Accumulating Composites and Improved Group Signing

Gene Tsudik and Shouhuai Xu

Department of Information and Computer Science
University of California at Irvine
{gts,shxu}@ics.uci.edu

Abstract. Constructing practical and provably secure group signature schemes has been a very active research topic in recent years. A group signature can be viewed as a digital signature with certain extra properties. Notably, anyone can verify that a signature is generated by a legitimate group member, while the actual signer can only be identified (and linked) by a designated entity called a group manager. Currently, the most efficient group signature scheme available is due to Camenisch and Lysyanskaya [CL02]. It is obtained by integrating a novel dynamic accumulator with the scheme by Ateniese, et al. [ACJT00].

In this paper, we construct a dynamic accumulator that accumulates *composites*, as opposed to previous accumulators that accumulated *primes*. We also present an efficient method for proving knowledge of factorization of a committed value. Based on these (and other) techniques we design a novel provably secure group signature scheme. It operates in the *common auxiliary string* model and offers two important benefits: 1) the Join process is very efficient: a new member computes only a single exponentiation, and 2) the (unoptimized) cost of generating a group signature is 17 exponentiations which is appreciably less than the state-of-the-art.

1 Introduction

The notion of group signatures was introduced by Chaum and van Heyst in 1991 [CvH91]. Since then, seeking practical and provably secure group signature schemes – and their interactive dual known as identity escrow [KP98] – has been a very active research area in applied cryptography. A group signature can be seen as a normal digital signature with the following extra properties: anyone can verify that a signature is generated by a legitimate group member, while the actual signer can only be identified and linked by a designated trusted entity called a group manager. It is important to point out that the mechanism, which allows the group manager to identify the actual signer of a group signature, acts as a crucial deterrent that prevents group members from misbehaving.

The basic idea underlying most group signature schemes (as well as ours) is the following: In order for a group member (Alice) to sign a message, she needs to construct an *authorization-proof* to show that she has a legitimate membership certificate, and an *ownership-proof* to demonstrate knowledge of the secret corresponding to the membership certificate. The issues in these two proofs are similar to those encountered in a normal public key infrastructure (PKI) setting, namely, a signature can be verified using the alleged signer's public key contained in a certificate which has not been revoked. However, the group signature scenario is more complicated, since a signer cannot show her membership certificate without compromising her anonymity. It is precisely this anonymity requirement that makes it very difficult to have a practical solution that facilitates revocation of membership certificates (a concept compatible to certificate revocation in a normal PKI), or the validity check of non-revoked membership certificates.

Early group signature schemes (e.g., [CP94]) have the characteristics that the sizes of the group public key and/or of group signatures *linearly* depend on the number of group members. The advantages of these schemes include: (1) many of the schemes have been proven secure using some standard cryptographic assumptions (such as the hardness of computing discrete logarithms), and (2) *authorization-proof* is trivial since revoking a member is done by the group manager by removing the corresponding membership certificate from the group public key. The disadvantage of such schemes is that the complexity of *ownership-proof*, namely proving and verifying that one knows the secret corresponding to a (non-identified yet non-revoked) membership certificate, is linear in the number of current members and thus becomes inefficient for large groups.

To combat linear complexity incurred as part of *ownership-proof*, Camenisch and Stadler [CS97] took a different approach where the sizes of the group public key and of group signatures are constant and independent of the number of current group members. This approach has been adopted in some follow-on results, e.g., [CM98,CM99a,ACJT00].[1] As initially presented, these schemes only support adding new members. Since then, [CS97] and [ACJT00] have been extended to support membership revocation [BS01,S01,AST02]. However, revocation incurs certain significant costs due to some (or all) of the following:

- Group manager re-issuing all certificates for each revocation interval.
- Group member (signer) proving, as part of signing, that her certificate is not revoked.
- Verifier checking each group signature against the current list of revoked certificates.

As pointed out in [CL02], each of the above has a linear dependency either on the number of current, or the total number of deleted, members.

State-of-the-Art. Currently, the most efficient group signature scheme is due to Camenisch and Lysyanskaya [CL02]. It is constructed by incorporating a *dynamic accumulator*, which allows efficient *authorization-proofs*, into the group signature scheme due to Ateniese, et al. [ACJT00], which allows efficient *ownership-proofs*.[2] The concept of dynamic accumulators introduced in [CL02] is a variant of the accumulator due to Baric and Pfitzmann [BP97]. It enables a group member to conduct a light-weight authorization-proof such that both the proving and verifying complexities are independent of the number of the current, or total deleted, members. We note that the use of dynamic accumulators to facilitate *authorization-proofs*, requires the group manager to disseminate certain information, such as the values deleted from the accumulator whenever a member (or a set of thereof) joins or leaves the group.

1.1 Contributions

The main contribution of this paper is a new group signature scheme provably secure against adaptive adversaries, i.e., adversaries allowed to adaptively join and leave the group. The scheme is obtained by incorporating several building blocks, some of which are new (e.g., the dynamic composites accumulator), while others are more efficient than previous techniques providing the same functionality (e.g., the multiplication protocol that allows one to prove that she knows the factorization of a committed value). More specifically:

- A new dynamic accumulator that accumulates *composites* (see Section 5.1), as opposed to the prior construct that accumulates *primes* [CL02]. This accumulator fits well into a group signature scheme because it allows us to conduct simultaneous *authorization-proofs* and *ownership-proofs* based on the factorizations of accumulated *composites*.
- A protocol (in Section 5.2) for proving knowledge of factorization of a committed value, which, in our case, corresponds to an accumulated composite. This protocol is more efficient than prior art, such as [DF02].
- A protocol (in Section 5.3) for verifiable encryption of discrete logarithms, based on the public key cryptosystem due to Catalano, et al. [CGHN01]. This protocol is more efficient than a similar protocol (e.g., the one presented in [MR01]) based on the Pallier cryptosystem [P99].

As mentioned earlier, the state-of-the-art group signature scheme by Camenisch and Lysyanskaya is obtained by integrating a dynamic prime accumulator [CL02] with the *bare* group signature scheme in [ACJT00]. This integration was needed since a prime accumulator cannot be used for *ownership-proof*. (See Section 3 for further discussion.) In comparison with the [CL02] scheme, our approach has three major benefits:

- Use of the new accumulator construct for both *ownership-proof* and *authorization-proof*. This yields a conceptually simpler scheme.

[1] An interactive variant, referred to as "Identity Escrow", was introduced by Kilian and Petrank [KP98].

[2] The credit for this scheme should be given to all authors of [ACJT00,CL02], and we refer to it as the Camenisch-Lysyanskaya scheme only for the purpose of simplicity.

– Efficient Join: a new member only computes a single exponentiation in order to verify that her *composite* has been correctly accumulated. In comparison, Join involves more than 30 exponentiations in [CL02]. We note that this complexity does not stem from the use of the dynamic accumulator; it is inherited from Join of [ACJT00].
– Efficient Sign: the computational complexity of signing is 17 exponentiations (without any optimizations) which is notably lower than 25 in the Camenisch-Lysyanskaya scheme.[3]

Our scheme also has some potential drawbacks. They are discussed in Section 7.

1.2 Organization

In Section 2, we overeview the model and goals of group signatures. Then, in Section 3, we introduce the basic ideas underlying our group signature scheme. Section 4 presents some cryptographic preliminaries and Section 5 describes some building blocks. The new group signature scheme is found in Section 6; its features and potential drawbacks are discussed in Section 7. Due to space limitations, technical details of the security proof are deferred to the appendix.

2 Model and Goals

Participants. A group signature scheme involves a *group manager* (responsible for admitting/deleting members and for revoking anonymity of group signatures, e.g., in cases of dispute or fraud), a set of *group members*, and a set of *signature verifiers*. All participants are modeled as probabilistic polynomial-time interactive Turing machines.

Communication Channels. All communication channels are assumed asynchronous. The communication channel between a signer and a receiver is assumed to be anonymous.

Trust. We assume that the *group manager* will not admit unauthorized individuals into the group. This is reasonable, since, otherwise, the group manager can issue valid membership certificates to rogue members and thus make the group signature scheme useless. We assume that the group members, whether honest or not, behave *rationally*. More precisely, a dishonest group member may seek to undermine the system (e.g., by colluding with other internal or external parties) as long as the attack will not be traced back to herself. Nonetheless, she will not take the chance if she (or anyone else colluding with her) is bound to be caught. This assumption is also reasonable since, in any group signature scheme (indeed, in any cryptographic setting), a dishonest user could (for instance) simply give away her own secrets. However, she is bound to be held accountable for any consequences of such misbehavior.

2.1 Definitions

A group signature scheme consists of the following procedures:

– Setup. On input a security parameter, this probabilistic algorithm outputs the initial group public key and the secret key for the group manager.
– Join. This is a protocol executed between the group manager and a user who is to become a group member. The user's output is a membership certificate and a membership secret; the group manager's output is some updated information that indicates the current state of the system.
– Revoke. This is a deterministic algorithm which, on input a membership certificate, outputs some updated information that indicates the current state of the system after revoking the given membership certificate.
– Update. This is a deterministic algorithm that may be triggered by any Join or Revoke operation. It is run by the group members after obtaining certain information from the group manager.
– Sign. This is a probabilistic algorithm which, on input of: a group public key, a membership certificate, a membership secret and a message, outputs a group signature.

[3] However, the [ACJT00] scheme can be slightly improved to reduce (by 2) the number of exponentiations in Sign. Consequently, the cost of Sign in the Camenisch-Lysyanskaya scheme can be commensurately lowered.

– Verify. This is a deterministic algorithm for establishing the validity of an alleged group signature on a message with respect to the group public key.
– Open. This is an algorithm which, on input of: a message, a valid group signature, a group public key and a group manager's secret key, determines the identity of the actual signer.

2.2 The Goals

A secure group signature scheme must satisfy the following properties:

– Correctness. Any signatures produced by a group member using Sign must be accepted by Verify.
– Unforgeability. Only group members are able to sign messages on behalf of the group.
– Anonymity. Given a valid group signature, identifying the actual signer is computationally hard for everyone but the group manager.
– Unlinkability. Deciding whether two different group signatures were generated by the same member is computationally hard for everyone but the group manager.
– No-framing. No combination of a group manager and a subset of dishonest group members can sign on behalf of a single honest member. That is, no honest member can be made responsible for a signature she did not produce.
– Traceability. The group manager is always able to identify the actual signer of any valid group signature.
– Coalition-resistance. A colluding subset of group members (even all members) cannot generate a signature that the group manager cannot trace.

3 Basic Ideas

The basic idea underlying our group signature scheme is to utilize an accumulator that accumulates *composites*, where the factorization of a *composite* is only known to the user who generates it. More specifically, suppose a group member has a witness w such that $w^e = v \bmod n$ where v is the public accumulator value and n is the product of two safe primes, The factorization of $e = e_1 e_2$ (i.e., the primes e_1 and e_2) is only known to the member. This knowledge allows the user to conduct an *ownership-proof* by demonstrating that $e = e_1 e_2$. The witness w facilitates an *authorization-proof* that $w^e = v \bmod n$.

While the basic idea is quite simple, we must deal with potential abuses. We now present an informal discussion of some subtleties, and suggest countermeasures. Readers who prefer to commence with the more in-depth technical description may wish to skip this section.

Q: How to ensure anonymity while preserving authenticity?
A: A signer "encrypts" both w and e such that the required properties regarding them can be shown on the corresponding "ciphertexts". In particular, a signer needs to show $w^e = v$ for the *authorization-proof*, and $e = e_1 e_2$ for the *ownership-proof.* As long as e is chosen such that it is infeasible to factor, no group of participants (including the group manager) can frame an honest group member.
Q: How to deal with multiple dishonest group members who collude (by revealing to each other factorizations of their respective composites) and produce new membership certificates? For example, if Alice chooses $e_1 = e_{1,1}e_{1,2}$ and Bob chooses $e_2 = e_{2,1}e_{2,2}$, they can collude to obtain new membership certificates for the values such as $(e_1 e_{2,1})$ or $(e_{1,1}e_{2,1})$.
A: Although we cannot prevent such abuses, we can ensure that, the group manager can factor at least one of the colluding group member's e (e_1, or e_2, or even both) and thus identify at least one of the miscreants. One way to do this, as we shall see, is to use a public key encryption scheme (for which the group manager knows the private key) so that the signer is forced to encrypt an "accumulated" value she is claiming. Note that even a dishonest member cannot afford to encrypt $e_{1,1}$, since, otherwise, the group manager can factor her composite and forge signatures that will be traced back to the dishonest member.
Q: How to deal with multiple dishonest group members who collude (but *do not reveal* to each other the factorizations of their composites) and produce new membership certificates? For example, suppose that Alice holds (w_1, e_1) and Bob holds (w_2, e_2), where $e_1 = e_{1,1}e_{1,2}$, $e_2 = e_{2,1}e_{2,2}$, $w_1^{e_1} = w_2^{e_2} = v$. They can collude and generate $(w', e' = e_1 e_2)$ such that $(w')^{e_1 e_2} = v$.

A: We prevent such attacks by requiring all verifiers to check that e' falls within a certain range.

Q: Does the group manager need to check whether a composite presented by a new user during Join is well-formed, i.e., a product of two large primes? If not, what if a dishonest group member chooses e to be a single prime or a product of multiple (more than 2) primes?

A: We do not aim to prevent such abuses (this also justifies our efficiency gains). However, will be shown, no adversary can gain any benefit from any such abuse since the group manager is always able to identify at least one of the colluding group members.

Q: What if the group manager attempts to frame an honest group member by using the group member's membership certificate (w, e) where $w^e = v$ while providing a proof of factorization of some value $e' \neq e$.

A: The Sign process ensures that, if the group manager proves knowledge of the factorization of an "accumulated" value $e' \neq e$, then the witness value that the group manager (or any impersonator) is showing is $w' \neq w$. Moreover, the group manager is required to conduct a zero-knowledge proof as part of Open such that the decryption corresponding to an ElGamal ciphertext (of w) is correct.

Remark. The *ownership-proof* in our scheme is conducted by a signer to prove knowledge of the factorization of a unique accumulated composite $e = e_1 e_2$, where $w^e = v$ for a public accumulator value v. Therefore, we say that our ownership-proof is based on the factoring problem. Whereas, the ownership-proof in the Camenisch-Lysyanskaya scheme [CL02] is done by a signer to prove knowledge of x, such that $w^e = a^x b$, where a and b are public, and e is prime. Therefore, we say that the ownership-proof is based on the discrete logarithm problem.

4 Preliminaries

In this section we go over some basic cryptographic assumptions and definitions necessary for our scheme.

4.1 RSA and Related Settings

Definition 1. (safe RSA modulus). *We say $n = pq$ is a* safe RSA modulus*, if $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are all primes.*

By convention, let $gcd(0, n) = n$, and $\mathbb{QR}_n$ be the subgroup of quadratic residues modulo n.

The Strong RSA Assumption (SRSA). This assumption was independently introduced by Baric and Pfitzmann [BP97] and by Fujisaki and Okamoto [FO97].

Definition 2. (Strong RSA Problem). *Let $n = pq$ be a RSA-like modulus and $\mathbb{G}$ be a cyclic subgroup of $\mathbb{Z}_n^*$, where $|ord(\mathbb{G})| = l_{\mathbb{G}}$. Given n and $z \in_R \mathbb{G}$, the* Strong RSA Problem *consists of finding $w \in \mathbb{G}$ and $e > 1$ such that $z = w^e \mod n$.*

Assumption 1 (Strong RSA Assumption). *Suppose a RSA-like modulus n and $z \in_R \mathbb{G}$ are obtained according to a given security parameter $l_{\mathbb{G}}$. The assumption states that any probabilistic polynomial-time algorithm $\mathcal{A}$ can solve the* Strong RSA Problem *with only negligible probability.*

The following lemma is useful and has appeared in many places (e.g.,[GKR00]).

Lemma 1. *Suppose $n = pq$ is a safe RSA modulus. Given an element $w \in \mathbb{Z}_n^* \setminus \{1, -1\}$ of $ord(w) < p'q'$, either $gcd(w - 1, n)$ or $gcd(w + 1, n)$ is a prime factor of n.*

4.2 Discrete Logarithm Related Cryptographic Setting

The Decisional Diffie-Hellman Assumption (DDHA). This assumption was first explicitly introduced by Brands [B93], although it was implicitly assumed in earlier literature.

Definition 3. (Decisional Diffie-Hellman Problem). *Let $\mathbb{G} = \langle g \rangle$ be a cyclic group generated by g, where $|ord(\mathbb{G})| = l_{\mathbb{G}}$. Given g, g^x, g^y, and $g^z \in \mathbb{G}$, the* Decisional Diffie-Hellman Problem *consists of deciding whether $g^{xy} = g^z$.*

Assumption 2 (Decisional Diffie-Hellman Assumption). *Suppose a group* $\mathbb{G}$ *and an element* g *of order* $ord(\mathbb{G})$ *are obtained according to a given security parameter* $l_{\mathbb{G}}$. *The assumption states that there is no probabilistic polynomial-time algorithm that distinguishes with non-negligible probability* (g, g^x, g^y, g^{xy}) *from* (g, g^x, g^y, g^z), *where* $x, y, z \in_R \mathbb{Z}_{ord(\mathbb{G})}$.

We will utilize the ElGamal public key cryptosystem [E85] whose semantic security is based on DDHA [TY98]. Since we always work in the setting of modulo a safe RSA modulus, we need certain group in which the DDHA holds.

Fact 1 *If* n *is a safe RSA modulus, then* $\mathbb{QR}_n$ *is a cyclic subgroup of order* $p'q'$. *Moreover, if* $a \in \mathbb{Z}_n^*$ *and* $gcd(a \pm 1, n) = 1$, *then* $g = a^2 \bmod n$ *is of order* $p'q'$.

4.3 The CGHN Public Key Cryptosystem

We now briefly review Paillier's cryptosystem [P99]. Suppose $n = pq$ where p and q are large primes. Then we have Euler's Totient function $\phi(n) = (p-1)(q-1)$ and Carmichael's function $\lambda(n) = lcm(p-1, q-1)$. It follows that: $w^{\lambda(n)} = 1 \bmod n$ and $w^{n \cdot \lambda(n)} = 1 \bmod n^2$ for any $w \in \mathbb{Z}_{n^2}^*$. Let $(n, g; n, g, p, q)$ be a pair of Paillier public and private keys as specified in [P99]. To encrypt a message $m \in \mathbb{Z}_n$, one chooses $r \in_R \mathbb{Z}_n^*$ and computes the ciphertext $c = g^m r^n \bmod n^2$. Note that an interesting selection of g is $g = (1+n)$ because $(1+n)^m = 1 + mn \bmod n^2$.

A performance disadvantage of the Paillier cryptosystem is that one needs to compute $r^n \bmod n^2$. Catalano et al. [CGHN01] observed that if we always set $g = (1+n)$ then we can use any public exponent t as long as $gcd(t, \lambda(n^2)) = 1$, because a ciphertext $c = (1+mn)r^t \bmod n^2$ yields $c = r^t \bmod n$, thereby r can be recovered by a standard RSA decryption operation. This means that one only needs to compute an exponentiation operation modulo n^2 with respect to an exponent $|t| << |n|$. We call this variant the CGHN cryptosystem whose semantic security is based on the following DSRA assumption.

Definition 4. (Computational Small t-roots Problem). *This is a variant of the RSA problem in* $\mathbb{Z}_{n^2}^*$. *The problem is to invert* $y^t \bmod n^2$, *where* $y \in \mathbb{Z}_n$, $t \in \mathbb{Z}_n$, *and* $gcd(t, \lambda(n^2)) = 1$.

Definition 5. (Decisional Small Residuosity Problem, DSRP). *This is a decisional version of the above computational problem. Given an element* $x \in_R \mathbb{Z}_{n^2}^*$, *one needs to decide whether* x *is the form* y^t *with* $y \in \mathbb{Z}_n$.

Assumption 3 (Decisional Small Residuosity Assumption, DSRA) *Let* n *be a randomly chosen* l*-bit RSA modulus,* $t \in \mathbb{Z}_n$ *such that* $gcd(t, \lambda(n^2)) = 1$, *and* $x \in_R \mathbb{Z}_{n^2}^*$. *There exists no probabilistic polynomial-time algorithm that is able to decide, with non-negligible advantage, whether* x *is the form* y^t *with* $y \in \mathbb{Z}_n$.

The following lemma will be used (the proof is deferred to Appendix A).

Lemma 2. *Suppose* n *is a safe RSA modulus. If* $A^a = 1 \bmod n^2$ *where* $A \in \mathbb{Z}_{n^2}^*$ *and* $gcd(a, n \cdot \lambda(n)) = 1$ *or* 2, *then* $A = \pm 1 \bmod n^2$.

5 Building Blocks

Now we present three building blocks: a dynamic accumulator accumulating *composites* (Section 5.1), a protocol allowing very efficient *ownership-proofs* (Section 5.2), and a protocol ensuring verifiable encryption of an accumulated value (Section 5.3). Combining them appropriately, we obtain a group signature scheme.

5.1 A Composite Accumulator

We give a definition of dynamic accumulators that accumulate *composites*, which is a variant of the prior construct that accumulates *primes* [CL02].

Definition 6. *A dynamic accumulator for a family of inputs* $\{\mathfrak{X}_l\}$ *is a family of families of functions* $\{\mathcal{F}_l\}$ *with the following properties:*

- Generation. *There is an efficient probabilistic algorithm $\mathcal{G}$ that on input 1^l produces a random element f of $\mathcal{F}_l$, and some auxiliary information aux_f about f.*
- Evaluation. *$f \in \mathcal{F}_l$ is a polynomial-size circuit that, on input $(u,x) \in \mathfrak{U}_f \times \mathfrak{X}_l$, outputs a value $v \in \mathfrak{U}_f$, where $\mathfrak{U}_f$ is an efficiently-samplable input domain for the function f, $\mathfrak{X}_l$ is the intended input domain whose elements (i.e., composites) are to be accumulated.*
- Quasi-Commutative. *For all l, for all $f \in \mathcal{F}_l$, for all $u \in \mathfrak{U}_f$, for all $x_1, x_2 \in \mathfrak{X}_l$, $f(f(u,x_1),x_2) = f(f(u,x_2),x_1)$. If $\mathfrak{X} = \{x_1, \cdots, x_m\} \subset \mathfrak{X}_l$, then by $f(u,\mathfrak{X})$ we denote $f(\cdots f(f(u,x_1),\cdots),x_m)$.*
- Witness. *Let $v \in \mathfrak{U}_f$ and $x \in \mathfrak{X}_l$. A value $w \in \mathfrak{U}_f$ is called a witness for x in v under f if $v = f(w,x)$.*
- Addition. *Let $f \in \mathcal{F}_l$, and $v = f(u,\mathfrak{X})$ be the accumulator so far. There is an efficient algorithm $\mathcal{A}$ to accumulate a given value $x' \in \mathfrak{X}_l$. The algorithm outputs: (1) $\mathfrak{X}' = \mathfrak{X} \cup \{x'\}$ and $v' = f(v,x') = f(u,\mathfrak{X}')$; (2) w' which is the witness for $x \in \mathfrak{X}$ in v'.*
- Deletion. *Let $f \in \mathcal{F}_l$, and $v = f(u,\mathfrak{X})$ be the accumulator so far. There exist efficient algorithms $\mathcal{D}$, $\mathcal{W}$ to delete an accumulated value $x' \in \mathfrak{X}$. The functionality of the algorithms includes: (1) $\mathcal{D}(aux_f, v, x') = v'$ such that $v' = f(u, \mathfrak{X} \setminus \{x'\})$, and (2) $\mathcal{W}(w,x,x',v,v') = w'$ such that $f(w',x) = v'$, where $x \in \mathfrak{X}$ and $f(w,x) = v$.*

Definition 7. *Let $\mathfrak{U}'_f \times \mathfrak{X}'_l$ denote the domains for which the function $f \in \mathcal{F}_l$ is defined (thus $\mathfrak{U}_f \subseteq \mathfrak{U}'_f$, $\mathfrak{X}_l \subseteq \mathfrak{X}'_l$). To capture security of a dynamic accumulator accumulating composites, we consider the following game: At the beginning of the game, an accumulator manager sets up the function f and the value u and hides the trapdoor information aux_f. Then, the adversary $\mathcal{ADV}$ is allowed to adaptively modifies the set, $\mathfrak{X}$, of accumulated values: When a value $x \in \mathfrak{X}_l$ is added, the manager updates the accumulator value using algorithm $\mathcal{A}$; when a value $x \in \mathfrak{X}$ is deleted, the manager algorithm $\mathcal{D}$ publishes the result. We say $\mathcal{ADV}$ wins in this game, if it, with non-negligible probability, manages to output a witness w' for a value $x' \in \mathfrak{X}'_l$ such that $x' \nmid \prod_{\forall x \in \mathfrak{X}} x$. More formally, we require that:*

$$Pr[(f, aux_f) \leftarrow \mathcal{G}(1^l); u \leftarrow \mathfrak{U}_f; (w, x', \mathfrak{X}) \leftarrow \mathcal{ADV}^{\mathcal{O}_{add}, \mathcal{O}_{del}}(f, u, \mathfrak{U}_f) : \\ w' \in \mathfrak{U}'_f; x' \in \mathfrak{X}'_l; x' \nmid \prod_{\forall x \in \mathfrak{X}} x;\ f(w',x') = f(u,\mathfrak{X})]$$

to be negligible, where $\mathcal{O}_{add}$ ($\mathcal{O}_{del}$) is the oracle for the Addition *(resp.* Deletion*) operations. (Note that only a legitimately accumulated value x must belong to $\mathfrak{X}_l$, whereas a forged value x' can belong to a possibly larger set $\mathfrak{X}'_l$.)*

Construction. This construction is a variant of the one accumulating primes [CL02].

- $\mathcal{F}_l$ is the family of functions that correspond to exponentiation modulo safe RSA modulus drawn from the integers of length l. Choose $f \in \mathcal{F}_l$ amounts to choosing a random safe RSA modulus $n = pq$ of length l, where $p = 2p' + 1$, $q = 2q' + 1$. We will denote by f the function corresponding to modulus n and domain $\mathfrak{X}_{A,B}$ by $f_{n,A,B}$.
- $\mathfrak{X}_{A,B} = \{e_1 e_2 : e_1 \in \mathfrak{S}_1 \bigwedge e_2 \in \mathfrak{S}_2\}$, where $\mathfrak{S}_1 = \{e : e \in primes \bigwedge e \neq p' \bigwedge e \neq q' \bigwedge A_1 \leq e \leq B_1\}$, $\mathfrak{S}_2 = \{e : e \in primes \bigwedge e \neq p' \bigwedge e \neq q' \bigwedge A_2 \leq e \leq B_2\}$, A_1, A_2, B_1, and B_2 can be chosen with arbitrary polynomial dependence on the security parameter l as long as $4 < A_1$, $1 < A_2$, $B_1 < A_1^2$, $B_2 < A_1^2$, and $B_1 B_2 < p'q'$. Then, $\mathfrak{X}'_{A,B} \subseteq \{5, \cdots, A_1^4 - 1\}$ and $\mathfrak{X}_{A,B} \subseteq \mathfrak{X}'_{A,B}$.
- For $f = f_{n,A,B}$, the auxiliary information aux_f is the factorization of n.
- For $f = f_{n,A,B}$, $\mathfrak{U}_f = \{u \in \mathbb{QR}_n : u \neq 1\}$ and $\mathfrak{U}'_f = \mathbb{Z}_n^*$.
- For $f = f_{n,A,B}$, $f(w,x) = w^x \bmod n$. Note that $f(f(w,x_1),x_2) = f(w,\{x_1,x_2\}) = w^{x_1 x_2} \bmod n$.
- Update of the accumulator value. Adding a value x' to the accumulator value v is done by setting $v' = f(v,x') = v^{x'} \bmod n$. Deleting a value x' from the accumulator is done by setting $v' = \mathcal{D}((p,q),v,x') = v^{(x')^{-1} \bmod \phi(n)} \bmod n$.
- Update of witness. Updating the witness w after x' has been added can be done by $w' = f(w,x') = w^{x'}$. In the case that $x' \neq x \in \mathfrak{X}_{AB}$ has been deleted from the accumulator, the witness w can be updated as follows. By the extended GCD algorithm, one can compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha x + \beta x' = 1$ and then $w' = \mathcal{W}(w,x,x',v,v') = (v')^\alpha w^\beta$. This guarantees $f(w',x) = (w')^x = v' \bmod n$ because:

$$w' = (v')^{\alpha} w^{\beta} = (v^{(x')^{-1} \bmod \phi(n)})^{\alpha} w^{\beta} = w^{(\alpha x + \beta x')((x')^{-1} \bmod \phi(n))} = w^{(x')^{-1} \bmod \phi(n)} \bmod n.$$

Note that it is crucial $(x', \phi(n)) = 1$, but this is always guaranteed.

Theorem 1. *Under the* Strong RSA Assumption (SRSA)*, the above construction is a secure dynamic accumulator that accumulates* composites.

Properties other than security are easy to see. The security proof is in Appendix A.

5.2 Proving That One Knows the Factorization of a Committed Value

In order to enable *ownership-proofs*, we adopt the Damgard-Fujisaki commitment scheme [DF02] with slight modification. Nonetheless, our protocol for a signer to prove that she knows the factorization of a committed value is more efficient than the protocol presented in [DF02], and thus may be independently interesting.

The Commitment Scheme. Let l (for the length of the modulus) and k (for challenge length) be security parameters, where $l >> k$. This scheme consists of the following three algorithms.

- Set-up. This algorithm is run by a trusted third party (TTP). Given a security parameter l, TTP chooses a safe RSA modulus $N = PQ$, where $P = 2P' + 1$, $Q = 2Q' + 1$, and $|P'| = |Q'| = l/2$. Denote by $\mathbb{G} = \mathbb{QR}_N$ and $l_{\mathbb{G}} = |ord(\mathbb{G})| = l$. TTP chooses two generators of $\mathbb{G}$, G and H, uniformly at random; i.e., $\mathbb{G} = \langle G \rangle = \langle H \rangle$. Note that Fact 1 implies that this can be easily done.
- Commit. To commit to an integer x, the prover chooses $r \in_R \mathbb{Z}_{\lfloor N/4 \rfloor}$ and sends $C = H^x G^r \bmod N$ to the verifier.
- Open. To open a commitment, the prover must send x, r, b such that $C = H^x G^r b \bmod N$, $b = \pm 1$.

Lemma 3. ([DF02]) *The above commitment scheme is perfectly hiding and computationally binding.*

A Protocol for Proving That One Knows the Factorization of a Committed Value. Suppose X is a given random integer such that $|X| = \lambda_1$. Let $\epsilon > 1$ be a security parameter for statistical zero-knowledge, λ_2 denote length such that $l/2 > \lambda_1 > \epsilon(\lambda_2 + k) + 2$. Alice who holds e is to prove that she knows the factorization of $e = e_1 e_2$, where $e_1 \in \{X - 2^{\lambda_2}, \cdots, X + 2^{\lambda_2}\}$ and $e_2 \neq 0, \pm 1$. The protocol goes as follows.

1. The prover, Alice, chooses $r_1 \in_R \pm\{0,1\}^{l+k}$ and generates $C_1 = H^{e_1} G^{r_1} \bmod N$, $C_3 = (C_1)^{e_2} \bmod N$. In order to prove the knowledge of $e = e_1 e_2, e_1, e_2, r_1, r = r_1 e_2$ such that
$$C_1 = H^{e_1} G^{r_1} \bmod N \bigwedge C_3 = H^e G^r \bmod N \bigwedge C_3 = (C_1)^{e_2} \bmod N,$$
she executes as follows:
   - choose $e_1' \in_R \pm\{0,1\}^{\epsilon(\lambda_2+k)}$, $e_2' \in_R \pm\{0,1\}^{\epsilon(\lambda_1+k+1)}$, $e' \in_R \pm\{0,1\}^{\epsilon(2\lambda_1+k+1)}$, $r_1' \in_R \pm\{0,1\}^{\epsilon(l+2k)}$, $r' \in_R \pm\{0,1\}^{\epsilon(l+\lambda_2+2k+1)}$.
   - compute $C_1' = H^{e_1'} G^{r_1'} \bmod N$, $C_{3a}' = H^{e'} G^{r'} \bmod N$, $C_{3b}' = (C_1)^{e_2'} \bmod N$.
   - send $(C_1, C_3, C_1', C_{3a}', C_{3b}')$ to the verifier.
2. The verifier, Bob, chooses $c \in_R \{0,1\}^k$ and sends c to Alice.
3. Alice sends Bob $(s_{e_1}, s_{r_1}, s_{e_2}, s_e, s_r)$, where $s_{e_1} = e_1' - c(e_1 - X)$, $s_{r_1} = r_1' - c \cdot r_1$, $s_e = e' - c \cdot e$, $s_r = r' - cr$, $s_{e_2} = e_2' - c \cdot e_2$ (all in $\mathbb{Z}$).
4. Bob accepts if: $H^{s_{e_1}} G^{s_{r_1}} = C_1' C_1^{-c} H^{c 2^{\lambda_1}} \bmod N$, $H^{s_e} G^{s_r} = C_{3a}' C_3^{-c} \bmod N$, $C_1^{s_{e_2}} = C_{3b}' C_3^{-c} \bmod N$, $s_{e_1} \in \{-2^{\epsilon(\lambda_2+k)+1}, \cdots, 2^{\epsilon(\lambda_2+k)+1}\}$, $s_{e_2} \in \{-2^{\epsilon(\lambda_1+k+1)+1}, \cdots, 2^{\epsilon(\lambda_1+k+1)+1}\}$, $s_e \in \{-2^{\epsilon(2\lambda_1+k+1)+1}, \cdots, 2^{\epsilon(2\lambda_1+k+1)+1}\}$, $C_3 \neq 1$, and $C_3 \neq (C_1)^b \bmod N$ where $b = \pm 1$.

The proof of the following lemma is available in Appendix A.

Lemma 4. *The above protocol is an honest verifier statistical zero-knowledge proof of knowledge* e, e_1, e_2 *such that* $e = e_1 e_2$, $e_1 \in \{X - 2^{\epsilon(\lambda_2+k)+2}, \cdots, X + 2^{\epsilon(\lambda_2+k)+2}\}$, $e_2 \in \{-2^{\epsilon(\lambda_1+k+1)+2}, \cdots, 2^{\epsilon(\lambda_1+k+1)+2}\} \setminus \{0, \pm 1\}$, $e \in \{-2^{\epsilon(2\lambda_1+k+1)+2}, \cdots, 2^{\epsilon(2\lambda_1+k+1)+2}\}$.

5.3 Verifiable Encryption of a Committed Value

In order to facilitate the Open process, we need to force the signer to present an encryption of her accumulated value e for which she proves that she knows its non-trivial factorization $e = e_1 e_2$. For this purpose, we need a verifiable encryption scheme. Here we present such a scheme based on the CGHN public key cryptosystem.

Specifically, suppose public values N, G, and H are chosen according to the commitment scheme in Section 5.2. Let $pk = \langle n, t \rangle$ be a CGHN public key and $sk = \langle n, t, p, q \rangle$ be the corresponding private key, where $n = pq$, $|n| = |N|$, and t is a prime such that $|t| > k$. The prover generates a ciphertext $Y = (1+n)^x r^t \bmod n^2$ and a commitment $C = H^x G^z \bmod N$, where $r \in \mathbb{Z}_n^*$ and $z \in_R \mathbb{Z}_{\lfloor N/4 \rfloor}$. The prover needs to show that the ciphertext Y indeed corresponds to the committed secret x. The protocol is as follows:

1. The prover chooses $x' \in_R \pm\{0,1\}^{\epsilon(l_2+k)}$, $r' \in_R \mathbb{Z}_n^*$, $z' \in_R \{0,1\}^{\epsilon(l+k)}$, computes and sends to the verifier $Y' = (1+n)^{x'}(r')^t \bmod n^2$ and $C' = H^{x'} G^{z'} \bmod N$.
2. The verifier responses with a random challenge $c \in_R \{0,1\}^k$.
3. The prover responses with $s_x = x' - cx$ (in $\mathbb{Z}$), $s_r = r^{-c} r' \bmod n^2$, and $s_z = z' - cz$ (in $\mathbb{Z}$).
4. The verifier accepts if $s_x \in \{-2^{\epsilon(l_2+k)+1}, 2^{\epsilon(l_2+k)+1}\}$, $(1+n)^{s_x}(s_r)^t = Y'Y^{-c} \bmod n^2$, and $H^{s_x} G^{s_z} = C'C^{-c} \bmod N$.

The following lemma is proved in Appendix A.

Lemma 5. *The above protocol is an honest-verifier statistical zero-knowledge proof of knowledge x, r, z.*

6 A New Group Signature Scheme

As highlighted in Section 3, the basic idea underlying our group signature scheme is to utilize an accumulator accumulating *composites* such as $e = e_1 e_2$, where e_1 and e_2 are only known to the user who generates it. Suppose v is the accumulator value. This knowledge allows the user to conduct an *ownership-proof* by demonstrating that she knows the factorization of a committed e, whereas the witness w facilitates an *authorization-proof* that $w^e = v \bmod n$.

6.1 Setup

Initialization of the system includes that a group manager establishes some cryptographic parameters and that a TTP establishes some common auxiliary strings. (We discuss the implications of using a TTP in Section 7.4 below.) Specifically:

1. Let l, k, and $\epsilon > 1$ be security parameters. Let X be a random integer of length $|X| = \lambda_1$. Suppose λ_2 denotes length such that $l/2 > \lambda_1 > \epsilon(\lambda_2 + k) + 2$. Denote by $A = X - 2^{\lambda_2}$ and $B = X + 2^{\lambda_2}$. Define the integral ranges that $\Lambda_1 = \{A, \cdots, B\}$, $\Lambda_2 = \{2^{\lambda_1}, \cdots, 2^{\lambda_1+1} - 1\}$, and $\Gamma = \{-2^{2\lambda_1+1}, \cdots, 2^{2\lambda_1+1}\}$. Define $\mathfrak{X}_{A,B} = \{e_1 e_2 : e_1 \in \mathfrak{S}_1 \bigwedge e_2 \in \mathfrak{S}_2\}$, where $\mathfrak{S}_1 = \{e : e \in \mathit{primes} \bigwedge e \in \Lambda_1\}$ and $\mathfrak{S}_2 = \{e : e \in \mathit{primes} \bigwedge e \in \Lambda_2\}$. We assume that no probabilistic polynomial-time (in l) algorithm is able to factor $e \in_R \mathfrak{X}_{A,B}$; this is where we need the stronger factoring assumption (see Section 7.1 for more discussion). Note that we have (1) $4 < A$, (2) $B(2^{\lambda_1+1} - 1) < A^3$. Let $\mathfrak{X}'_{A,B}$ be (any subset of) of the set of integer from $\{5, \cdots, A^3 - 1\}$ such that $\mathfrak{X}_{A,B} \subseteq \mathfrak{X}'_{A,B}$. The group manager executes as follows:
   - It chooses a safe RSA modulus $n = (2p'+1)(2q'+1)$ such that $|p'| = |q'| = l/2$. This uniquely determines $\mathbb{QR}_n$, the quadratic residues subgroup modulo n.
   - It establishes an instance of ElGamal public key cryptosystem. Let $\langle y_1 = g_1^{x_1} \bmod n; x_1 \rangle$ be the pair of public and private keys such that $g_1 \in_R \mathbb{QR}_n$ and $x_1 \in_R \mathbb{Z}_{p'q'}^*$.
   - It establishes an instance of CGHN public key cryptosystem. Let $\langle n, t; n, t, p, q \rangle$ be the pair of public and private keys, where t is a prime such that $|t| > k$.
   - It establishes an instance of the dynamic accumulator by choosing $u \in_R \mathbb{QR}_n$, establishing (currently empty) public archives $\mathfrak{A}$ for storing values corresponding to added group members, and $\mathfrak{D}$ for storing values corresponding to deleted group members.

The public and private parameters of the group manager are $(n, t, g_1, y_1, u, \mathfrak{A}, \mathfrak{D}, \mathfrak{X}_{A,B}, \mathfrak{X}'_{A,B})$ and (p', q'), respectively. Note that a signature receiver can verify group signatures without knowing the dynamically updated $\mathfrak{A}$ or $\mathfrak{D}$.

2. Given a security parameter l, a TTP initializes a safe RSA modulus $N = (2P' + 1)(2Q' + 1)$, where $|P'| = |Q'| = l/2$. It also chooses and publishes two random elements $G, H \in_R \mathbb{QR}_N$, where the logarithm of G and H to each other is unknown to any participant in the group signature scheme.

6.2 Join

This protocol is executed between a group member, Alice, and the group manager.

1. Alice chooses two primes $e_1 \in_R \mathfrak{S}_1$ and $e_2 \in_R \mathfrak{S}_2$. This step can be done before the execution of the protocol.
2. Alice sends $e = e_1 e_2$ (in $\mathbb{Z}$) to the group manager.
3. If $A \cdot 2^{\lambda_1} < e < B \cdot (2^{\lambda_1+1} - 1)$, e is odd, and $e \notin \mathfrak{A}$, the group manager stores Alice's membership certificate (v, e) where v is the current accumulator value (when the first user joins the group, $v = u$). It also updates v in the public key file as $v' = f_n(v, e)$, and adds e to $\mathfrak{A}$.
4. Alice gets her membership certificate (w, e) and checks if $f_n(w, e) = w^e = v' \bmod n$, where $w = v$.

Remark. The Join process is very efficient (1 exponentiation for both group manager and new user) because of the following: If a dishonest user, Eve, does not choose e that is hard to factor, then any participant (internal or external) who can find certain non-trivial factor of e may be able to sign on her behalf.

6.3 Revoke

Suppose Eve, who has membership certificate (w, e), is to be expelled from the group. Then the group manager can revoke her membership by updating the current accumulator value v in the public key file: It simply sets $v' = \mathcal{D}(\phi(n), v, e)$, deletes e from $\mathfrak{A}$, and adds e to $\mathfrak{D}$.

6.4 Update

Whenever there is a Join and/or Revoke event, the group manager updates the accumulator value from v to v'. Correspondingly, every group member needs to update her membership certificate. An entry in the archives is called "new" if it was entered after the last time a legitimate group member performed an update. Suppose Bob holds a membership certificate (w, e) such that $f_n(w, e) = v$. Then, he updates his membership certificate to (w', e) such that $f_n(w', e) = v'$:

- For all new $e^* \in \mathfrak{A}$, $w'' = f_n(w, \prod e^*)$ and $v'' = f_n(v, \prod e^*)$.
- For all new $e^* \in \mathfrak{D}$, $w' = \mathcal{W}(w'', e, \prod e^*, v'', v')$.

The cost of this process, for a current member, is is very much in line with the cost of Update in [CL02].

6.5 Sign

Recall that $\langle n, t \rangle$ is the group manager's CGHN public key, and that $y_1 = g_1^{x_1} \bmod n$ is the group manager's ElGamal public key. Suppose that v is the current accumulator value, and that Alice holds (w, e) such that $w^e = v \bmod n$, where $e = e_1 e_2$. Given a message m, Alice generates a group signature as follows.

1. She executes as follows.
   - She chooses $r_1 \in_R \mathbb{Z}_n^*$ and computes a CGHN ciphertext $\delta = (1 + en)r_1^t \bmod n^2$.
   - She chooses $r_2 \in_R \pm\{0, 1\}^{l+k}$ and computes an ElGamal ciphertext (α, β) where $\alpha = g_1^{r_2} \bmod n$ and $\beta = w \cdot y_1^{r_2} \bmod n$.
   - She chooses $r_4 \in_R \pm\{0, 1\}^{l+k}$ and generates commitments $\sigma = H^{e_1} G^{r_4} \bmod N$, $\tau = \sigma^{e_2} = H^e G^{r_4 e_2} \bmod N$.
2. She needs to prove the knowledge of:

– (w, e) such that $w^e = v \bmod n$, where w corresponds to the ElGamal ciphertext (α, β), and e corresponds to the CGHN ciphertext δ.
– e_1 and e_2 such that $e_1 \in \Lambda_1$, $e_2 \in \Lambda_2$, and $e = e_1 e_2 \in \Gamma$.

For this purpose, she needs to prove the knowledge of $e, e_1, e_2, r_1, r_2, r_3 = r_2 e, r_4, r_5 = r_4 e_2$ such that:

$$\delta = (1+n)^e r_1^t \bmod n^2 \bigwedge$$
$$\alpha = g_1^{r_2} \bmod n \bigwedge v = \beta^e (\frac{1}{y_1})^{r_3} \bmod n \bigwedge 1 = \alpha^e (\frac{1}{g_1})^{r_3} \bmod n \bigwedge$$
$$\tau = H^e G^{r_5} \bmod N \bigwedge \sigma = H^{e_1} G^{r_4} \bmod N \bigwedge \tau = \sigma^{e_2} \bmod N \bigwedge$$
$$e \in \Gamma \bigwedge e_1 \in \Lambda_1 \bigwedge e_2 \in \Lambda_2.$$

Specifically, she executes as follows:

(a) She executes the following steps:
   – Choose $e' \in \pm\{0,1\}^{\epsilon(2\lambda_1+k+1)}$ and $r_1' \in_R \mathbb{Z}_n^*$, and compute $\delta' = (1+n)^{e'} (r_1')^t \bmod n^2$.
   – Choose $r_2' \in_R \pm\{0,1\}^{\epsilon(l+2k)}$, $r_3' \in_R \pm\{0,1\}^{\epsilon(l+2\lambda_1+2k+1)}$, and generate:
   $$\alpha' = g_1^{r_2'} \bmod n, \quad v' = \beta^{e'} (\frac{1}{y_1})^{r_3'} \bmod n, \quad \omega' = \alpha^{e'} (\frac{1}{g_1})^{r_3'}.$$
   – Choose $e_1' \in_R \pm\{0,1\}^{\epsilon(\lambda_2+k)}$, $e_2' \in \pm\{0,1\}^{\epsilon(\lambda_1+k+1)}$ $r_4' \in_R \pm\{0,1\}^{\epsilon(l+2k)}$, $r_5' \in_R \pm\{0,1\}^{\epsilon(l+\lambda_1+2k+1)}$, and generate:
   $$\tau_1' = H^{e'} G^{r_5'} \bmod N, \quad \sigma' = H^{e_1'} G^{r_4'} \bmod N, \quad \tau_2' = \sigma^{e_2'} \bmod N.$$

(b) She computes $c = \mathcal{H}(m, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \tau, \sigma, \delta', \alpha', v', \omega', \tau_1', \sigma', \tau_2')$, where $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ behaves like a random oracle.

(c) She computes (all the operations, except the computation of s_{r_1}, are in $\mathbb{Z}$):
$$s_e = e' - c \cdot e, \quad s_{e_1} = e_1' - c \cdot (e_1 - X), \quad s_{e_2} = e_2' - c \cdot e_2,$$
$$s_{r_1} = r_1^{-c} \cdot r_1' \bmod n^2, \quad s_{r_2} = r_2' - c \cdot r_2, \quad s_{r_3} = r_3' - c \cdot r_3,$$
$$s_{r_4} = r_4' - c \cdot r_4, \quad s_{r_5} = r_5' - c \cdot r_5.$$

(d) She sends Bob $(m, c, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \sigma, \tau, s_e, s_{e_1}, s_{e_2}, s_{r_1}, s_{r_2}, s_{r_3}, s_{r_4}, s_{r_5})$.

Cost: Sign requires 17 exponentiations. Note that 2 of the 17 exponentiations are $r^t \bmod n^2$ but $t << n$ (e.g., $|t| = 161$).

Note that, in counting exponentiations above, we did not consider various implementation speedup techniques, such as savings in computing the multiplication of double- or triple-based exponentiations (e.g., $x^a y^b$ or $x^a y^b z^c$). Our conservative cost assessment is not uncommon in the literature, perhaps because it better reflects the underlying techniques rather than specific implementation speedups. Still, when we take into account speedup techniques, our Sign process would require approximately 12.85 exponentiations, This is in contrast to the optimized Sign cost of roughly 16 exponentiations in [CL02].

6.6 Verify

Given $(m, c, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \sigma, \tau, s_e, s_{e_1}, s_{e_2}, s_{r_1}, s_{r_2}, s_{r_3}, s_{r_4}, s_{r_5})$, Bob checks if it is a valid signature as follows.

1. Bob computes $c' = \mathcal{H}(m, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \tau, \sigma, \delta', \alpha', v', \omega', \tau_1', \sigma', \tau_2')$, where
$$\delta' = (1+n)^{s_e} (s_{r_1})^t \delta^c \bmod n^2, \qquad \alpha' = g_1^{s_{r_2}} \alpha^c \bmod n, \qquad v' = \beta^{s_e} (\frac{1}{y_1})^{s_{r_3}} v^c \bmod n,$$
$$\omega' = \alpha^{s_e} (\frac{1}{g_1})^{s_{r_3}} \bmod n, \quad \tau_1' = H^{s_e} G^{s_{r_5}} \tau^c \bmod N, \quad \sigma' = H^{s_{e_1} - c \cdot 2^{\lambda_1}} G^{s_{r_4}} \sigma^c \bmod N,$$
$$\tau_2' = \sigma^{s_{e_2}} \tau^c \bmod N.$$

2. Bob accepts if $c = c'$, $s_{e_1} \in \{-2^{\epsilon(\lambda_2+k)+1}, \cdots, 2^{\epsilon(\lambda_2+k)+1}\}$, $s_{e_2} \in \{-2^{\epsilon(\lambda_1+k+1)+1}, \cdots, 2^{\epsilon(\lambda_1+k+1)+1}\}$, $s_e \in \{-2^{\epsilon(2\lambda_1+k+1)+1}, \cdots, 2^{\epsilon(2\lambda_1+k+1)+1}\}$, $\tau \neq 1 \bmod N$, and $\tau \neq \sigma^b \bmod N$ where $b = \pm 1$.

Cost: Verify, without any optimizations, requires 16 exponentiations which is somewhat more efficient than 21 exponentiations in [CL02]. Furthermore, if (as in **Sign**) we factor in the speedup due to double-based exponentiations, the cost of Verify would go down to approximately 8.4 exponentiations. We note that optimized Verify cost of [CL02] is roughly 9.1 exponentiations. However, we believe that the Verify process in the latter is incomplete; a complete version would require a few more exponentiations in both non-optimized and optimized cases.

6.7 Open

Given a valid group signature $(m, c, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \sigma, \eta, \tau, s_e, s_{e_1}, s_{e_2}, s_{r_1}, s_{r_2}, s_{r_3}, s_{r_4}, s_{r_5})$, the group manager can identify the signer by decrypting both w and e such that $w^e = v \bmod n$. It also needs to prove that the decryption of w is correct; namely $DLOG(g_1, y_1) = DLOG(\alpha, \beta/w)$.

1. It decrypts the CGHN ciphertext δ to obtain e, and decrypts the ElGamal ciphertext $\langle\alpha, \beta\rangle$ to obtain w. It must hold that $A^3 > e > 1$.
2. There are further two cases.
   (a) If $e \in \mathfrak{A}$, then it publishes: (1) the values w and e, and (2) the proof that $DLOG(g_1, y_1) = DLOG(\alpha, \beta/w)$. Note that knowing w and e does not expose neither previous, nor future (even if the system policy allows), signatures generated by the same group member.
   (b) If $e \notin \mathfrak{A}$, then it must hold that $e | \prod_{\forall e' \in \mathfrak{A}} e'$. Therefore, there must exist $e' \in \mathfrak{A}$ such that $e' > gcd(e, e') > 1$. Therefore, the group member corresponding to accumulated e' is identified (and revoked).

   We note that the first case reflects the more common situation, i.e., a well-formed signature needs to be opened. The second, more infrequent, case involves a misbehaving user, or a set thereof. Here, the complexity of opening a signature is linear to the number of current group members (in terms of computing a GCD).

6.8 Analysis

The following theorem is proven in Appendix B.

Theorem 2. (informal) *The above scheme is a secure group signature scheme.*

Corollary 1. *The interactive version of the above group signature scheme is a secure identity escrow scheme.*

Note that the interactive scheme can be made concurrently secure in the standard model using the technique due to [D00]. This adds no extra complexity because we already work in the common auxiliary string model.

7 Discussion

7.1 Factorization Assumption

For typical group signature applications we suggest that the group manager use 2048-bit RSA moduli. For other parameters, we suggest (as an example): $\lambda_1 = 950$, $\lambda_2 = 700$, $\epsilon = 1.1$, $k = 160$. This means that we assume the hardness of factoring large 2-prime composites, where $(\lambda_1 - \lambda_2)$ high-order bits of one prime are known. This assumption is stronger than the standard factorization assumption. However, despite the fixed prefix, it still seems reasonable to assume the hardness of factoring such a composite. Given partial knowledge of the factorization, the best factoring algorithm currently available indicates that, if the higher 475-bits of a prime factor are known, then one can factor n [C96]. Beyond that, no better result is available [C03]. Note that if the higher bits of one prime factor are known, then the higher bits of another factor are also exposed. Nevertheless, knowing $\langle \sigma, \tau = \sigma^{e_2} \bmod N \rangle$ still requires an adversary to compute e_2 in $O(2^{350})$ time (see [G00] and the references therein).

The main reason that we rely on the hardness of factoring a product of two primes (where one prime is in the aforementioned "special" form) is only for the purpose of making the scheme more efficient. This is because we use a range proof technique that does not guarantee precise range. This can, of course, be easily overcome by adopting a more precise range proof technique due to [B00]. Although this would obviate the need for the special factoring assumption, approximately 15 extra exponentiations would be required. We believe there are other ways to obviate this assumption, however, all of them incur significant extra cost.

Finally, we note that a very similar assumption was used before (e.g., by Camenisch and Michels in [CM98]).

7.2 "Lazy" Accumulator Update?

In a group signature scheme based on a dynamic accumulator, it is necessary for both signer and verifier to get the updated accumulator whenever there is a member leaves. In the Camenisch-Lysyanskaya scheme, they suggest a nice trick whereby a Join may not have to trigger a group member to get the updated accumulator value. While this trick enables potential gain in communications, it may incur some serious problems in practice. Consider the following scenario: since Alice is lazy, she does not contact the group manager to check the current accumulator value. Instead, she waits for a broadcast message from the group manager. If this message is blocked by an adversary, there is no way for Alice to tell if there has been an accumulator update. Consequently, Alice would generate a group signature which is valid with respect to the outdated accumulator value, i.e., the previous accumulator incarnation. However, the signature is invalid with respect to the current accumulator value. It is unclear how a potential dispute involving this signature can be resolved. At best, the verifier can abuse such a signature.

We suggest that Alice should be diligent and prevent such anomalies by actively querying the group manager for the current accumulator value. This way, if she does not elicit any reply from the group manager, she can simple refuse to generate any group signatures.

7.3 Enhancing Anonymity

Recently, Bellare et al. [BMW03] proposed simplified security requirements for group signatures. The most relevant part of their definition that has an impact on our scheme is the so-called *full-anonymity*, which is strictly stronger than the *anonymity* specified in the present paper (as well as all previously proposed schemes). Essentially, *full-anonymity* allows the adversary to compromise *all* group members, whereas we assume that there are at least two non-compromised group members. Our scheme does not satisfy their *full-anonymity*, but can be extended to do so. The idea is to utilize the protocol of [DF02] to prove that a signer knows the factorization of a committed value. However, this will incur 6 extra exponentiations in Sign; this complexity justifies why we developed the more efficient protocol in Section 5.2.

7.4 TTP Presence

Our scheme operates in the *common auxiliary string* model which assumes a common string (the specification of a commitment scheme) generated by a trusted third party (TTP) and made available to all participants. The inconvenience posed by this is not significant owing to the following mitigating factors:

- The TTP's role is only to initialize the cryptographic setting of a commitment scheme. In fact, the TTP can simply disappear after publishing the commitment scheme parameters since it is not involved in any future transactions.
- A single TTP could serve multiple group signature settings, thereby amortizing the complexity. Moreover, threshold cryptography can be used to implement a distributed TTP (see [ACS02]).
- Currently, the most efficient method of obtaining identity escrow schemes (such as [KP98]) that are *concurrently secure* is based on the existence of common auxiliary strings [D00]. Therefore, the identity escrow scheme derived from our group signature scheme can be made concurrently secure without incurring any extra complexity.

One might question whether the *common auxiliary string* model is truly needed, given that we are proving factorization or multiplication in $\mathbb{Z}$. Unfortunately, we cannot afford to let the group manager act as a TTP since, otherwise, the commitment in Lemma 3 is not binding in the case the group manager attempts to frame an honest member.

8 Conclusion

We presented a dynamic accumulator construct that accumulates *composites*, and an efficient protocol for proving knowledge of the factorization of a committed value. Based on these techniques, we developed a novel, efficient and provably secure group signature scheme.

Acknowledgements

We thank Don Coppersmith, Ivan Damgard, and Moti Yung for valuable feedback and suggestions. We also acknowledge the anonymous reviewers for Crypto'03 for their useful comments. Finally, we are grateful to Mihir Bellare and Daniele Micciancio for a preview copy of [BMW03].

References

[ACS02] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient Computation Modulo a Shared Secret with Application to the Generation of Shared Safe-Prime Products. Crypto'02.

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. Crypto'00.

[AST02] G. Ateniese, D. Song, and G. Tsudik. Quasi-Efficient Revocation of Group Signatures. Financial Crypto'02.

[AT99] G. Ateniese and G. Tsudik. Some Open Issues and New Directions in Group Signatures. Financial Crypto'99.

[BP97] N. Baric and B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. Eurocrypt'97.

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. FOCS'97.

[BMW03] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction based on General Assumptions. Eurocrypt'03.

[BR93] M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. ACM CCS'93.

[B00] F. Boudot. Efficient Proof that a Committed Number lies in an Interval. Eurocrypt'00.

[B93] S. Brands. An Efficient Off-Line E-Cash Scheme System Based on the Representation Problem. CWI TR CS-R9323.

[BS01] E. Bresson and J. Stern. Group Signatures with Efficient Revocation. PKC'01.

[C98] J. Camenisch. Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD Thesis. ETH Zurich. 1998.

[CL02] J. Camenisch and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. Crypto'02.

[CM98] J. Camenisch and M. Michels. A Group Signature Scheme based on an RSA-variant. Tech. Report RS-98-27, BRICS. Preliminary version appeared at Asiacrypt'98.

[CM99a] J. Camenisch and M. Michels. Separability and Efficiency for Generic Group Signature Schemes (Extended Abstract). Crypto'99.

[CS97] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). Crypto'97.

[CGHN01] D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's Cryptosystem Revisited. ACM CCS'01.

[CFT98] A. Chan, Y. Frankel, and Y. Tsiounis. Each Come - Easy Go Divisible Cash. Eurocrypt'98.

[CP94] L. Chen and T. Pedersen. New Group Signature Schemes. Eurocrypt'94.

[CvH91] S. Chaum and E. van Heyst. Group Signatures. Eurocrypt'91.

[CP92] D. Chaum and T. P. Pedersen. Wallet Databases with Observers. Crypto'92.

[C96] D. Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with high bits known. Eurocrypt'96.

[C03] D. Coppersmith. Personal Communication. Jan. 2003.
[D00] I. Damgard. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. Eurocrypt'00.
[DF02] I. Damgard and E. Fujisaki. An Integer Commitment Scheme Based on Groups with Hidden Order. Asiacrypt'02.
[E85] T. ElGamal, A Public-Key Cryptosystem and a Signature Scheme Based on the Discrete Logarithm, IEEE Transactions of Information Theory, 31(4), 1985, pp 469–472.
[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. Crypto'86.
[FO97] E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. Crypto'97.
[G00] R. Gennaro. An Improved Pseudo-Random Generator Based on the Discrete Logarithm Problem. Crypto'00.
[GKR00] R. Gennaro, H. Krawczyk, and T. Rabin. RSA-Based Undeniable Signatures. J. Cryptology, (13)4, 2000, pp 397-416.
[GMR88] S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-message Attacks. SIAM J. Computing, 17(2), 1988.
[KP98] J. Kilian and E. Petrank. Identity Escrow. Crypto'98.
[MR01] P. MacKenzie and M. Reiter. Two-Party Generation of DSA Signatures. Crypto'01.
[P99] P. Paillier. Public Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt'99.
[S91] C. Schnorr. Efficient Signature Generation by Smart Cards. Journal of Cryptology 4(3) 161-174, 1991.
[S01] D. Song. Practical Forward Secure Group Signature Schemes. ACM CCS'01.
[TY98] Y. Tsiounis and M. Yung. On the Security of ElGamal Based Encryption. PKC'98.

A Proofs of Some Lemmas

Proof. (of Lemma 2) If $gcd(a, n \cdot \lambda(n)) = 1$, then we can compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha a + \beta n \cdot \lambda(n) = 1$. Then we have $A = (A^a)^\alpha (A^\beta)^{n\cdot\lambda(n)} = 1 \bmod n^2$.

If $gcd(a, n \cdot \lambda(n)) = 2$, then we can compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha a + \beta n \cdot \lambda(n) = 2$. Then we have $A^2 = (A^a)^\alpha (A^\beta)^{n\cdot\lambda(n)} = 1 \bmod n^2$. Note that we can write $A = \gamma n + \delta$, where $0 \le \gamma < n$ and $0 < \delta < n$. Thus, $2\gamma\delta n + \delta^2 = 1 \bmod n^2$, which implies $\delta^2 = 1 \bmod n$. Lemma 1 implies that $\delta = \pm 1$. There are two cases:

- $\delta = 1$. Then $2\gamma n = 0 \bmod n^2$. This, the fact that n is a safe RSA modulus, and $0 \le \gamma < n$ imply that $\gamma = 0$. So, $A = 1$.
- $\delta = n - 1$. Then $-2(\gamma + 1)n = 0 \bmod n^2$. This, the fact that n is a safe RSA modulus, and $0 \le \gamma < n$ imply that $\gamma = n - 1$. So, $A = -1 \bmod n^2$.

Proof. (of Theorem 1) This proof only focuses on security. Suppose the adversary $\mathcal{ADV}$ can break the dynamic accumulator with non-negligible probability, then there is a polynomial-time algorithm $\mathcal{B}$ that is able to break SRSA. Suppose $\mathcal{B}$ is given a SRSA instance (n, u), where n is a safe RSA modulus, $u \in_R \mathbb{QR}_n$. $\mathcal{B}$ simulates the system as follows:

- It feeds $(f, \mathfrak{U}_f, u)$ to $\mathcal{ADV}$. Let $\mathfrak{X} = \emptyset$. It executes as follows:
  1. When $\mathcal{ADV}$ asks to accumulate a value $x \notin \mathfrak{X}$, it simply updates $\mathfrak{X} = \mathfrak{X} \cup \{x\}$ and $v = f(u, \mathfrak{X})$.
  2. When $\mathcal{ADV}$ asks to delete a value $x \in \mathfrak{X}$, it simply updates $\mathfrak{X} = \mathfrak{X} \setminus \{x\}$ and $v = f(u, \mathfrak{X} \setminus \{x\})$.
- At some point, $\mathcal{ADV}$ comes up with a forgery $(w', x', \mathfrak{X} = \{x_1, \cdots, x_m\})$ such that (1) $(w')^{x'} = u^{\prod_{i=1}^m x_i} \bmod n$ where $w' \in \mathbb{Z}_n^*$ and $x' \in \mathfrak{X}'_{A,B}$; (2) $x' \nmid \prod_{i=1}^m x_i$.

Now, since $x' \nmid \prod_{i=1}^m x_i$, we have $gcd(x', \prod_{i=1}^m x_i) = c$ for some $c < x'$. So $\mathcal{B}$ can simply find $\alpha, \beta \in \mathbb{Z}$ such that $\alpha \prod_{i=1}^m x_i + \beta x' = c$. Since $u^{\prod_{i=1}^m x_i} = (w')^{x'} \bmod n$, $\mathcal{B}$ simply outputs $u^\beta (w')^\alpha$ such that $u^c = u^{\alpha \prod_{i=1}^m x_i + \beta x'} = ((w')^\alpha u^\beta)^{x'}$. Therefore, $u = ((w')^\alpha u^\beta)^{x'/c} b$, where $x'/c > 1$ and $b^c = 1$. Now we claim that $b = \pm 1$. Otherwise, $\mathcal{B}$ can factor n: If $c < p'q'$ which is unknown to $\mathcal{B}$ though, then Lemma 1 implies that $\mathcal{B}$ can factor n; If $c \ge p'q'$, then $\mathcal{B}$ can also factor n using the standard technique.

Proof. (of Lemma 4) Completeness can be checked by observation. It is a standard process to show that the protocol is honest verifier statistical zero-knowledge. So we focus on the property of proof of knowledge.

Suppose a prover presents two valid answers $(s_{e_1}, s_{r_1}, s_{e_2}, s_e, s_r)$ and $(s^*_{e_1}, s^*_{r_1}, s^*_{e_2}, s^*_e, s^*_r)$ to two different challenges c and c^*, respectively. Without loss of generality, suppose $c > c^*$. The result in [DF02] implies $C_1 = H^{\tilde{e_1}} G^{\tilde{r_1}} b_1 \bmod N$ and $C_3 = H^{\tilde{e}} G^{\tilde{r}} b_3 \bmod N$, where $\tilde{e_1} = X + \frac{s_{e_1} - s_{e^*_1}}{c^* - c}$, $\tilde{r_1} = \frac{s_{r_1} - s^*_{r_1}}{c^* - c}$, $b_1 = \pm 1$, $\tilde{e} = \frac{s_e - s_{e^*}}{c^* - c}$, $\tilde{r} = \frac{s_r - s^*_r}{c^* - c}$, $b_3 = \pm 1$.

Because of the fact that $C_1^{s_{e_2}} = C'_{3b} C_3^{-c} \bmod N$ and $C_1^{s^*_{e_2}} = C'_{3b} C_3^{-c^*} \bmod N$, we get $C_1^{s_{e_2} - s^*_{e_2}} = C_3^{-(c-c^*)} \bmod N$. Therefore, we have

$$(H^{\tilde{e_1}} G^{\tilde{r_1}} b_1)^{s_{e_2} - s^*_{e_2}} = (H^{\tilde{e}} G^{\tilde{r}} b_3)^{c^* - c} \bmod N.$$

Thus, we get

$$H^{\tilde{e_1}(s_{e_2} - s^*_{e_2}) - \tilde{e}(c^* - c)} G^{\tilde{r_1}(s_{e_2} - s^*_{e_2}) - \tilde{r}(c^* - c)} b_1^{(s_{e_2} - s^*_{e_2})} b_3^{c - c^*} = 1 \bmod N.$$

Since $G, H \in \mathbb{QR}_N$, we have $b_1^{s_{e_2} - s^*_{e_2}} b_3^{c - c^*} = 1 \bmod N$.

The discrete logarithm assumption (which is implied by the Decisional Diffie-Hellman Assumption) implies

$$\tilde{e_1}(s_{e_2} - s^*_{e_2}) - \tilde{e}(c^* - c) = 0 \bmod ord(G)$$

and

$$\tilde{r_1}(s_{e_2} - s^*_{e_2}) - \tilde{r}(c^* - c) = 0 \bmod ord(G).$$

Recall that $ord(G) = P'Q' = ord(\mathbb{QR}_N)$ where N is a safe RSA modulus. The hardness of factoring (which is implied by the Strong RSA Assumption) implies

$$\tilde{e_1}(s_{e_2} - s^*_{e_2}) - \tilde{e}(c^* - c) = 0 \text{ (in } \mathbb{Z})$$

and

$$\tilde{r_1}(s_{e_2} - s^*_{e_2}) - \tilde{r}(c^* - c) = 0 \text{ (in } \mathbb{Z}).$$

In other words, we get

$$\tilde{e_1}(s_{e_2} - s^*_{e_2}) = \tilde{e}(c^* - c)$$

and

$$\tilde{r_1}(s_{e_2} - s^*_{e_2}) = \tilde{r}(c^* - c).$$

The range condition says that $\tilde{e_1} \in \{X - 2^{\epsilon(\lambda_2 + k) + 2}, X + 2^{\epsilon(\lambda_2 + k) + 2}\}$. Since $X - 2^{\epsilon(\lambda_2 + k) + 2} > 2^k$, we get $\tilde{e_1} > c$, which means that $\tilde{e_1} \nmid (c^* - c)$ and that $\bar{e_1} = \frac{\tilde{e_1}}{gcd(\tilde{e_1}, c^* - c)} > 1$. Similarly, we can define $\bar{r_1} = \frac{\tilde{r_1}}{gcd(\tilde{r_1}, c^* - c)} > 1$. Therefore, $\bar{e_1} | \tilde{e}$.

Proof. (of Lemma 5) It is easy to check completeness. It is also a standard process to prove honest verifier statistical zero-knowledge. So we only focus on proof of knowledge. For simplifying notations, denote $(1 + n)$ by g.

Suppose the adversary is able to present two valid answers (s_x, s_r, s_z) and (s^*_x, s^*_r, s^*_z) to two different challenges $c^* \neq c$, respectively. Without loss of generality, assume $c > c^*$. Then we have $g^{s_x}(s_r)^n = Y'Y^{-c} \bmod n^2$, $H^{s_x} G^{s_z} = C'C^{-c} \bmod N$, $g^{s^*_x}(s^*_r)^t = Y'Y^{-c^*} \bmod n^2$, and $H^{s^*_x} G^{s^*_z} = C'C^{-c^*} \bmod N$. The result in [DF02] imply that $(c^* - c) | (s_x - s^*_x)$ and $(c^* - c) | (s_z - s^*_z)$, and $C = H^{\frac{s_x - s^*_x}{c^* - c}} G^{\frac{s_z - s^*_z}{c^* - c}} b_1 \bmod N$ where $b_1 = \pm 1$. On the other hand, we have $g^{s_x - s^*_x}(\frac{s_r}{s^*_r})^t = Y^{-(c - c^*)} \bmod n^2$. Therefore, we have

$$(\frac{s_r}{s_r^*})^t = (g^{-\frac{s_x - s_x^*}{c-c^*}} Y^{-1})^{c-c^*} \bmod n^2. \tag{1}$$

Since $gcd(c - c^*, t) = 1$, we can compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha(c - c^*) + \beta t = 1$. Therefore, we have

$$\begin{aligned}\frac{s_r}{s_r^*} &= (\frac{s_r}{s_r^*})^{\alpha(c-c^*)} (g^{-\frac{s_x - s_x^*}{c-c^*}} Y^{-1})^{\beta(c-c^*)} \\ &= ((\frac{s_r}{s_r^*})^{\alpha} (g^{-\frac{s_x - s_x^*}{c-c^*}} Y^{-1})^{\beta})^{c-c^*} \bmod n^2.\end{aligned}$$

Let $\sigma = (\frac{s_r}{s_r^*})^{\alpha} (g^{-\frac{s_x - s_x^*}{c-c^*}} Y^{-1})^{\beta} b_2 \bmod n^2$ where $b_2^{c-c^*} = 1 \bmod n^2$. Then we have

$$\frac{s_r}{s_{r^*}} = \sigma^{c-c^*} \bmod n^2. \tag{2}$$

It is straightforward to check that $gcd(b, n^2) = 1$. Since $|c| << l/2$ and $c > c^*$, we have $gcd(c-c^*, n \cdot \lambda(n)) = 1$ or 2. So Lemma 2 implies that $b_2 = \pm 1 \bmod n^2$.

Note that (1) and (2) imply $(g^{-\frac{s_x - s_x^*}{c-c^*}} Y^{-1})^{c-c^*} (\sigma^{-(c-c^*)})^t = 1 \bmod n^2$, and thus $(g^{-\frac{s_x - s_x^*}{c-c^*}} Y^{-1} \sigma^{-t})^{c-c^*} = 1 \bmod n^2$. It is easy to check that $gcd(g^{-\frac{s_x - s_x^*}{c-c^*}} Y^{-1} \sigma^{-t}, n^2) = 1$. Since $gcd(c - c^*, n \cdot \lambda(n)) = 1$ or 2, Lemma 2 implies that $g^{-\frac{s_x - s_x^*}{c-c^*}} Y^{-1} \sigma^{-t} b_3 = 1 \bmod n^2$ with $b_3 = \pm 1 \bmod n^2$. So, $Y = g^{-\frac{s_x - s_x^*}{c-c^*}} (\sigma^{-1})^t b_3 \bmod n^2$.

B Security Analysis of the Group Signature Scheme

Lemma 6. *The interactive proof corresponding to* Sign *is an honest verifier statistical zero-knowledge proof of knowledge* (w, e, e_1, e_2) *such that* $w^e = v \bmod n$, $e_1 \in \{X - 2^{\epsilon(\lambda_2+k)+2}, \cdots, X - 2^{\epsilon(\lambda_2+k)+2}\}$, $abs(e_2) > 1$, *and* $e = e_1 e_2 \in \{-2^{\epsilon(2\lambda_1+k+1)+2}, \cdots, 2^{\epsilon(2\lambda_1+k+1)+2}\}$.

Proof. It is a standard process to show it is honest verifier statistical zero-knowledge (see, for instance, [C98]). We thus only focus on proof of knowledge.

Suppose the prover, given two different challenges $c, c^* \in_R \{0,1\}^k$, can present two accepting answers $(s_e, s_{e_1}, s_{e_2}, s_{r_1}, s_{r_2}, s_{r_3}, s_{r_4}, s_{r_5})$ and $(s_e^*, s_{e_1}^*, s_{e_2}^*, s_{r_1}^*, s_{r_2}^*, s_{r_3}^*, s_{r_4}^*, s_{r_5}^*)$, respectively.

Lemma 4 and Lemma 5 imply the following:

- It holds that $(c^* - c)|(s_e - s_e^*)$ except with negligible probability, and that τ can be opened as $\tau = H^{\frac{s_e - s_e^*}{c^*-c}} G^{\frac{s_{r_5} - s_{r_5}^*}{c^*-c}} b_\tau$ where $b_\tau = \pm 1$.
- It holds that $(c^* - c)|(s_{e_1} - s_{e_1}^*)$ except with negligible probability, and that σ can be opened as $\sigma = H^{\frac{s_{e_1} - s_{e_1}^*}{c^*-c} + 2^{\lambda_1}} G^{\frac{s_{r_4} - s_{r_4}^*}{c^*-c}} b_\sigma$ where $b_\sigma = \pm 1$.
- It holds that the committed secret $-\frac{s_e - s_e^*}{c-c^*}$ is indeed encrypted under the CGHN public key.

Now we prove that the committed e in τ is indeed accumulated. Since $g_1^{s_{r_2}} = \alpha' \alpha^{-c}$ and $g_1^{s_{r_2}^*} = \alpha' \alpha^{-c^*} \bmod n$, therefore $g_1^{s_{r_2} - s_{r_2}^*} = \alpha^{c^*-c}$. It follows that (unless SRSA is broken)

$$(c^* - c)|(s_{r_2} - s_{r_2}^*). \tag{3}$$

Therefore we have $r_2 = \frac{s_{r_2} - s_{r_2}^*}{c^*-c}$ and thus

$$w = \beta / y_1^{r_2} \bmod n.$$

Since we have $\alpha^{s_e} (\frac{1}{g_1})^{s_{r_3}} = \omega' = \alpha^{s_e^*} (\frac{1}{g_1})^{s_{r_3}^*}$, we get $\alpha^{s_e - s_e^*} = g_1^{s_{r_3} - s_{r_3}^*}$. It follows that (unless SRSA is broken)

$$(s_e - s_e^*)|(s_{r_3} - s_{r_3}^*). \tag{4}$$

Since $\beta^{s_e}(\frac{1}{y_1})^{s_{r3}} = v' \cdot v^{-c}$ and $\beta^{s_e^*}(\frac{1}{y_1})^{s^*_{r3}} = v' \cdot v^{-c^*}$, we get $\beta^{s_e - s_e^*}(\frac{1}{y_1})^{s_{r3}-s^*_{r3}} = v^{c^*-c}$. This and (4) imply $(\beta(\frac{1}{y_1})^{\frac{s_{r3}-s^*_{r3}}{s_e-s^*_e}})^{s_e-s_e^*} = v^{c^*-c}$, which and $(c^*-c)|(s_e-s_e^*)$ imply $((\beta(\frac{1}{y_1})^{\frac{s_{r3}-s^*_{r3}}{s_e-s^*_e}})^{\frac{s_e-s_e^*}{c^*-c}})^{c^*-c} = v^{c^*-c}$. Therefore, $v = (\beta(\frac{1}{y_1})^{\frac{s_{r3}-s^*_{r3}}{s_e-s^*_e}})^{\frac{s_e-s_e^*}{c^*-c}} \cdot b$ where $b = \pm 1$. Let $sign(z) = -1$ if $z < 0$ and $sign(z) = 1$ otherwise. Thus we have

$$v = \begin{cases} ((\beta(\frac{1}{y_1})^{\frac{s_{r3}-s^*_{r3}}{s_e-s^*_e}} b)^{sign(\frac{s_e-s_e^*}{c^*-c})})^{abs(\frac{s_e-s_e^*}{c^*-c})} & \text{if } \frac{s_e-s_e^*}{c^*-c} \text{ is odd} \\ ((\beta(\frac{1}{y_1})^{\frac{s_{r3}-s^*_{r3}}{s_e-s^*_e}})^{sign(\frac{s_e-s_e^*}{c^*-c})})^{abs(\frac{s_e-s_e^*}{c^*-c})} & \text{if } \frac{s_e-s_e^*}{c^*-c} \text{ is even} \end{cases}$$

Since that $X - 2^{\epsilon(\lambda_2+k)+2} > 4$, that $abs(e_2) > 1$, and that $-2^{\epsilon(2\lambda_1+k+1)+2} < \frac{s_e-s_e^*}{c^*-c} < 2^{\epsilon(2\lambda_1+k+1)+2} < A^3$, we have $abs(\frac{s_e-s_e^*}{c^*-c}) \in \mathfrak{X}'_{A,B}$.

Lemma 7. (No-Framing). *Suppose the group manager, perhaps in collusion with some dishonest group members, is able to generate a signature traceable to an honest group member with membership certificate* (w, e). *Then, there exists a probabilistic polynomial-time algorithm that is able to factor* $e = e_1e_2$ *where* $e_1 \in_R \mathfrak{S}_1$ *and* $e_2 \in_R \mathfrak{S}_2$.

Proof. (sketch) Suppose the group manager can frame an honest group member with non-negligible probability. We construct a probabilistic polynomial-time algorithm $\mathcal{F}$ that is able to factor a challenge $e = e_1e_2$ where $e_1 \in_R \mathfrak{S}_1$ and $e_2 \in_R \mathfrak{S}_2$. The basic idea underlying $\mathcal{F}$ is to emulate an environment for the real-world group signature setting in the random oracle model. Suppose there is already a group signature setting established by the group manager. The system parameters are exactly the same as in the real-world system. $\mathcal{F}$ executes as follows.

1. It joins the group by executing the Join protocol. Denote by (w, e) its membership certificate.
2. It executes exactly as in the real-world system when there is a need to Update its membership certificate.
3. It signs messages as in the real-world system by utilizing the fact that it controls the specification of the random oracle.

Suppose the group manager, perhaps colluding with some dishonest group members, is able to frame $\mathcal{F}$. Then, Lemma 6 implies that there is a polynomial-time algorithm that is able to extract (w, e, e_1, e_2), where e_1 is a non-trivial factor of e.

Theorem 3. *The above construction is a secure group signature scheme.*

Proof. (sketch) We show that the above scheme satisfies all the security properties specified in Section 2.2.

- Correctness. This is immediate to see.
- Unforgeability. Only group members can sign on behalf of the group. Lemma 6 implies that if a group non-member can sign on behalf of the group, then there is a probabilistic polynomial-time extractor to get e^*, e_1^*, e_2^* such that $e^* = e_1^*e_2^*$ (in $\mathbb{Z}$). This and Theorem 1 imply that $gcd(e^*, e) > 1$ for some accumulated value $e \in \mathfrak{A}$. So we can construct a polynomial-time simulator that is able to factor a challenge product of two large primes.
- Anonymity. Recall that the underlying interactive proof protocol is statistically zero-knowledge, and that both the ElGamal and the CGHN cryptosystems are semantically secure. If there is a polynomial-time adversary that is able to identify any signature generated by an honest group member, then we can construct a polynomial-time simulator to break the assumptions underlying the semantic security of the public key cryptosystems.
- Unlinkability. It is computationally infeasible for a polynomial-time adversary to decide whether two signatures, corresponding to two pairs of ciphertexts $(\delta_1; \alpha_1, \beta_1)$ and $(\delta_2; \alpha_2, \beta_2)$, are generated by the same group member. This is so because, in the definition of semantic security, it does not matter whether the adversary (i.e., distinguisher) is allowed to ask one or multiple challenge ciphertexts [BDJR97].
- No-Framing. This is proved in Lemma 7.

– Traceability. Recall that Lemma 6 ensures that $w^e = v \bmod n$, that (α, β) is the ElGamal ciphertext of w, and that δ is the CGHN encryption of e. Similar to the reasoning for unforgeability, we can show that Traceability holds. We stress that the verifiable ElGmal decryption, which is ensured by the soundness of protocol proving $log_{g_1} y_1 = log_{\alpha}\beta/w$, is crucial.
– Coalition-resistance. Suppose there is a successful coalition that cannot be traced back to any colluding member. Consider the worst case where all the group members collude. Then, we get a pair (w^*, e^*) such that $(w^*)^{e^*} = v \bmod n$, that $e^* \notin \mathfrak{A}$, and that $gcd(e^*, e) = 1$ or e for any $e \in \mathfrak{A}$. Note that the range condition ensures that $e^* < A^3 < A^4 < e_i e_j$ for any $e_i, e_j \in \mathfrak{A}$. On the other hand, theorem 1 shows that $e^* | \prod_{\forall e \in \mathfrak{A}} e$. This is a contradiction.

C Proving Knowledge in Group(s) of Unknown Order(s)

This is a 3-move honest-verifier zero-knowledge protocol. The corresponding non-interactive scheme is secure in the random oracle model [FS86,BR93], and can be naturally turned into a *signatures of knowledge* scheme [CS97].

Given a cyclic group $\mathbb{G} = \langle g \rangle$ such that $ord(\mathbb{G})$ is unknown but $|ord(\mathbb{G})| = l_{\mathbb{G}}$ (i.e., $2^{l_{\mathbb{G}}-1} < ord(\mathbb{G}) \leq 2^{l_{\mathbb{G}}}$) is publicly known. Fujisaki and Okamoto [FO97] proved that under SRSA, the standard proofs of knowledge protocols that work for a group of known order also work for a group of unknown order. For $y \in \mathbb{G}$, we denote by $x = log_g y$ the discrete logarithm of y to base g as an integer $x \in \mathbb{Z}$. We assume a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ which maps a binary string of arbitrary length to a k-bit hash value. Let $\epsilon > 1$ be a security parameter.

Proving Knowledge of a DLOG in a Group of Unknown Order. This is in parallel to [S91].

Definition 8. *Let $g, y \in \mathbb{G}$. A pair $(c, s) \in \{0,1\}^k \times \{-2^{\epsilon(l_{\mathbb{G}}+k)+1}, \cdots, 2^{\epsilon(l_{\mathbb{G}}+k)+1}\}$ such that $c = \mathcal{H}(g, y, g^s y^c, m)$ is a signature on message $m \in \{0,1\}^*$ with respect to the knowledge of discrete logarithm $x = log_g y \in \{0,1\}^{l_{\mathbb{G}}}$.*

The prover (or signer) knowing x can generate such a signature by choosing $r \in_R \pm\{0,1\}^{\epsilon(l_{\mathbb{G}}+k)}$ and computing

$$c = \mathcal{H}(g, y, g^r, m) \text{ and } s = r - cx \text{ (in } \mathbb{Z}).$$

Lemma 8. ([CM98]). *Assume* SRSA *holds. Then, the interactive protocol corresponding to Definition 8 is an honest-verifier statistical zero-knowledge proof of knowledge of $x = log_g y$.*

Proving That a DLOG Lies in an Interval. This protocol is related to the protocols originally presented in [FO97,CFT98].

Definition 9. *Suppose $g, y \in \mathbb{G}$. A pair $(c, s) \in \{0,1\} \times \{-2^{\epsilon(l_2+k)+1}, \cdots, 2^{\epsilon(l_2+k)+1}\}$ such that $c = \mathcal{H}(g, y, g^{s-cX} y^c, m)$ is a signature on message $m \in \{0,1\}^*$ with respect to the discrete logarithm of knowledge $x = log_g y$ such that $x \in \{X - 2^{\epsilon(l_2+k)+2}, X + 2^{\epsilon(l_2+k)+2}\}$.*

The prover knowing $x \in \{X - 2^{l_2}, \cdots, X + 2^{l_2}\}$ can generate such a signature by choosing $r \in_R \pm\{0,1\}^{\epsilon(l_2+k)}$ and computing

$$c = \mathcal{H}(g, y, g^r, m) \text{ and } s = r - c(x - X) \text{ (in } \mathbb{Z}).$$

Lemma 9. ([CM98]). *Assume* SRSA *holds. Then, the interactive protocol corresponding to Definition 9 is an honest-verifier statistical zero-knowledge proof of knowledge $x = log_g y$ such that $x \in \{X - 2^{\epsilon(l_2+k)+2}, \cdots, X + 2^{\epsilon(l_2+k)+2}\}$.*

Proving Equality of Two Discrete Logarithms in a Group of Unknown Order. This is in parallel to [CP92]. Suppose $g_1, g_2, y_1, y_2 \in \mathbb{G} = \langle g \rangle$.

Definition 10. *A pair* $(c, s) \in \{0,1\}^k \times \{-2^{\epsilon(l_{\mathbb{G}}+k)+1}, \cdots, 2^{\epsilon(l_{\mathbb{G}}+k)+1}\}$ *such that* $c = \mathcal{H}(g_1, g_2, y_1, y_2, g_1^s y_1^c, g_2^s y_2^c, m)$ *is a signature on message* $m \in \{0,1\}^*$ *with respect to the knowledge of discrete logarithm* $x = log_{g_1} y_1 = log_{g_2} y_2 \in \{0,1\}^{l_{\mathbb{G}}}$.

The prover knowing x can generate such a signature by choosing $r \in_R \pm\{0,1\}^{\epsilon(l_{\mathbb{G}}+k)}$ and computing

$$c = \mathcal{H}(g_1, g_2, y_1, y_2, g_1^r, g_2^r, m) \text{ and } s = r - cx \text{ (in } \mathbb{Z}).$$

Proving Equality of Two Discrete Logarithms in Two Groups of Unknown Orders. This protocol is a natural extension of the protocol corresponding to Definition 10. Suppose $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$. Let $l_{\mathbb{G}} = min\{|ord(\mathbb{G}_1)|, |ord(\mathbb{G}_2)|\}$. Then the above protocol can be used to prove that $x_1 = x_2$, where $x_1 = log_{g_1} y_1$ and $x_2 = log_{g_2} y_2$.

Definition 11. *Let* l_1 *and* l_2 *denote lengths, where* $l_1 < l_{\mathbb{G}}$ *and* $\epsilon(l_2 + k) + 2 < l_1$. *A pair* $(c, s) \in \{0,1\}^k \times \{-2^{\epsilon(l_2+k)+1}, \cdots, 2^{\epsilon(l_2+k)+1}\}$ *such that* $c = \mathcal{H}(g_1, g_2, y_1, y_2, g_1^s y_1^c, g_2^s y_2^c, m)$ *is a signature on message* $m \in \{0,1\}^*$ *with respect to the knowledge of discrete logarithm* $x = log_{g_1} y_1 = log_{g_2} y_2 \in \{0,1\}^{l_{\mathbb{G}}}$.