user $\mathsf{AddU}(\cdot)$, corrupt user $\mathsf{CrptU}(\cdot,\cdot)$, send to issuer $\mathsf{SndToI}(\cdot,\cdot)$, send to user $\mathsf{SndToU}(\cdot,\cdot)$, user secret keys $\mathsf{USK}(\cdot)$, read registration table $\mathsf{RReg}(\cdot)$, write registration table $\mathsf{WReg}(\cdot,\cdot)$, signing oracle $\mathsf{GSig}(\cdot,\cdot)$, challenge oracle $\mathsf{Ch}(b,\cdot,\cdot,\cdot)$ and open oracle $\mathsf{Open}(\cdot,\cdot)$. Their descriptions are provided in Appendix A.

2.3 Security Requirements

The security requirements are modelled by experiments. We briefly recall the requirements and formulas of experiments and refer the reader to [5] for further detail. A group signature scheme must satisfy the following security requirements:

– $\mathsf{Correctness}$: In this experiment the adversary is not computationally restricted and has access to $\mathsf{AddU}(\cdot)$ and $\mathsf{RReg}(\cdot)$ oracles. The adversary returns a message and the identity of an honest group member and the group member produces a signature of the message. The correctness condition holds if the probability that one of the following steps fails is 0: given the message and signature, GVf algorithm accepts the signature; Open algorithm returns the correct group member; and Judge algorithm accepts the proof returned by Open algorithm.

– Anonymity: The anonymity experiment involves a polynomial-time adversary, who knows the issuing key ik and has access to $\mathsf{Ch}(b,\cdot,\cdot,\cdot)$, $\mathsf{Open}(\cdot,\cdot)$, $\mathsf{SndToI}(\cdot,\cdot)$, $\mathsf{SndToU}(\cdot,\cdot)$, $\mathsf{WReg}(\cdot,\cdot)$, $\mathsf{USK}(\cdot)$ and $\mathsf{CrptU}(\cdot,\cdot)$ oracles. The adversary provides the $\mathsf{Ch}(b,\cdot,\cdot,\cdot)$ oracle identities of two honest members and a message and is returned a signature of the message generated by one of the members (according to bit b). The anonymity condition holds if the probability that the adversary can correctly guess the bit b is negligible. Note that the adversary can not send the challenge signature to $\mathsf{Open}(\cdot,\cdot)$ oracle and the opener is uncorrupt.
– Traceability: The traceability experiment involves a polynomial-time adversary, who knows the opening key ok and has access to $\mathsf{AddU}(\cdot)$, $\mathsf{RReg}(\cdot)$, $\mathsf{SndToI}(\cdot,\cdot)$, $\mathsf{USK}(\cdot)$ and $\mathsf{CrptU}(\cdot,\cdot)$ oracles. The adversary returns a message and a signature. The traceability condition holds if the probability that all of the following steps succeed is negligible: given the message and signature, GVf algorithm accepts the signature; Open algorithm can not return the identity of the signer, or Open algorithm can return the identity of the signer but Judge algorithm rejects the proof returned by Open algorithm. Note that the issuer is uncorrupt and the opener is at worst partially corrupted, that means he performs correctly but his secret key is available to the adversary.
– Non-frameability: The non-frameability experiment involves a polynomial-time adversary, who knows the opening key ok and the issuing key ik, and has access to $\mathsf{SndToU}(\cdot,\cdot)$, $\mathsf{WReg}(\cdot,\cdot)$, $\mathsf{GSig}(\cdot,\cdot)$, $\mathsf{USK}(\cdot)$ and $\mathsf{CrptU}(\cdot,\cdot)$ oracles. The adversary returns a message, a signature, an identity of an honest group member and a proof of an opening claim. The non-frameability condition holds if the probability that the following steps succeed is negligible: GVf algorithm accepts the signature; and Judge algorithm accepts the proof returned by the adversary, who claims that the honest group member is the signer. Note that the adversary can not send the challenge member identity and the challenge message to $\mathsf{USK}(\cdot)$ and $\mathsf{GSig}(\cdot,\cdot)$.

3 Preliminaries

In this section, we first briefly describe groups from bilinear pairing, their properties and then present two bilinear pairing versions for El Gamal public key system (El Gamal^{BP1} and El Gamal^{BP2}), one provides Indistinguishability against adaptive Chosen Plaintext Attack (IND-CPA) and the other provides Indistinguishability against adaptive Chosen Ciphertext Attack (IND-CCA).

Appendix B presents the well-known Forking Lemma [30], the random oracle model and complexity assumptions that are used to prove security of our group signature schemes. Descriptions of Public-key Encryption and Digital Signature Primitives and their security requirements, including IND-CPA and IND-CCA for Public-key Encryption schemes and Unforgeability against Chosen Message Attack (UNF-CMA) for Digital Signature schemes, can be founded in [19].

3.1 Bilinear Pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic additive groups generated by P_1 and P_2, respectively, both with order p, a prime, and $\mathbb{G}_M$ be a cyclic multiplicative group with the same order. Suppose there is an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(P_2) = P_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_M$ be a bilinear pairing with the following properties:

1. **Bilinearity:** $e(aP, bQ) = e(P, Q)^{ab}$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b \in \mathbb{Z}_p$
2. **Non-degeneracy:** $e(P_1, P_2) \neq 1$
3. **Computability:** There is an efficient algorithm to compute $e(P, Q)$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$

For simplicity, hereafter, we set $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$ but our group signature schemes can be easily modified for the case when $\mathbb{G}_1 \neq \mathbb{G}_2$. For a group $\mathbb{G}$ of prime order, hereafter, we denote the set $\mathbb{G}^* = \mathbb{G}\backslash\{\mathcal{O}\}$ where $\mathcal{O}$ is the identity element of the group.

We define a Bilinear Pairing Instance Generator as a Probabilistic Polynomial Time (PPT) algorithm $\mathcal{G}$ that takes as input a security parameter 1^l and returns a uniformly random tuple $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P)$ of bilinear pairing parameters, including a prime number p of size l, a cyclic additive group $\mathbb{G}_1$ of order p, a multiplicative group $\mathbb{G}_M$ of order p, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_M$ and a generator P of $\mathbb{G}_1$. Hereafter, unless stated otherwise, we assume all computations of elements in $\mathbb{Z}_p$ are in modulo p.

3.2 Bilinear Pairing versions of El Gamal public key system

El Gamal^{BP1}

Key generation: Let $p, \mathbb{G}_1, \mathbb{G}_M, e$ be bilinear pairing parameters, as defined above, and G be a generator of $\mathbb{G}_1$. Suppose $x \in_R \mathbb{Z}_p^*$ and $\Theta = e(G, G)^x$. The public key $pk = (G, \Theta)$ and the secret key is $sk = x$.
Encryption: Plaintext $\Delta \in \mathbb{G}_M$ can be encrypted by choosing an $t \in_R \mathbb{Z}_p^*$ and computing the ciphertext $(E, \Lambda) = (tG, \Delta\Theta^t)$.
Decryption: Ciphertext (E, Λ) can be decrypted as $\Delta = \Lambda / e(E, G)^x$.
Security: The security of El Gamal^{BP1} system is stated in Theorem 1. The first statement can be proved exactly the same way as the proof for the El Gamal encryption scheme [33], except that it is based on DDHV assumption (see Appendix B) instead of DDH assumption. The second statement can be seen as a result of the first statement and Theorem 10.

Theorem 1. *El Gamal^{BP1} encryption scheme is IND-CPA if and only if DDHV assumption holds. El Gamal^{BP1} encryption scheme is IND-CPA if DBDH assumption holds.*

El Gamal^{BP2}

We next present an extension, El Gamal^{BP2}, which is IND-CCA in the random oracle model. This is the bilinear pairing version of the scheme presented and proved by Fouque and Pointcheval [18], that uses the twin-encryption paradigm of [28] and a simulation-sound proof of equality of plaintexts.

Key generation: Let $p, \mathbb{G}_1, \mathbb{G}_M, e$ be bilinear pairing parameters, as defined above, and G be a generator of $\mathbb{G}_1$. Suppose $x_a, x_b \in_R \mathbb{Z}_p^*$ and $\Theta_a = e(G,G)^{x_a}$ and $\Theta_b = e(G,G)^{x_b}$. The public key $pk = (G, \Theta_a, \Theta_b)$ and the secret key is $sk = (x_a, x_b)$. Choose a hash function $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{Z}_p$ (a random oracle).
Encryption: Plaintext $\Delta \in \mathbb{G}_M$ can be encrypted by choosing $t_a, t_b \in_R \mathbb{Z}_p^*$ and computing $(E_a, \Lambda_a) = (t_a G, \Delta\Theta_a^{t_a})$, $(E_b, \Lambda_b) = (t_b G, \Delta\Theta_b^{t_b})$ and a non-interactive zero-knowledge proof $\varsigma = (c, \rho_a, \rho_b)$ of equality of plaintexts between (E_a, Λ_a) and (E_b, Λ_b). The proof ς can be computed by choosing $w_a, w_b \in_R \mathbb{Z}_p$ and computing $c = \mathcal{H}_1(G||\Theta_a||\Theta_b||E_a||\Lambda_a||E_b||\Lambda_b||w_a G||w_b G||\Theta_a^{w_a}\Theta_b^{w_b})$, $\rho_a = w_a - t_a c$ and $\rho_b = w_b + t_b c$. The ciphertext is $(E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$.
Decryption: Given a ciphertext $(E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$, first check the validity of ς by verifying

$$c \stackrel{?}{=} \mathcal{H}_1(G||\Theta_a||\Theta_b||E_a||\Lambda_a||E_b||\Lambda_b||\rho_a G + cE_a||\rho_b G - cE_b||\Theta_a^{\rho_a}\Theta_b^{\rho_b}(\Lambda_a/\Lambda_b)^c)$$

then compute the plaintext $\Delta = \Lambda_a/e(E_a, G)^{x_a} = \Lambda_b/e(E_b, G)^{x_b}$.
Security: The security of El Gamal^{BP2} system is stated in Theorem 2. The proof of the first statement is the same as the proof in [18], except that it is based on DDHV assumption instead of DDH assumption. The second statement can be seen as a result of the first statement and Theorem 10.

Theorem 2. *El Gamal^{BP2} encryption scheme is IND-CCA if DDHV assumption holds, in the random oracle model. El Gamal^{BP2} encryption scheme is IND-CCA if DBDH assumption holds, in the random oracle model.*

4 The Group Signature scheme

4.1 Overview

Our group signature scheme is built upon two ordinary signature schemes. The first one is used in the Join, Iss protocol for the issuer to generate a signature (a_i, S_i) for each x_i, which is randomly generated by both a member and the issuer, but known only to the member. The second ordinary signature scheme is used in the GSig algorithm as the non-interactive version of a zero-knowledge protocol, that proves the signer's knowledge of (a_i, S_i) and x_i. The security of the two signature schemes underlies the security of the group signature scheme.

Our group signature scheme is constructed in cyclic groups with bilinear mappings. For simplicity, we present the scheme when the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are the same, however, it can be very easily modified for the general case when $\mathbb{G}_1 \neq \mathbb{G}_2$. The pairing operation play an important role in the verification algorithm GVf. Intuitively, bilinear pairings allow a party, given $A, B, C, D \in \mathbb{G}_1$, to prove that $log_A B = log_C D$ without knowing $log_A B$ or $log_A C$. This is not possible in cyclic groups without bilinear pairings and where the DDH assumption holds.

4.2 Descriptions

Our group signature scheme uses a trusted party in the initial set-up, two group managers (the issuer and the opener), and users, each with a unique identity $i \in \mathbb{N}$, that may become group members. The scheme is a tuple $\mathcal{GS}1$ =(GKg, UKg, Join, Iss, GSig, GVf, Open, Judge) of polynomial-time algorithms which are defined as follows.

GKg: Suppose l is a security parameter and the Bilinear Pairing Instance Generator $\mathcal{G}$ generates a tuple of bilinear pairing parameters $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$, that is also the publicly shared parameters. Choose a hash function $\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, which is assumed to be a random oracle in the security proofs.
Choose $P_0, G, H \in_R \mathbb{G}_1$, $x, x'_a, x'_b \in_R \mathbb{Z}_p^*$ and compute $P_{pub} = xP$, $\Theta_a = e(G,G)^{x'_a}$ and $\Theta_b = e(G,G)^{x'_b}$. The group public key is $gpk =(P, P_0, P_{pub}, H, G, \Theta_a, \Theta_b)$, the issuing key is $ik = x$, and the opening key is $ok = (x'_a, x'_b)$.

UKg: This algorithm generates keys that provide authenticity for messages sent by the user in the (Join, Iss) protocol. This algorithm is the key generation algorithm K_S of any digital signature scheme (K_S, $Sign$, Ver) that is unforgeable against chosen message attacks (UNF-CMA). A user i runs the UKg algorithm that takes as input a security parameter 1^l and outputs a personal public and private signature key pair $(upk[i], usk[i])$. Public Key Infrastructure (PKI) can be used here. Although any UNF-CMA signature scheme can be used, but using schemes, whose security is based on DBDH or SDH assumptions, will reduce the underlying assumptions of our group signature scheme. One example of such scheme is in [6].

Join, Iss: In this protocol, a user i and the issuer first jointly generate a random value $x_i \in \mathbb{Z}_p^*$ whose value is only known by the user. The issuer then generates (a_i, S_i) for the user so that $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$. The user uses $usk[i]$ to sign his messages in the protocol. Note that the formal model assumes the communication to be private and authenticated. We also assume that the communication is protected from replay attacks. The protocol is as follows.

1. $\mathsf{user}\ i \longrightarrow \mathsf{issuer}$: $I = yP + rH$, where $y, r \in_R \mathbb{Z}_p^*$.
2. $\mathsf{user}\ i \longleftarrow \mathsf{issuer}$: $u, v \in_R \mathbb{Z}_p^*$.
3. The user computes $x_i = uy + v$, $P_i = x_iP$.
4. $\mathsf{user}\ i \longrightarrow \mathsf{issuer}$: P_i and a proof of knowledge of (x_i, r') such that $P_i = x_iP$ and $vP + uI - P_i = r'H$ (see [13] for this proof).
5. The issuer verifies the proof, then chooses $a_i \in_R \mathbb{Z}_p^*$ different from all corresponding elements previously issued, and computes $S_i = \frac{1}{a_i+x}(P_i + P_0)$.
6. $\mathsf{user}\ i \longleftarrow \mathsf{issuer}$: a_i, S_i.
7. The user computes $\Delta_i = e(P, S_i)$, verifies if $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$, and stores the *private signing key* $gsk[i] = (x_i, a_i, S_i, \Delta_i)$. Note that only the user knows x_i. The issuer also computes Δ_i and makes an entry in the table reg: $reg[i] = (i, \Delta_i, \langle\mathsf{Join}, \mathsf{Iss}\rangle$ transcript).

GSig: A group signature of a user i shows his knowledge of (a_i, S_i) and a secret x_i such that: $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$. The signature does not reveal any information about his knowledge to anyone, except for the opener, who can compute Δ_i by decrypting an encryption of that value. The algorithm for a user i to sign a message $m \in \{0,1\}^*$ is as follows.

1. Encrypt Δ_i by El Gamal^{BP2} with public key (G, Θ_a, Θ_b) as $(E_a = tG, \Lambda_a = \Delta_i\Theta_a^t, E_b, \Lambda_b, \varsigma)$.
2. Perform the non-interactive version of a protocol, which we call the Signing protocol, as follows.
   (a) Generate $r, r', k_0, ..., k_6 \leftarrow \mathbb{Z}_p$ and compute:
   $$V = S_i + rH;\ R = rG + r'H;\ T_1 = k_1G + k_2H;$$
   $$T_2 = k_3G + k_4H - k_5R;\ T_3 = k_6G;$$
   $$\Pi_1 = e(P,P)^{k_0}e(P,V)^{-k_5}e(P,H)^{k_3}e(P_{pub},H)^{k_1};\ \Pi_2 = e(P,H)^{-k_1}\Theta_a^{k_6}$$
   (b) Compute $c = \mathcal{H}_2(P||P_0||P_{pub}||H||G||\Theta_a||\Theta_b||E_a||\Lambda_a||E_b||\Lambda_b||\varsigma||V||R||T_1 ||T_2||T_3||\Pi_1||\Pi_2||m)$
   (c) Compute in $\mathbb{Z}_p$: $s_0 = k_0 + cx_i$; $s_1 = k_1 + cr$; $s_2 = k_2 + cr'$; $s_3 = k_3 + cra_i$; $s_4 = k_4 + cr'a_i$; $s_5 = k_5 + ca_i$; $s_6 = k_6 + ct$
3. Output the signature $(c, s_0, ..., s_6, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ for message m.

GVf: The verification algorithm for $m, (c, s_0, ..., s_6, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ outputs accept if and only if verifying the proof ς outputs accept and the following equation holds.

$$\begin{aligned} c = \mathcal{H}_2(&P||P_0||P_{pub}||H||G||\Theta_a||\Theta_b||E_a||\Lambda_a||E_b||\Lambda_b||\varsigma||V||R|| \\ &s_1G + s_2H - cR||s_3G + s_4H - s_5R||s_6G - cE_a|| \\ &e(P,P)^{s_0}e(P,V)^{-s_5}e(P,H)^{s_3}e(P_{pub},H)^{s_1}e(P,P_0)^c e(P_{pub},V)^{-c}|| \\ &e(P,H)^{-s_1}\Theta_a^{s_6}\Lambda_a^{-c}e(P,V)^c||m) \end{aligned}$$

Open: To open m and its valid signature $(c, s_0, ..., s_6, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ to find the signer, the opener performs the following steps.

1. Use GVf algorithm to check the signature's validity. If the algorithm rejects, return $(0, \varepsilon)$, where ε denotes an empty string.
2. Compute $\Delta_i = \Lambda_a e(E_a, G)^{-x'_a}$ and find the corresponding entry i in the table reg. If no entry is found, return $(0, \varepsilon)$.
3. Return $reg[i]$ and a non-interactive zero-knowledge proof ϱ of knowledge of x'_a so that $\Theta_a = e(G,G)^{x'_a}$ and $\Lambda_a/\Delta_i = e(E_a, G)^{x'_a}$ (see [13] for this proof).

Judge: On an output by the Open algorithm for a message m and its signature ω, the Judge algorithm is performed as follows:

1. If Open algorithm outputs $(0, \varepsilon)$, run GVf algorithm on m, ω. If GVf rejects, return accept; otherwise, return reject.

2. If Open algorithm outputs $(reg[i], \varrho)$, return reject if one of the following happens: (i) on m, ω, GVf algorithm rejects; (ii) verification of the proof ϱ rejects; (iii) the $\langle \mathsf{Join}, \mathsf{Iss} \rangle$ transcript is invalid with regard to $upk[i]$; (iv) $\Delta_i \neq e(P, S_i)$ where S_i is extracted from the $\langle \mathsf{Join}, \mathsf{Iss} \rangle$ transcript. Otherwise, return accept.

Remarks:

- Our scheme is trapdoor-free. This improves efficiency and manageability, and various groups can share the same initial set-up $p, \mathbb{G}_1, \mathbb{G}_M, e, P, P_0, G, H$.
- In most previous schemes, including the ACJT00 and KY04 schemes, the protocol underlying the GSig algorithm is statistically zero-knowledge (under the Strong RSA assumption). Our Signing protocol is perfectly zero-knowledge. This indicates a higher level of unconditional security: from a signature, an adversary with unlimited power (but without access to the reg table) can compute only a part of the signer's registration information (S_i), whereas, in the ACJT00 and KY04 schemes, the adversary can find all parts of the signer's private signing key.
- Threshold Open is also possible by using a Threshold Encryption scheme similar to the scheme in [18].

4.3 Security Proofs

Security of the group signature scheme $\mathcal{GS}1$ is stated in Theorems 3, 4, 5 and 6. Proofs of Theorems 4, 5 and 6 are provided in Appendix C. Theorem 3 can easily be proved by checking equations.

Theorem 3. *The group signature scheme $\mathcal{GS}1$ provides Correctness.*

Theorem 4. *The group signature scheme $\mathcal{GS}1$ provides Anonymity in the random oracle model if the Decisional Bilinear Diffie-Hellman assumption holds.*

Theorem 5. *The group signature scheme $\mathcal{GS}1$ provides Traceability in the random oracle model if the q-Strong Diffie-Hellman assumption holds, where q is the upper bound of the group size.*

Theorem 6. *The group signature scheme $\mathcal{GS}1$ provides Non-frameability in the random oracle model if the Discrete Logarithm assumption holds over the group $\mathbb{G}_1$ and the digital signature scheme (K_S, $Sign$, Ver) is UNF-CMA.*

5 Variations

In this section, we propose Weak Anonymity requirement as an alternative for Anonymity requirement. We then present a second group signature scheme, $\mathcal{GS}2$, and prove that it provides Weak Anonymity, Traceability and Non-Frameability. We also prove that the ACJT00 scheme provides the same properties. We also discuss the possibility that the ACJT00 and $\mathcal{GS}2$ schemes provide Anonymity.

5.1 Weak Anonymity requirement

We introduce this security requirement to account for a class of group signature schemes, including ACJT00 scheme, which can not be proved to achieve Anonymity requirement.

Weak Anonymity requirement is defined exactly the same as Anonymity requirement, except that the adversary does not have access to the $\mathsf{Open}(\cdot,\cdot)$ oracle. In practice, when the opener is assumed to be uncorrupted as in Anonymity requirement, it could be hard for the adversary to have access to the Open oracle. As Open oracle is not used in the conventional list of requirements, the same argument as in [4, 5] shows that Weak anonymity, Traceability and Non-frameability are sufficient to imply the conventional list of requirements. For a group signature scheme $\mathcal{GS}$, an adversary $\mathcal{A}$, a bit $b \in \{0,1\}$ and a security parameter $l \in N$, the experiment for Weak Anonymity is as follows.

Experiment $Exp^{\mathsf{weak.anon\text{-}b}}_{\mathcal{GS},\mathcal{A}}(l)$ // $b \in \{0,1\}$
$(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^l)$; $\mathsf{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$; $\mathsf{GSet} \leftarrow \emptyset$
$d \leftarrow \mathcal{A}(gpk, ik : \mathsf{Ch}(b,\cdot,\cdot,\cdot),\ \mathsf{SndToI}(\cdot,\cdot),\ \mathsf{SndToU}(\cdot,\cdot),\ \mathsf{WReg}(\cdot,\cdot),\ \mathsf{USK}(\cdot),\ \mathsf{CrptU}(\cdot,\cdot))$
Return d

The group signature scheme $\mathcal{GS}$ provides Weak Anonymity if the following function $Adv^{\mathsf{weak.anon}}_{\mathcal{GS},A}(l)$ is negligible.

$$Adv^{\mathsf{weak.anon}}_{\mathcal{GS},A}(l) = |\Pr[Exp^{\mathsf{weak.anon\text{-}1}}_{\mathcal{GS},A}(l) = 1] - \Pr[Exp^{\mathsf{weak.anon\text{-}0}}_{\mathcal{GS},A}(l) = 1]|$$

5.2 A Variant Group Signature scheme, $\mathcal{GS}2$

The scheme $\mathcal{GS}2$ is the same as $\mathcal{GS}1$, except that in the signature, Δ_i is encrypted by El Gamal^{BP1} encryption scheme instead of El Gamal^{BP2}. So in GKg, x'_b and Θ_b are not generated and in GSig, Δ_i is encrypted by El Gamal^{BP1} public key (G, Θ_a) as $(E_a = tG, \Lambda_a = \Delta_i \Theta_a^t)$. So there is no E_b, Λ_b or ς in the signature and in the executions of GSig, GVf, Open and Judge algorithms. Security of $\mathcal{GS}2$ is stated in Theorem 7, whose proof is shown in Appendix C.

Theorem 7. *$\mathcal{GS}2$ provides Correctness. $\mathcal{GS}2$ provides Weak Anonymity if the Decisional Bilinear Diffie-Hellman assumption holds. $\mathcal{GS}2$ provides Traceability in the random oracle model if the q-Strong Diffie-Hellman assumption holds, where q is the upper bound of the group size. $\mathcal{GS}2$ provides Non-frameability in the random oracle model if the Discrete Logarithm assumption holds over the group $\mathbb{G}_1$ and the digital signature scheme (K_S, $Sign$, Ver) is UNF-CMA.*

5.3 Do ACJT00 and $\mathcal{GS}2$ schemes provide Anonymity?

We first state the security of the ACJT00 scheme in Theorem 8. The ACJT00 scheme refers to the scheme proposed in [1], plus some simple extensions to accommodate the Judge algorithm (defining the UKg algorithm as in our scheme, using $usk[i]$ to sign messages in the Join, Iss protocol, and verifying signatures in the Open and Judge algorithms). The methodology of the proof for Theorem 8 is very similar to the proof of Theorem 7, and the exact details of each step can be extracted from the proofs in [22].

Theorem 8. *The ACJT00 scheme provides Correctness; Weak Anonymity if the DDH-Compo-KF assumption holds; Traceability in the random oracle model if the Strong RSA assumption holds; Non-frameability in the random oracle model if the Discrete Logarithm assumption holds over the quadratic residues group of a product of two known large primes, and the digital signature scheme for* UKg *is UNF-CMA. (See [22] for assumptions used in this theorem).*

It is an open question if the ACJT00 and $\mathcal{GS}2$ schemes provide Anonymity, in line with the open problem whether a combination of an El Gamal encryption (IND-CPA) and a Schnorr proof of knowledge of the plaintext can provide IND-CCA. This combination has been proved to provide IND-CCA in the random oracle model, but the proof has required either another very strong assumption [33] or is in generic model [31]. In ACJT00 and $\mathcal{GS}2$ signatures, the identity-bound information is encrypted by variations of El Gamal encryption and the other part of the signatures proves knowledge of the information. The Open oracle plays a similar role as the Decryption oracle in the model of IND-CCA.

5.4 Variants based on the DDH assumption

We can build variants of $\mathcal{GS}1$ and $\mathcal{GS}2$, whose security is based on the DDH assumption over the group $\mathbb{G}_M$ instead of the DBDH (DDHV) assumption. Specifically, Δ_i will be encrypted by the normal El Gamal encryption scheme or the twin-paradigm extension of El Gamal encryption scheme (proposed in [18]). The Open algorithm in these variant schemes requires one less pairing operation than in $\mathcal{GS}1$ and $\mathcal{GS}2$.

We can actually provide a group signature with 4 options, where the users, the issuer and the opener use the same keys for all options. The first two options are $\mathcal{GS}1$ and $\mathcal{GS}2$, offering smaller signature size and more efficient signing and verification. The last two options are the variant schemes based on the normal DDH assumption, with more efficient opening.

6 Extensions

6.1 A Traceable Signature scheme

We extend $\mathcal{GS}2$ to be a traceable signature scheme $\mathcal{TS}$ =(Setup, Join, Sign, Verify, Open, Reveal, Trace, Claim, Claim-Verify) with similar advantages over the only other traceable signature scheme [20]. We provide background about Traceable signatures in Appendix A.

Setup: This is the same as GKg for $\mathcal{GS}2$, but the group public key also includes a $Q \in_R \mathbb{Z}_p^*$. The group public key is $gpk = (P, P_0, P_{pub}, Q, H, G, \Theta_a)$, the issuing key is $ik = x$, and the opening key is $ok = x'_a$. Choose a hash function $\mathcal{H}_3 : \{0,1\}^* \to \mathbb{Z}_p$ (a random oracle).

Join: This protocol is very similar to the Join, Iss protocol in Section 4.2 and described as follows:

1. user $i \longrightarrow$ GM: $I = yP + rH$, where $y, r \in_R \mathbb{Z}_p^*$.
2. user $i \longleftarrow$ GM: $u, v \in_R \mathbb{Z}_p^*$.
3. The user computes $x_i = uy + v$, $P_i = x_i P$.
4. user $i \longrightarrow$ GM: P_i and a proof of knowledge of (x_i, r') such that $P_i = x_i P$ and $vP + uI - P_i = r'H$ (see [13] for this proof).
5. The GM verifies the proof, then chooses $a_i, \bar{x_i} \in_R \mathbb{Z}_p^*$ so that a_i is different from all corresponding elements previously issued, and computes $S_i = \frac{1}{a_i+x}(P_i + \bar{x_i}Q + P_0)$.
6. user $i \longleftarrow$ GM: $a_i, S_i, \bar{x_i}$.
7. The user computes $\Delta_i = e(P, S_i)$, verifies if $e(a_i P + P_{pub}, S_i) = e(P, x_i P + \bar{x_i}Q + P_0)$, and stores the *private signing key* $gsk[i] = (x_i, \bar{x_i}, a_i, S_i, \Delta_i)$. Note that only the user knows x_i. The GM also computes Δ_i and stores it with the protocol's transcript.

Sign: The algorithm for an user i to sign a message $m \in \{0,1\}^*$ is as follows.

1. Compute $E_a = tG$, $\Lambda_a = \Delta_i \Theta_a^t$, $\Upsilon_1 = \Theta_a^{\bar{x_i}r}$, $\Upsilon_2 = \Theta_a^r$, $\Upsilon_3 = \Theta_a^{x_i r'}$ and $\Upsilon_4 = \Theta_a^{r'}$, where $t, r, r' \in_R \mathbb{Z}_p^*$.
2. Generate $r_1, r_2, k_0, ..., k_7 \leftarrow \mathbb{Z}_p$ and compute:
   (a) $V = S_i + r_1 H$; $R = r_1 G + r_2 H$; $T_1 = k_1 G + k_2 H$; $T_2 = k_3 G + k_4 H - k_5 R$; $T_3 = k_6 G$; $\Pi_1 = e(P,Q)^{k_7} e(P,P)^{k_0} e(P,V)^{-k_5} e(P,H)^{k_3} e(P_{pub},H)^{k_1}$; $\Pi_2 = e(P,H)^{-k_1} \Theta_a^{k_6}$; $\Pi_3 = \Upsilon_2^{k_7}$; $\Pi_4 = \Upsilon_4^{k_0}$
   (b) $c = \mathcal{H}_3(P||P_0||P_{pub}||H||G||\Theta_a||E_a||\Lambda_a||V||R||T_1||T_2||T_3||\Pi_1||\Pi_2||\Pi_3||\Pi_4 ||m)$
   (c) Compute in $\mathbb{Z}_p$: $s_0 = k_0 + cx_i$; $s_1 = k_1 + cr_1$; $s_2 = k_2 + cr_2$; $s_3 = k_3 + cr_1 a_i$; $s_4 = k_4 + cr_2 a_i$; $s_5 = k_5 + ca_i$; $s_6 = k_6 + ct$; $s_7 = k_7 + c\bar{x_i}$
3. Output the signature $(c, s_0, ..., s_7, V, R, E_a, \Lambda_a, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4)$ for message m.

Verify: The verification algorithm for $m, (c, s_0, ..., s_7, V, R, E_a, \Lambda_a, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4)$ outputs accept if and only if the following equation holds: $c = \mathcal{H}_3(P||P_0||P_{pub}||H ||G||\Theta_a||E_a||\Lambda_a||V||R||s_1 G + s_2 H - cR||s_3 G + s_4 H - s_5 R||s_6 G - cE_a||e(P,Q)^{s_7} e(P,P)^{s_0} e(P,V)^{-s_5} e(P,H)^{s_3} e(P_{pub},H)^{s_1} e(P,P_0)^c e(P_{pub},V)^{-c}||e(P,H)^{-s_1}\Theta_a^{s_6} \Lambda_a^{-c} e(P,V)^c||\Upsilon_2^{s_7}\Upsilon_1^{-c}||\Upsilon_4^{s_0}\Upsilon_3^{-c}||m)$.

Open: To open m and its valid signature $(c, s_0, ..., s_7, V, R, E_a, \Lambda_a, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4)$ to find the signer, the GM computes $\Delta_i = \Lambda_a e(E_a, G)^{-x'_a}$ and finds the corresponding entry i in the table of stored Join transcripts. The GM returns i and a non-interactive zero-knowledge proof ϱ of knowledge of x'_a so that $\Theta_a = e(G,G)^{x'_a}$ and $\Lambda_a/\Delta_i = e(E_a, G)^{x'_a}$ (see [13] for this proof).

Reveal and Trace: Given the Join transcript of user i, the GM recovers the tracing trapdoor $trace_i = \bar{x}_i$. Given $trace_i$ and a message-signature pair, a designated party recovers Υ_1 and Υ_2 and checks if $\Upsilon_1 = \Upsilon_2^{\bar{x}_i}$. If the equation holds, the tracer concludes that user i has produced the signature.

Claim and Claim-Verify: Given a message-signature pair, a user i can claim that he is the signer by recovering Υ_3 and Υ_4 and producing a non-interactive proof of knowledge of the discrete-log of Υ_3 base Υ_4. Any party can run Claim-Verify by verifying the signature and the proof.

Security The security of $\mathcal{TS}$ is stated in Theorem 9. The proof of this theorem uses techniques similar to those in [20] and arguments similar to those in Appendix C.

Theorem 9. *In the random oracle model, $\mathcal{TS}$ provides (i) security against misidentification attacks based on the q-SDH and the DDH assumptions, where q is the upper bound of the group size; (ii) security against anonymity attacks based on the DBDH and DDH assumptions; (iii) security against framing attacks based on the DL assumption.*

6.2 Group Signature schemes with Membership Revocation

As shown in [29], our group signature schemes can be extended to support efficient membership revocation. That means the issuer can remove members from the group and the cost of removing does not depend on the size of the group. It is also possible to use the accumulator scheme in [11] to provide membership revocation. More specifically, at step 5 of the Join, Iss protocol, the issuer generates a prime a'_i in the range of values accumulatable by the dynamic accumulator, so that the value $a_i = a'_i \bmod p$ ($a_i \in \mathbb{Z}_p^*$) is different from all corresponding elements previously issued. The value to be accumulated is a'_i. Security of the new schemes also depends on Strong-RSA assumption that underlies the security of the dynamic accumulator.

6.3 Identity Escrow schemes

Identity escrow schemes can be derived directly from the group signature schemes. Specifically, the GSig and GVf algorithms are replaced by the corresponding interactive protocol between a group member and a verifier, where the random challenge c is generated by the verifier instead of being computed from the hash function. Note that in this protocol, verification can start right after the first round by checking $e(U, V) = e(P, W)$ and it can be done concurrent with the next rounds.

7 Efficiency

The sizes of signatures and keys in our schemes are much shorter than those used in the Strong-RSA-based schemes at a similar level of security. This difference grows when higher level of security is required. In this section, we compare sizes in our new group signature schemes with those in ACJT00 scheme.

We assume that our scheme is implemented using an elliptic curve or hyperelliptic curve over a finite field. p is a 170-bit prime, $\mathbb{G}_1$ is a subgroup of an elliptic curve group or a Jacobian of a hyperelliptic curve over a finite field of order p. $\mathbb{G}_M$ is a subgroup of a finite field of size approximately 2^{1024}. A possible choice for these parameters can be found in [9], where $\mathbb{G}_1$ is derived from the curve $E/GF(3^\iota)$ defined by $y^2 = x^3 - x + 1$. We assume that system parameters in ACJT00 scheme are $\epsilon = 1.1$, $l_p = 512$, $k = 160$, $\lambda_1 = 838$, $\lambda_2 = 600$, $\gamma_1 = 1102$ and $\gamma_2 = 840$. We summarize the result in Table 1.

Table 1. Comparison of sizes (in Bytes)

	Signature	gpk	gsk	ik	ok	Security
ACJT00	1087	768	370	128	128	Weak Anonymity
$\mathcal{GS}1$	574	363	192	22	44	Anonymity
$\mathcal{GS}2$	361	235	192	22	22	Weak Anonymity

8 Conclusions

We proposed new group signature schemes from bilinear pairings and proved their security in BSZ04 formal model. The new schemes have shorter sizes for signatures and keys, are trapdoor-free and provide higher level of unconditional security for signers. We also extended the schemes to achieve membership revocation and constructed a traceable signature scheme and identification escrow systems.

Acknowledgements. Authors thank anonymous referees of Asiacrypt 2004 for constructive comments and Fangguo Zhang for helpful discussions.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. CRYPTO 2000, Springer-Verlag, LNCS 1880, pp. 255-270.
2. G. Ateniese, and B. de Medeiros. Efficient Group Signatures without Trapdoors. ASIACRYPT 2003, Springer-Verlag, LNCS 2894, pp. 246-268.
3. G. Ateniese, and B. de Medeiros. Security of a Nyberg-Rueppel Signature Variant. Cryptology ePrint Archive, Report 2004/093, http://eprint.iacr.org/.
4. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. EUROCRYPT 2003, Springer-Verlag, LNCS 2656, pp. 614-629.

5. M. Bellare, H. Shi, and C. Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. Cryptology ePrint Archive: Report 2004/077.
6. D. Boneh, and X. Boyen. Short Signatures Without Random Oracles. EUROCRYPT 2004, Springer-Verlag, LNCS 3027, pp. 56-73.
7. D. Boneh, and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. EUROCRYPT 2004, Springer-Verlag, LNCS 3027, pp. 223-238.
8. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. CRYPT0 2004, Springer-Verlag, LNCS, to appear.
9. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. ASIACRYPT 2001, Springer-Verlag, LNCS 2248, pp.514-532.
10. J. Camenisch. Efficient and generalized group signatures. EUROCRYPT 1997, Springer-Verlag, LNCS 1233, pp. 465-479.
11. J. Camenisch, and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. CRYPTO 2002, Springer-Verlag, LNCS 2442, pp. 61-76.
12. J. Camenisch, and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. CRYPTO 2004, Springer-Verlag, LNCS, to appear.
13. J. Camenisch, and M. Michels. A group signature scheme with improved efficiency. ASIACRYPT 1998, Springer-Verlag, LNCS 1514.
14. J. Camenisch, and M. Stadler. Efficient group signature schemes for large groups. CRYPTO 1997, Springer-Verlag, LNCS 1296.
15. D. Chaum, and E. van Heyst. Group signatures. CRYPTO 1991, LNCS 547, Springer-Verlag.
16. L. Chen, and T. P. Pedersen. New group signature schemes. EUROCRYPT 1994, Springer-Verlag, LNCS 950, pp. 171-181.
17. A. Fiat, and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. CRYPTO 1986, Springer-Verlag, LNCS 263, pp. 186-194.
18. P. Fouque and D. Pointcheval, Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks, Asiacrypt 2001. LNCS 2248.
19. O. Goldreich. Foundations of Cryptography, Basic Applications. Cambridge University Press 2004.
20. A. Kiayias, Y. Tsiounis and M. Yung. Traceable Signatures. EUROCRYPT 2004, Springer-Verlag, LNCS 3027, pp. 571-589.
21. A. Kiayias, and Moti Yung. Extracting Group Signatures from Traitor Tracing Schemes. EUROCRYPT 2003, Springer-Verlag, LNCS 2656, pp. 630-648.
22. A. Kiayias, and Moti Yung. Group Signatures: Provable Security, Efficient Constructions and Anonymity from Trapdoor-Holders. Cryptology ePrint Archive: Report 2004/076.
23. J. Killian, and E. Petrank. Identity escrow. CRYPTO 1998, Springer-Verlag, LNCS 1642, pp. 169-185.
24. S. Kim, S. Park, and D. Won. Convertible group signatures. ASIACRYPT 1996, Springer-Verlag, LNCS 1163, pp. 311-321.
25. A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. SAC 1999, Springer-Verlag, LNCS 1758.
26. M. Michels. Comments on some group signature schemes. TR-96-3-D, Department of Computer Science, University of Technology, Chemnitz-Zwickau, Nov. 1996.
27. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. IEICE Trans. Vol. E85-A, No.2, pp.481-484, 2002.

28. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertexts Attacks. In Proc. of the 22nd STOC, pages 427-437. ACM Press. New York, 1990.
29. L. Nguyen. Accumulators from Bilinear Pairings and Applications. CT-RSA 2005, Springer-Verlag, LNCS, to appear.
30. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. Journal of Cryptology, 13(3):361396, March 2000.
31. P. Schnorr and M. Jakobsson. Security of signed El Gamal encryption. ASIACRYPT 2000, pages 73-89, LNCS 1976, 2000.
32. V. To, R. Safavi-Naini, and F. Zhang. New traitor tracing schemes using bilinear map. DRM Workshop 2003.
33. Y. Tsiounis and M. Yung. On the security of El Gamal based encryption. First International Workshop on Practice and Theory in Public Key Cryptography - PKC '98, pages 117-134, LNCS 1431, 1998.
34. F. Zhang, R. Safavi-Naini and W. Susilo. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. PKC 2004, Springer-Verlag, LNCS 2947, pp.277-290.

A Model

A.1 Oracles in the BSZ04 model

- $\mathsf{AddU}(\cdot)$: The add user oracle with argument $i \in \mathbb{N}$, an identity, allows the adversary to add i to the group as an honest user. The oracle adds i to the set HU of honest users, and picks a personal public and private key pair $(upk[i], usk[i])$ for i. It then executes the group-joining protocol by running Join (on behalf of i, initialized with $gpk, upk[i], usk[i]$) and Iss (on behalf of the issuer, initialized with $gpk, ik, i, upk[i]$). When Iss accepts, its final state is recorded as entry $reg[i]$ in the registration table. When Join accepts, its final state is recorded as the private signing key $gsk[i]$ of i. The calling adversary is returned $upk[i]$, but not the transcript of the interaction generated by the oracle.
- $\mathsf{CrptU}(\cdot, \cdot)$: The corrupt user oracle with arguments $i \in \mathbb{N}$, an identity, and a string upk, allows the adversary to corrupt user i and set its personal public key $upk[i]$ to the value upk chosen by the adversary. The oracle initializes the issuer's state in anticipation of a group-joining protocol with i.
- $\mathsf{SndToI}(\cdot, \cdot)$: Having corrupted user i, the adversary can use this send to issuer oracle to engage in a group-joining protocol with the honest issuer, itself playing the role of i and not necessarily executing the interactive algorithm Join prescribed for an honest user. The adversary provides the oracle with i and a message M_{in} to be sent to the issuer. The oracle, which maintains the issuer's state (the latter having been initialized by an earlier call to $\mathsf{CrptU}(i, \cdot)$), computes a response as per Iss, returns the outgoing message to the adversary, and sets entry $reg[i]$ of the registration table to Iss's final state if the latter accepts.
- $\mathsf{SndToU}(\cdot, \cdot)$: In some definitions we will want to consider an adversary that has corrupted the issuer. The send to user oracle can be used by such an

adversary to engage in a group-joining protocol with an honest user, itself playing the role of the issuer and not necessarily executing the interactive algorithm Iss prescribed for the honest issuer. The adversary provides the oracle with i and a message M_in to be sent to i. The oracle maintains the state of user i, initializing it the first time it is called by choosing a personal public and private key pair for i, computing a response as per Join, returning the outgoing message to the adversary, and setting the private signing of i to Join's final state if the latter accepts.

– $\mathsf{USK}(\cdot)$: The adversary can call this user secret keys oracle with argument the identity $i \in \mathbb{N}$ of a user to expose both the private signing key $gsk[i]$ and the personal private key $usk[i]$ of this user.
– $\mathsf{RReg}(\cdot)$: The adversary can read the contents of entry i of the registration table reg by calling the read registration table oracle with argument $i \in \mathbb{N}$.
– $\mathsf{WReg}(\cdot,\cdot)$: In some definitions we will allow the adversary to write/modify the entry i of the registration table reg by calling the write registration table oracle with argument $i \in \mathbb{N}$.
– $\mathsf{GSig}(\cdot,\cdot)$: This signing oracle enables the adversary to specify the identity i of a user and a message m, and obtain the signature of m under the private signing key $gsk[i]$ of i, as long as i is an honest user whose private signing key is defined.
– $\mathsf{Ch}(b,\cdot,\cdot,\cdot)$: A challenge oracle is provided to an adversary against anonymity, and depends on a challenge bit b set by the experiment. The adversary provides a pair i_0, i_1 of identities and a message m, and obtains the signature of m under the private signing key of i_b, as long as both i_0, i_1 are honest users with defined private signing keys. The oracle records the message-signature pair in GSet to ensure that the adversary does not later call the Open oracle on it.
– $\mathsf{Open}(\cdot,\cdot)$: The adversary can call this oracle with arguments a message m and signature ω to obtain the output of the opening algorithm on m, ω, computed under the opener's key ok, as long as ω was not previously returned in response to a query to $\mathsf{Ch}(b,\cdot,\cdot,\cdot)$.

A.2 Traceable Signatures

Kiayias et. al. [20] introduced Traceable Signature primitive, which is the Group Signature system with two added properties: (i) User Tracing means given a group member, all his signatures can be traced by a designated party, called *tracer*, without using the Open procedure; (ii) Signature Claiming means a given signature can be provably claimed by its signer. Compared with the traditional group signature mechanism, traceable signatures allow a variety of privacy levels for users. For example, tracing all signatures of a misbehaving user can be done without opening signatures and revealing identities of other users in the system. In [20], a traceable signature scheme was defined with a single group manager (GM) and a tuple (Setup, Join, Sign, Verify, Open, Reveal, Trace, Claim, Claim-Verify), where Setup, Join, Sign, Verify and Open are the same as GKg, (Join, Iss), GSig, GVf and Open, respectively, in a Group Signature scheme. Other procedures are:

- Reveal The GM runs this PPT algorithm that takes as input the Join transcript of a user i and outputs the tracing trapdoor $trace_i$ of that user.
- Trace The Tracer runs this deterministic polynomial time (DPT) algorithm that takes as input the group public key, a message-signature pair and the tracing trapdoor of a user, and checks if the signature was signed by the user.
- Claim Any user, who wants to claim a signature of a message, runs this PPT algorithm that takes the group public key, his private signing key and the message-signature pair and outputs a proof that he produced the signature.
- Claim-Verify A party can run this DPT algorithm that takes the group public key, a message-signature pair and a claim proof and checks if the proof holds.

Kiayias et al. [20] defined security of a traceable signature scheme in terms of providing Correctness, and also security against three types of attacks: Misidentification, Anonymity and Framing. Security against misidentification attacks is similar to the Traceability requirement in BSZ04 model, but it also requires that the adversary not be able to produce a signature that does not trace to any of the users controlled by the adversary. Security against anonymity attacks is similar to the Anonymity requirement in BSZ04 model, except that there is no Open oracle for the adversary. Security against framing attacks is similar to the Non-frameability requirement in BSZ04 model, but it requires that the adversary not be able to produce a signature that traces to an honest user, or claim a signature that was generated by an honest user.

B Preliminaries

B.1 Forking Lemma and Random Oracle model

The Forking Lemma, introduced by Pointcheval and Stern [30], is related to the *Random Oracle* model. The Random Oracle model assumes an oracle that generates a perfectly random value for each new query, and produces the same answer for two identical queries. We can assume the oracle maintains a query-answer table, which is initially empty. When receiving a query, the oracle first looks up the table to find if the query has been asked before. If it has, the oracle returns the answer in the table. Otherwise, it generates a new random value as the answer and appends the new query-answer pair in the table.

The Forking Lemma was originally stated in a specific model of signature schemes, where the signer needed to query the random oracle. Kiayias and Yung generalized the lemma so that it can be used for different primitives [22]. We present the General Forking Lemma as follows.

The General Forking Lemma *Consider a PPT $\mathcal{P}$, a PPT predicate $\mathcal{Q}$ and a random oracle $\mathcal{H}$ with output range $\{0,1\}^l$. The predicate $\mathcal{Q}$ satisfies the property $\mathcal{Q}(x) = 1 \Rightarrow (x = \langle \rho_1, c, \rho_2 \rangle) \wedge (c = \mathcal{H}(\rho_1))$. $\mathcal{R}$ is a process that given (t, c)*

appends or overwrites $\mathcal{H}$'s table so that $\mathcal{H}(t) = c$. $\mathcal{P}$ is allowed to ask queries on $\mathcal{H}$ and on $\mathcal{R}$. Moreover, it is assumed that $\mathcal{P}$ behaves in such a way so that queries (t, c) submitted by $\mathcal{P}$ to $\mathcal{R}$ adhere to the following conditions:

- *The component c is uniformly distributed over $\{0,1\}^l$.*
- *The component t follows a probability distribution so that the probability of occurrence of a specific t_0 is bounded by $2/2^l$.*

Assume now that $\mathcal{P}^{\mathcal{H},\mathcal{R}}(\mathsf{param})$ returns output x such that $\mathcal{Q}(x) = 1$ with non-negligible probability $\epsilon \geq 10(s+1)(s+q)/2^l$, where q is the number of $\mathcal{H}$-queries performed by $\mathcal{P}$, and s is the number of $\mathcal{R}$-queries. Then, there exists a PPT $\mathcal{P}'$ so that if $y \leftarrow \mathcal{P}'(\mathsf{param})$ it holds with probability 1/9 (i) $y = (\rho_1, c, \rho_2, c', \rho_2')$, (ii) $\mathcal{Q}(\langle\rho_1, c, \rho_2\rangle) = 1$, (iii) $\mathcal{Q}(\langle\rho_1, c', \rho_2'\rangle) = 1$, (iv) $c \neq c'$. 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.

B.2 Complexity Assumptions

For a function $f : \mathbb{N} \rightarrow \mathbb{R}^+$, if for every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$, it holds that $f(l) < l^{-\alpha}$, then f is said to be *negligible*.

The q-SDH assumption originates from a weaker assumption introduced by Mitsunari et. al. [27] to construct traitor tracing schemes [32] and later used by Zhang et al. [34] and Boneh et al. [6] to construct short signatures. It intuitively means that there is no PPT algorithm that can compute a pair $(c, \frac{1}{x+c}P)$, where $c \in \mathbb{Z}_p$, from a tuple $(P, xP, \ldots, x^q P)$, where $x \in_R \mathbb{Z}_p^*$.

q-Strong Diffie-Hellman (q-SDH) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{\textsf{q-SDH}}(l)$ is negligible.*

$$Adv_{\mathcal{A}}^{\textsf{q-SDH}}(l) = \mathsf{Pr}[(\mathcal{A}(\mathbf{t}, P, xP, \ldots, x^q P) = (c, \frac{1}{x+c}P)) \wedge (c \in \mathbb{Z}_p)]$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$ and $x \leftarrow \mathbb{Z}_p^*$.

Intuitively, the DBDH assumption [7] states that there is no PPT algorithm that can distinguish between a tuple $(aP, bP, cP, e(P,P)^{abc})$ and a tuple (aP, bP, cP, Γ), where $\Gamma \in_R \mathbb{G}_M^*$ (i.e., chosen uniformly random from $\mathbb{G}_M^*$) and $a, b, c \in_R \mathbb{Z}_p^*$. It is defined as follows.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{\textsf{DBDH}}(l)$ is negligible.*

$$\begin{aligned} Adv_{\mathcal{A}}^{\textsf{DBDH}}(l) = |&\mathsf{Pr}[\mathcal{A}(\mathbf{t}, aP, bP, cP, e(P,P)^{abc}) = 1] - \\ &\mathsf{Pr}[\mathcal{A}(\mathbf{t}, aP, bP, cP, \Gamma) = 1]| \end{aligned}$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$, $\Gamma \leftarrow \mathbb{G}_M^*$ and $a, b, c \leftarrow \mathbb{Z}_p^*$.

It is easy to see that if the q-SDH assumption or the DBDH assumption holds, then DL assumption (see Appendix B) holds. We also present a Decisional Diffie-Hellman Variant assumption and show that it is weaker than DBDH assumption

in Theorem 10. This assumption is very similar to the DDH assumption, but it works over groups $\mathbb{G}_1$ and $\mathbb{G}_M$.
Decisional Diffie-Hellman Variant (DDHV) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{\mathsf{DDHV}}(l)$ is negligible.*

$$Adv_{\mathcal{A}}^{\mathsf{DDHV}}(l) = |Pr[\mathcal{A}(\mathbf{t}, P, rP, e(P,P)^x, e(P,P)^{xr}) = 1] - Pr[\mathcal{A}(\mathbf{t}, P, rP, e(P,P)^x, e(P,P)^s) = 1]|$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$ *and* $x, r, s \leftarrow \mathbb{Z}_p^*$.

Theorem 10. *If the DBDH assumption holds then the DDHV assumption also holds.*

Proof. To prove the theorem, we show that if a PPT algorithm $\mathcal{A}$ has non-negligible $Adv_{\mathcal{A}}^{\mathsf{DDHV}}(l)$ (i.e., DDHV assumption does not hold), then we can build an algorithm $\mathcal{B}$ that has non-negligible $Adv_{\mathcal{B}}^{\mathsf{DBDH}}(l)$ (i.e., DBDH assumption does not hold). Suppose $a, b, c \in \mathbb{Z}_p^*$ and $\Gamma \in \mathbb{G}_M^*$, we observe that if a and b are uniformly distributed in $\mathbb{Z}_p^*$, then $x = ab$ is also uniformly distributed in $\mathbb{Z}_p^*$ and if Γ is uniformly distributed in $\mathbb{G}_M^*$, then s is also uniformly distributed in $\mathbb{Z}_p^*$, where $\Gamma = e(P,P)^s$. So to distinguish between $(aP, bP, cP, e(P,P)^{abc})$ and (aP, bP, cP, Γ), the algorithm $\mathcal{B}$ can simply return the outputs by $\mathcal{A}$ when it takes as input $(\mathbf{t}, P, cP, e(aP, bP), e(P,P)^{(ab)c})$ or $(\mathbf{t}, P, cP, e(aP, bP), \Gamma)$.

The Discrete Logarithm assumption in the group $\mathbb{G}_1$ is as follows.
Discrete Logarithm (DL) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{\mathsf{DL}}(l)$ is negligible.*

$$Adv_{\mathcal{A}}^{\mathsf{DL}}(l) = Pr[\mathcal{A}(\mathbf{t}, Q, xQ) = x]$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$, $Q \leftarrow \mathbb{G}_1^*$ *and* $x \leftarrow \mathbb{Z}_p^*$.

We now present the Decisional Diffie-Hellman assumption in the group $\mathbb{G}_M$. It can also be stated in many other cyclic groups of prime order, such as the subgroup of order p of group $\mathbb{Z}_{p'}$, where p and p' are large primes and $p \mid p' - 1$.
Decisional Diffie-Hellman (DDH) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{\mathsf{DDH}}(l)$ is negligible.*

$$Adv_{\mathcal{A}}^{\mathsf{DDH}}(l) = |Pr[\mathcal{A}(\mathbf{t}, \Gamma, \Gamma^r, \Gamma^x, \Gamma^{xr}) = 1] - Pr[\mathcal{A}(\mathbf{t}, \Gamma, \Gamma^r, \Gamma^x, \Gamma^s) = 1]|$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$, $\Gamma \leftarrow \mathbb{G}_M^*$ *and* $x, r, s \leftarrow \mathbb{Z}_p^*$.

C Security Proofs for the Group Signature schemes $\mathcal{GS}1$ and $\mathcal{GS}2$

Before proving security of $\mathcal{GS}1$ and $\mathcal{GS}2$, we prove the Zero-knowledge property of the Signing protocol in GSig algorithm and the Coalition-Resistance of $\mathcal{GS}1$

and $\mathcal{GS}2$. In our definition, Coalition-Resistance intuitively means that a colluding group of signers, with the knowledge of the opening key and access to some oracles, should not be able to generate a new valid user private signing key. For a group signature scheme $\mathcal{GS}$, a PPT adversary $\mathcal{A}$, a PPT predicate $\mathcal{U}$ that can determine the validity of a user private signing key, and any security parameter $l \in \mathbb{N}$, the formula of the experiment for Coalition-Resistance is as follows.

Experiment $Exp^{\mathsf{coal.re}}_{\mathcal{GS},\mathcal{A},\mathcal{U}}(l)$
$(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^l)$; $\mathsf{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$
$gsk' \leftarrow \mathcal{A}(gpk, ok : \mathsf{CrptU}(\cdot,\cdot), \mathsf{SndToI}(\cdot,\cdot), \mathsf{AddU}(\cdot), \mathsf{RReg}(\cdot), \mathsf{USK}(\cdot))$
If $gsk' \in \{gsk[i] | \; i \in \mathsf{CU} \cup \mathsf{HU}\}$ then return 0 else return $\mathcal{U}(gpk, gsk')$

The group signature scheme $\mathcal{GS}$ provides Coalition-Resistance if the following function $Adv^{\mathsf{coal.re}}_{\mathcal{GS},A,\mathcal{U}}(l)$ is negligible.

$$Adv^{\mathsf{coal.re}}_{\mathcal{GS},A,\mathcal{U}}(l) = \mathsf{Pr}[Exp^{\mathsf{coal.re}}_{\mathcal{GS},A,\mathcal{U}}(l) = 1]$$

Lemma 1. *Under the Discrete Log assumption on $\mathbb{G}_1$, the interactive Signing protocol underlying the* GSig *algorithm is a (honest-verifier) perfect zero-knowledge proof of knowledge of (a_i, S_i), x_i and t such that $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$, $E_a = tG$ and $\Lambda_a = e(P, S_i)\Theta_a^t$.*

Proof. The proof for completeness is straightforward. The proofs of Soundness and Zero-knowledge property are as follows.
Soundness: If the protocol accepts with non-negligible probability, we show that under the Discrete Log assumption on $\mathbb{G}_1$, a PPT prover must have the knowledge of (a_i, S_i), x_i and t satisfying the relations stated in the theorem. Suppose the protocol accepts for the same commitment $(V, R, T_1, T_2, T_3, \Pi_1, \Pi_2)$, two different pairs of challenges and responses $(c, s_0, ...s_6)$ and $(c', s'_0, ..., s'_6)$. Let $f_i = \frac{s_i - s'_i}{c - c'}$, $i = 0, ..., 6$, then $R = f_1G + f_2H$; $f_5R = f_3G + f_4H$; $E_a = f_6G$; $e(P_{pub}, V)e(P, P_0)^{-1} = e(P, P)^{f_0}e(P, V)^{-f_5}e(P, H)^{f_3}e(P_{pub}, H)^{f_1}$; $\Lambda_a e(P, V)^{-1} = e(P, H)^{-f_1}\Theta_a^{f_6}$.

From the first two equations, the prover has $\mathcal{O} = (f_3 - f_5f_1)G + (f_4 - f_5f_2)H$ ($\mathcal{O}$ is the identity element of $\mathbb{G}_1$). Under the Discrete Log assumption on $\mathbb{G}_1$, it implies that $f_3 = f_5f_1$.

Let $a_i = f_5$, $S_i = V - f_1H$, $x_i = f_0$, $t = f_6$, then $E_a = tG$, $\Lambda_a = e(P, S_i)\Theta_a^t$ and $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$. So the prover has the knowledge of (a_i, S_i), x_i and t satisfying the relations.
Zero-knowledge: The simulator chooses $c, s_0, ...s_6 \in_R \mathbb{Z}_p$, $V, R \in_R \mathbb{G}_1$ and compute $T_1 = s_1G + s_2H - cR$; $T_2 = s_3G + s_4H - s_5R$; $T_3 = s_6G - cE_a$; $\Pi_1 = e(P, P)^{s_0}e(P, V)^{-s_5}e(P, H)^{s_3}e(P_{pub}, H)^{s_1}e(P, P_0)^c e(P_{pub}, V)^{-c}$; $\Pi_2 = e(P, H)^{-s_1}\Theta_a^{s_6}\Lambda_a^{-c}e(P, V)^c$. We can see that the distribution of the simulation is the same as the distribution of the real transcript.

Lemma 2. *If the q-SDH assumption holds, then the group signature schemes $\mathcal{GS}1$ and $\mathcal{GS}2$, whose group sizes are bounded by q, provide Coalition-Resistance, where the predicate $\mathcal{U}$ is defined as:*
$\mathcal{U}(\langle P, P_0, P_{pub}, ...\rangle, \langle x_i, a_i, S_i, \Delta_i\rangle) = 1 \Leftrightarrow e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$.

Proof. We prove the lemma for both $\mathcal{GS}1$ and $\mathcal{GS}2$. Suppose there is a PPT adversary $\mathcal{A}$ that can break the Coalition-Resistance property of $\mathcal{GS}1$ or $\mathcal{GS}2$ with respect to the predicate $\mathcal{U}$ defined above. Let the set of private signing keys generated during $\mathcal{A}$'s attack be $\{(x_i, a_i, S_i, \Delta_i)\}_{i=1}^q$ and let his output be a new private signing key $(x^*, a^*, S^*, \Delta^*)$ with non-negligible probability (that means $(a^*, S^*) \notin \{(a_i, S_i)\}_{i=1}^q$). We show a construction of a PPT adversary $\mathcal{B}$ that can break the q-SDH assumption. Suppose a tuple *challenge* $= (Q, zQ, \ldots, z^q Q)$ is given, where $z \in_R \mathbb{Z}_p^*$; we show that $\mathcal{B}$ can compute $(c, 1/(z+c)Q)$, where $c \in \mathbb{Z}_p$ with non-negligible probability. We consider two cases.

Case 1: This is a trivial case, where $\mathcal{A}$ outputs $S^* \in \{S_1, ..., S_q\}$ with non-negligible probability. In this case, $\mathcal{B}$ chooses $x, x'_a, x'_b \in_R \mathbb{Z}_p^*$ and $G, H \in_R \mathbb{G}_1$, gives $\mathcal{A}$ the group signature public key $(P = Q, P_0 = zQ, P_{pub} = xP, H, G, \Theta_a = e(G,G)^{x'_a}, \Theta_b = e(G,G)^{x'_b})$ and the opening key (x'_a, x'_b) (no x'_b, Θ'_b in case of $\mathcal{GS}2$), and simulates a set of possible users. Then $\mathcal{B}$ can simulate all oracles that $\mathcal{A}$ needs to access. Suppose a set of private signing keys $\{(x_i, a_i, S_i, \Delta_i)\}_{i=1}^q$ is generated and $\mathcal{A}$ outputs a new $(x^*, a^*, S^*, \Delta^*)$ with non-negligible probability such that $S^* \in \{S_1, ..., S_q\}$. Suppose $S^* = S_j$, where $j \in \{1, ..., q\}$, then $\frac{1}{a^*+x}(x^*P + P_0) = \frac{1}{a_j+x}(x_j P + P_0)$, so $(a_j - a^*)P_0 = (a^* x_j - a_j x^* + x_j x - x^* x)P$. Therefore, z is computable by $\mathcal{B}$ from this, and so is $(c, 1/(z+c)Q)$, for any $c \in \mathbb{Z}_p$.

Case 2: This is when the first case does not hold. That means $\mathcal{A}$ outputs $S^* \notin \{S_1, ..., S_q\}$ with non-negligible probability. Then $\mathcal{B}$ plays the following game:

1. Generate $\alpha, a_i, x_i \in_R \mathbb{Z}_p^*$, $i = 1, ..., q$, where a_is are different from one another, then choose $m \in_R \{1, ..., q\}$.
2. Let $x = z - a_m$ ($\mathcal{B}$ does not know x), then the following P, P_{pub}, P_0 are computable by $\mathcal{B}$ from the tuple *challenge*.

$$P = \prod_{i=1, i\neq m}^{q} (z + a_i - a_m)Q$$

$$P_{pub} = xP = (z - a_m) \prod_{i=1, i\neq m}^{q} (z + a_i - a_m)Q$$

$$P_0 = \alpha \prod_{i=1}^{q} (z + a_i - a_m)Q - x_m \prod_{i=1, i\neq m}^{q} (z + a_i - a_m)Q$$

3. Generate $x'_a, x'_b \in_R \mathbb{Z}_p^*$ and $G, H \in_R \mathbb{G}_1$ and give $\mathcal{A}$ the group signature public key $(P, P_0, P_{pub}, H, G, \Theta_a = e(G,G)^{x'_a}, \Theta_b = e(G,G)^{x'_b})$ and the opening key (x'_a, x'_b) (no x'_b, Θ'_b in case of $\mathcal{GS}2$) and simulates a set of possible users.
4. With the capabilities above, $\mathcal{B}$ can simulate oracles $\mathsf{CrptU}(\cdot,\cdot)$, $\mathsf{RReg}(\cdot)$ and $\mathsf{USK}(\cdot)$) that $\mathcal{A}$ needs to access. For $\mathsf{AddU}(\cdot)$ or $\mathsf{SndToI}(\cdot,\cdot)$, $\mathcal{B}$ simulates the addition of an honest or corrupted user i as follows. As playing both sides of the Join, Iss protocol or being able to extract information from $\mathcal{A}$, $\mathcal{B}$ simulates the protocol as specified so that the prepared a_i, x_i above are computed in the protocol to be the corresponding parts of the user i's private signing key. $\mathcal{B}$ can compute S_i as follows:

– If $i = m$, then

$$S_m = \frac{1}{a_m + x}(x_m P + P_0) = \alpha \prod_{i=1, i \neq m}^{q} (z + a_i - a_m) Q$$

This is computable from the tuple *challenge*.

– If $i \neq m$, then

$$S_i = \frac{1}{a_i + x}(x_i P + P_0) = (x_i - x_m) \prod_{j=1, j \neq m, i}^{q} (z + a_j - a_m) Q + \alpha \prod_{j=1, j \neq i}^{q} (z + a_j - a_m) Q$$

This is computable from the tuple *challenge*.

5. Get the output $(x^*, a^*, S^*, \Delta^*)$ from $\mathcal{A}$, where

$$S^* = \frac{1}{a^* + x}(x^* P + P_0) = \frac{1}{z + a^* - a_m}(\alpha z + x^* - x_m) \prod_{i=1, i \neq m}^{q} (z + a_i - a_m) Q$$

We can see that the case $\alpha z + x^* - x_m = \alpha(z + a^* - a_m)$ happens with negligible probability, as it results in $S^* = S_m$. So the case $\alpha z + x^* - x_m \neq \alpha(z + a^* - a_m)$ happens with non-negligible probability ϵ_1. Suppose in this case, the probability that $a^* \in \{a_1, ..., a_q\}$ is ϵ_2. Then the probability that $a^* \notin \{a_1, ..., a_q\} \backslash \{a_m\}$ is $\epsilon_1 - \frac{q-1}{q}\epsilon_2$ (as $m \in_R \{1, ..., q\}$), which is also non-negligible if q is bound by a polynomial of l. If $\alpha z + x^* - x_m \neq \alpha(z + a^* - a_m)$ and $a^* \notin \{a_1, ..., a_q\} \backslash \{a_m\}$, then $\frac{1}{z + a^* - a_m} Q$ is computable from the tuple *challenge* and S^* and so $\mathcal{B}$ can compute $(c, \frac{1}{z+c} Q)$, where $c = a^* - a_m$.

C.1 Proof of Theorem 4 and Theorem 7-Weak Anonymity

We prove Anonymity of $\mathcal{GS}1$ and Weak Anonymity of $\mathcal{GS}2$ at the same time. Suppose there is a PPT adversary $\mathcal{A}$ that can break Anonymity property of $\mathcal{GS}1$ (or Weak Anonymity of $\mathcal{GS}2$). We show a construction of a PPT adversary $\mathcal{B}$ that can break IND-CCA property of El Gamal^{BP2} (or IND-CPA property of El Gamal^{BP1}). Suppose an El Gamal^{BP2} public key (G, Θ_a, Θ_b) and a Decryption oracle (or only an El Gamal^{BP1} public key (G, Θ_a)) are given, $\mathcal{B}$ constructs an instance of $\mathcal{GS}1$ (or $\mathcal{GS}2$) by generating the issuing key $ik = x$ and the group public key $gpk = (P, P_0, P_{pub}, H, G, \Theta_a, \Theta_b)$ (no Θ_b for $\mathcal{GS}2$ case). The opening key ok is the private key of the El Gamal^{BP2} (or El Gamal^{BP1}) public key, and is unknown to $\mathcal{B}$. In GSig, we assume the signer, instead of using the hash function $\mathcal{H}$, queries a random oracle, whose query-answer table can be appended by $\mathcal{B}$. Let $\mathcal{B}$ play the role of the issuer, simulating the set of possible users and providing $\mathcal{A}$ with gpk, ik and the following simulated oracles:

– $\mathsf{SndToI}(\cdot,\cdot)$, $\mathsf{SndToU}(\cdot,\cdot)$, $\mathsf{WReg}(\cdot,\cdot)$, $\mathsf{USK}(\cdot)$ and $\mathsf{CrptU}(\cdot,\cdot)$. With the above capabilities, $\mathcal{B}$ can easily simulate these oracles.
– $\mathsf{Ch}(d,\cdot,\cdot,\cdot)$. When receiving a query (i_0, i_1, m) from $\mathcal{A}$, $\mathcal{B}$ finds Δ_{i_d} and asks for an El Gamal^{BP2} challenge encryption $cip = (E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ (or an El Gamal^{BP1} challenge encryption $cip = (E_a, \Lambda_a)$) of Δ_{i_d}. From that, $\mathcal{B}$ simulates $c, s_0, ...s_6$, V, R, $T_1, T_2, T_3, \Pi_1, \Pi_2$ as in the $\mathsf{Zero\text{-}knowledge}$ proof of Lemma 1, such that the value $que = (P||P_0||P_{pub}||H||G||\Theta_a||\Theta_b||E_a||\Lambda_a||E_b|| \Lambda_b||\varsigma||V||R||T_1||T_2||T_3||\Pi_1||\Pi_2||m)$ has not been queried to the random oracle. Then $\mathcal{B}$ appends (que, c) to the random oracle's table and returns to $\mathcal{A}$ the challenge signature $(c, s_0, ..., s_6, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ (no $\Theta_b, E_b, \Lambda_b, \varsigma$ in $\mathcal{GS}2$ case).
– $\mathsf{Open}(\cdot,\cdot)$. In $\mathcal{GS}2$ case, this oracle is not accessible by the adversary. In $\mathcal{GS}1$ case, when receiving a query (m, ω) from $\mathcal{A}$, $\mathcal{B}$ answers $\mathcal{A}$ by extracting the El Gamal^{BP2} ciphertext part from ω, sending that ciphertext to the Decryption oracle and from that finding an answer to $\mathcal{A}$. We will later discuss the case when the extracted El Gamal^{BP2} ciphertext is the same as the challenge ciphertext cip with non-negligible probability.

At last, $\mathcal{B}$ outputs the bit returned by $\mathcal{A}$. As $\mathcal{A}$ can break Anonymity property (or Weak Anonymity for $\mathcal{GS}2$), $\mathcal{B}$ outputs the correct b with non-negligible probability.

Now we discuss the case that for querying Open, $\mathcal{A}$ manages to find a signature $(\bar{c}, \bar{s_0}, ..., \bar{s_6}, \bar{V}, \bar{R}, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$, whose El Gamal^{BP2} ciphertext part is the same as the challenge ciphertext cip with non-negligible probability (the partly omitted part of the signature is $\rho_1 = (gpk, m, T_1, T_2, T_3, \Pi_1, \Pi_2, \bar{V}, \bar{R}, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$). We observe that the General Forking Lemma is applicable to $\mathcal{A}$ where GVf plays as the predicate $\mathcal{Q}$. So, by applying the General Forking Lemma, $\mathcal{B}$ can obtain for the same ρ_1 two valid signatures $(\bar{c}, \bar{s_0}, ..., \bar{s_6}, \bar{V}, \bar{R}, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ and $(\bar{c}', \bar{s_0}', ..., \bar{s_6}', \bar{V}, \bar{R}, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ with $\bar{c} \neq \bar{c}'$. Following the same arguments as in the $\mathsf{Soundness}$ proof of Lemma 1, $\mathcal{B}$ can find t, Δ_i so that $E_a = tG$ and $\Lambda_a = \Delta_i \Theta_a^t$ and, thereby, output the correct b with non-negligible probability.

C.2 Proof of Theorem 5 and Theorem 7-Traceability

We prove Traceability of $\mathcal{GS}1$ and $\mathcal{GS}2$ at the same time. Suppose there is a PPT adversary $\mathcal{A}$ that can break Traceability property of $\mathcal{GS}1$ (or $\mathcal{GS}2$). We show that there exists a PPT adversary $\mathcal{B}$ that can break Coalition-Resistance of $\mathcal{GS}1$ (or $\mathcal{GS}2$). Suppose $\mathcal{A}$ can output a valid message-signature pair (m, ω) so that the opener can not trace the identity of the signer, or the opener can find the identity but can not prove that to the Judge. By applying the General Forking Lemma to $\mathcal{A}$ where GVf plays as the predicate $\mathcal{Q}$, there is a PPT adversary $\mathcal{A}'$ that can output two valid signatures $\omega = (c, s_0, ..., s_6, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ and $(c', s_0', ..., s_6', V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ with $c \neq c'$ for the same $\rho_1 = (gpk, m, T_1, T_2, T_3, \Pi_1, \Pi_2, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ (no $\Theta_b, E_b, \Lambda_b, \varsigma$ in $\mathcal{GS}2$ case). Following the same arguments as in the $\mathsf{Soundness}$ proof of Lemma 1, $\mathcal{B}$ can find a_i, S_i, x_i and t such that $E_a = tG$, $\Lambda_a = e(P, S_i)\Theta_a^t$ and $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$. So the opener, which is assumed to operate accurately, should find $\Delta_i = e(P, S_i)$ from the signature. The issuer is assumed to be uncorrupted and no oracle accessible by the adversaries can write on *reg* table or overwrite $upk[j]$ of a group member j (CrptU does not apply to group members). So if Δ_i can not be found on *reg*, $\mathcal{B}$ has produced a new valid user private signing key $(x_i, a_i, S_i, \Delta_i)$.

C.3 Proof of Theorem 6 and Theorem 7-Non-frameability

We prove Non-frameability of $\mathcal{GS}1$ and $\mathcal{GS}2$ at the same time. Suppose there is a PPT adversary $\mathcal{A}$ that can break Non-frameability property of $\mathcal{GS}1$ (or $\mathcal{GS}2$), we show that there exists a PPT adversary $\mathcal{B}$ that can break Discrete Logarithm Assumption over $\mathbb{G}_1$. Suppose that $\mathcal{B}$ is given a challenge $(P, P^* = zP)$, where $P \leftarrow \mathbb{G}_1^*$ and $z \leftarrow \mathbb{Z}_p^*$, and $\mathcal{B}$ needs to compute z. $\mathcal{B}$ constructs an instance of $\mathcal{GS}1$ (or $\mathcal{GS}2$) by generating $x, x'_a, x'_b, d \in_R \mathbb{Z}_p^*$ and $G, H \in_R \mathbb{G}_1$ and give $\mathcal{A}$ the group signature public key $(P, P_0 = dP, P_{pub} = xP, H, G, \Theta_a = e(G, G)^{x'_a}, \Theta_b = e(G, G)^{x'_b})$, the issuing key $ik = x$ and the opening key (x'_a, x'_b) (no x'_b, Θ'_b in case of $\mathcal{GS}2$). $\mathcal{B}$ simulates a set of possible users $\{1, ..., q\}$, where q is the upper bound of the group size, chooses $i_0 \in_R \{1, ..., q\}$ and provides $\mathcal{A}$ access to the following simulated oracles:

- SndToU(i, M_{in}). If $i \neq i_0$, $\mathcal{B}$ just plays as a honest user i and executes Iss as specified in M_{in}. If $i = i_0$, $\mathcal{B}$ simulates the Join, Iss protocol so that $P_{i_0} = P^*$ (by controlling the random oracle, $\mathcal{B}$ can simulate the proof of knowledge in the protocol). Suppose the private signing key obtained for i_0 is $(x_{i_0}, a_{i_0}, S_{i_0}, \Delta_{i_0})$, where $x_{i_0} = z$ is unknown to $\mathcal{B}$.
- WReg$(\cdot,\cdot)$, GSig$(\cdot,\cdot)$, USK$(\cdot)$ and CrptU$(\cdot,\cdot)$. With the capabilities above, $\mathcal{B}$ can simulate all these oracles, except the case when he gets a query USK(i_0). In this case, $\mathcal{B}$ fails.

If $\mathcal{A}$ succeeds with probability ϵ, then the probability that he can output a valid message-signature pair (m, ω) of i_0 is at least ϵ/q, as $i_0 \in_R \{1, ..., q\}$. By applying the General Forking Lemma to $\mathcal{A}$ where GVf plays as the predicate $\mathcal{Q}$, there is a PPT adversary $\mathcal{A}'$ that can output two valid signatures $\omega = (c, s_0, ..., s_6, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ and $(c', s'_0, ..., s'_6, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ with $c \neq c'$ for the same $\rho_1 = (gpk, m, T_1, T_2, T_3, \Pi_1, \Pi_2, V, R, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ (no $\Theta_b, E_b, \Lambda_b, \varsigma$ in $\mathcal{GS}2$ case). Following the same arguments as in the Soundness proof of Lemma 1, $\mathcal{B}$ can find a_{i_1}, S_{i_1}, x_{i_1} and t such that $E_a = tG$, $\Lambda_a = e(P, S_{i_1})\Theta_a^t$ and $e(a_{i_1}P + P_{pub}, S_{i_1}) = e(P, x_{i_1}P + P_0)$. The digital signature scheme (K_S, $Sign$, Ver) is UNF-CMA and so $e(P, S_{i_0}) = e(P, S_{i_1})$ or $S_{i_0} = S_{i_1}$. So $\frac{1}{a_{i_0}+x}(x_{i_0}P + dP) = \frac{1}{a_{i_1}+x}(x_{i_1}P + dP)$, from that, $\mathcal{B}$ can compute $z = x_{i_0}$.