Aggregate Group Signature and Anonymous Delegation

Danfeng Yao and Roberto Tamassia

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-04-17
November 2004

Aggregate Group Signature and Anonymous Delegation

Danfeng Yao and Roberto Tamassia

Department of Computer Science
Brown University
Providence, RI 02912
{dyao, rt}@cs.brown.edu

Abstract. Group signatures are an effective solution for protecting the privacy of signers. When used in role-based delegation, they allow anonymous delegations by role members on behalf of a role. To make role-based delegation using group signatures efficient in a decentralized trust environment, we propose a group signature scheme that supports aggregation. Multiple group signatures can be aggregated into one compact signature of constant length. Our scheme is based on the ID-based group signature scheme by Chen, Zhang, and Kim [13] and the aggregate signature scheme by Boneh, Gentry, Lynn, and Shacham [8], and uses bilinear maps in gap Diffie-Hellman groups. Aggregate group signatures are suitable for financial applications where a large number of signatures are generated and the role (or group) membership of a signer is needed for verification. We show how aggregate group signatures can be used to implement an anonymous role-based cascaded delegation (RBCD) protocol that protects sensitive role-membership information of delegators. This protocol also yields a proxy signature scheme for groups.

Keywords: Group signatures, Aggregate signatures, Decentralized trust management, Proxy signatures

1 Introduction

In Role-Based Access Control (RBAC) systems [24], a user is assigned one or more roles by administrators, and privileges are associated with roles. Role members prove their memberships with public keys and digital credentials. Role-based delegation is a crucial concept in RBAC and in decentralized role-based trust management [14, 20], where resources can be shared with role members of organizations that are unknown to resource owners. A delegation credential is a digital certificate signed by a delegator on a statement that gives authorizations of certain access rights to delegatees. In some role-based delegation models [20, 25], a member E of role r is allowed to delegate privileges associated with r to other roles by issuing delegation credentials. To do this, E needs to reveal his role membership credential in order to prove that he is qualified to delegate the privileges.

However, role membership of a delegator is sensitive information. A delegator may not want to reveal his or her identity and role membership at each delegation transaction. There has been a significant amount of work on trust negotiation frameworks [27, 28], whose aim is to strategically control the release of sensitive credentials to unknown parties. In addition, organizations may want to hide their internal structures from the outside world.

To address these privacy concerns, we propose an *anonymous role-based delegation* protocol based on group signatures, in which a signature proves the membership of a signer without revealing the identity [12]. The anonymous signing feature of group signatures are particularly suitable for role-based delegation, because what is essential to verifying a delegation credential is the proof of delegator's role membership, rather than the identity. However, a practical concern about group signatures is its efficiency in a decentralized environment. Next, we introduce the technique of aggregate signatures and explain the need for an *aggregate group signature* scheme for implementing an efficient anonymous role-based delegation protocol.

1.1 Credential Efficiency and Aggregate Signatures

Lengthy digital credentials are inefficient to transmit and store. In decentralized trust management systems [20, 25], a delegation chain represents how trust or a delegated privilege is transferred from one user to another. It contains a sequence of delegation credentials, which connects unknown entities and resource owners. For example, a user with role *manager* at Central Bank issues a delegation credential C_1 to role *clerk* at a state bank, to authorize the right of signing e-coins. A member of role *clerk* at the state bank delegates this right to a role at a county bank by issuing another delegation credential C_2. Credential C_1 and C_2 form a delegation chain connecting the county bank role with Central Bank. The number of credentials required to authenticate a delegation chain is linear in the length of the chain. In distributed and pervasive computing environment where devices may have small storage units and limited bandwidth, credentials associated with a delegation chain need to be compact.

Aggregate signatures [8, 21] are an effective solution for shortening credential size. Namely, multiple signatures on different messages can be aggregated into one signature of constant size. The hierarchical certificate-based encryption scheme by Gentry [15], which is based on aggregate signatures of Boneh *et al.* [8], was used in [25] to implement a scalable role-based delegation protocol with an efficient representations of delegation credentials.

An interesting question is how to construct an aggregate *group* signature scheme so that it can be used to realize an efficient *anonymous* role-based delegation protocol. An aggregate group signature scheme not only has properties of a group signature scheme, such as anonymity, unlinkability, and revokability, but also is efficient. In organizations with hierarchical or streamlined management structures, an authorization of actions is based on a collection of permissions that can be in the form of aggregate group signatures. Although aggregate signature schemes [8, 21] and group signature schemes [6] are known, they do not

generalize to aggregate group signature schemes, and therefore are not suitable for implementing anonymous role-based delegation protocols.

In this paper, we present an aggregate group signature scheme and then use it to build an anonymous role-based delegation protocol. In the following, we first introduce group signature schemes. Then, we describe the role-based cascaded delegation (RBCD) protocol [25].

1.2 Group signatures

Group signatures, introduced by Chaum and van Heijst [12], allow members of a group to sign messages anonymously on behalf of the group. Only a designated group manager is able to identify the group member who issued a given signature. Furthermore, it is computational hard to decide whether two different signatures are issued by the same member. There are a number of group signature schemes proposed [1, 2, 6, 11, 12, 18, 19]. Efficient constructions [1, 10, 19, 26] were given based on the strong-RSA assumption [4]. Recently, bilinear pairing [7] is used to construct group signature schemes [6, 11, 13], whose security is based on variants of Diffie-Hellman assumptions. Our aggregate group signature scheme is based on the ID-based group signature by Chen, Zhang and Kim [13] and the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham [8]. We are able to achieve aggregate group signatures, and apply them to role-based cascaded delegation, the concept of which is introduced next.

1.3 Role-based Cascaded Delegation (RBCD)

RBCD [25] is a flexible and scalable role-based delegation framework. It comprises four operations: **Initiate**, **Extend**, **Prove**, and **Verify**. **Initiate** and **Extend** are used by a resource owner and an intermediate delegator, respectively, to delegate a privilege to a role. **Prove** is used by a requester to produce a proof of a delegation chain that connects the resource owner with the requester. **Verify** decides whether the requester is granted acess based on the proof.

In RBCD [25], delegation credentials include role membership certificates of each intermediate delegator, and delegation extension credentials that are proofs of delegation transactions signed by delegators. Credentials associated with a delegation chain are accumulated and transmitted to delegated role members at each delegation transaction. Therefore, for a delegation chain of length n, the number of certificates required to verify the delegation path is $2n$. An efficient RBCD protocol [25] with compact delegation credentials was realized based on the hierarchical certificate-based encryption scheme [15]. In this paper, we use our aggregate group signatures to extend the original RBCD protocol to support the anonymity of delegators.

1.4 Our Contributions

The contributions of this paper are summarized as follows. In order to construct aggregate group signatures, we present the first group signature scheme that supports aggregation. It allows multiple group signatures signed by different group

members on different messages to be aggregated into one short signature. Our scheme is based on the group signature scheme by Chen, Zhang, and Kim [13] and the aggregate signature technique by Boneh, Gentry, Lynn, and Shacham [8]. It uses bilinear maps in gap groups, where the computation Diffie-Hellman problem is hard and the decision Diffie-Hellman problem is easy. In our scheme, a group member E has a public and private key pair. In addition, E computes a set of secret signing keys from his private key. Then, the public information associated with these secret signing keys are certified by the group manager. The resulting certificates are called signing permits. To generate a group signature, the member E first signs with one of the secret signing keys, then the signature is aggregated with the corresponding signing permit. This gives a group signature from E.

Our group signature scheme supports signature aggregation and is efficient. The length of a group signature or a public key can be as short as 170 bits. The size of public keys needed for signature verification is constant. And group members can join and leave at any time, without requiring existing members to perform any update. The proposed scheme is compared with existing schemes in Table 1 (Section 6).

We then define and construct the first aggregate group signature scheme based on our group signatures. Aggregate group signatures are compact, and efficient to transmit and store. In this scheme, individual group signatures to be aggregated may be generated by members of different groups. Even if a signature is aggregated with other signatures, a group manager can trace the signer by showing a proof that cannot be denied by the signer. The security of our aggregate group signatures is based on the security of the aggregate signature scheme [8].

Aggregate group signatures are useful for on-line transactions that have high privacy and efficiency requirements, such as e-commerce. They are also important in decentralized trust management systems and especially in role-based cascaded delegation (RBCD) [25]. We describe how aggregate group signatures can be used to implement an anonymous and efficient role-based delegation protocol, where a delegator issues delegation credentials and proves role membership without disclosing the identity. Although anonymous RBCD can be realized using any group signature scheme, the implementation based on aggregate group signatures allows compression of delegation credentials and significantly improves efficiency.

Finally, we point out that anonymous role-based delegation implemented with aggregate group signatures gives rise to a proxy signature scheme for groups, which may be of separate interest. In this scheme, an original signer E serves as a delegator, and delegates the signing power to a certain group G of proxy signers by issuing a delegation certificate. Each of the proxy signers can sign anonymously on behalf of E, provided that the proxy is a valid group member. The anonymity can be revoked by the manager of group G. The signature from a proxy signer needs to demonstrate the group membership of the proxy, and that group G is authorized by the original signer. Proxy signature scheme for

groups is scalable, and is particularly suitable for role-based systems [24]. For example, Central Bank needs to delegate the signing power to all members of role *clerk* at a local bank. To do this, Central Bank just needs to generate one proxy signature for the role *clerk*, instead of issuing one for each role member. We omit the details of our proxy signature scheme for groups in this paper, as it can be easily derived from our anonymous RBCD protocol. This scheme can also be made into an aggregate proxy signature scheme for groups.

2 Preliminaries

Here, we describe the aggregate signature scheme [8] that is used to construct our signature schemes. The aggregate signature scheme by Boneh, Gentry, Lynn, and Shacham [8] supports aggregation of multiple signatures on distinct messages from distinct users into one signature. It uses bilinear maps [7] and works in any group where the decision Diffie-Hellman problem (DDH) is easy, but the computational Diffie-Hellman problem (CDH) is hard. Such groups are referred as gap groups [23] and are explained further in Section 3.1. The aggregate signature scheme comprises five algorithms: **KeyGen, Signing, Verification, Aggregation**, and **Aggregate Verification**. The first three are as in ordinary signature schemes; **Aggregation** merges multiple signatures into one signature of constant length; **Aggregate Verification** verifies aggregate signatures. Informally, the security of aggregate signature schemes is equivalent to the nonexistence of an adversary capable of existentially forging an aggregate signature [8]. Here, existential forgery means that the adversary attempts to forge an aggregate signature by some set of users, on messages of her choice. The formal proof of security defines an aggregate chosen-key security model, where the adversary is given a single public key, and her goal is the existential forgery of an aggregate signature. The adversary is given the power to choose all public keys except the challenge public key, and she is also given access to a signing oracle on the challenge key [8]. We refer readers to the aggregate signature paper of Boneh *et al.* [8] for further details.

3 Group Signature Scheme

In this section, we present our group signature scheme. First, we list the number theoretic assumptions needed in our scheme, and then describe the algorithms.

3.1 Assumptions

Similar to the aggregate signature scheme [8], our aggregate group signature scheme uses bilinear maps and works in gap groups [9, 23], which is explained next. Let G_1 and G_2 be two cyclic groups of some large prime order q. We write G_1 additively and G_2 multiplicatively.

Computation Diffie-Hellman (CDH) Problem: Given a randomly chosen $P \in G_1$, aP, and bP (for unknown randomly chosen $a, b \in \mathbb{Z}_q$), compute abP.

Decision Diffie-Hellman (CDH) Problem: Given a randomly chosen $P \in G_1$, aP, bP, and cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q$), decide whether $c = ab$. (If so, (P, aP, bP, cP) is called a valid Diffie-Hellman tuple.)

We call G_1 a GDH or gap group, if DDH problem can be solved in polynomial time but no probabilistic algorithm can solve CDH problem with non-negligible advantage within polynomial time [23]. As observed in the aggregate signature scheme [8], general gap groups are insufficient for constructing efficient aggregate signatures, therefore our scheme also makes use of bilinear maps. We refer the readers to papers by Boneh and Franklin [7] for examples and discussions of groups that admit such pairings.

Admissible pairings: Following Boneh and Franklin [7], we call $\hat{e}$ an admissible pairing if $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a map with the following properties: bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$ and all $a, b \in \mathbb{Z}$; non-degenerate: the map does not send all pairs in $G_1 \times G_1$ to the identity in G_2; computable: there is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

3.2 Definitions for Group Signature

A group signature scheme is consisted of the following algorithms: SETUP, JOIN, SIGN, VERIFY, and OPEN.

SETUP: On input a security parameter k, a probabilistic algorithm outputs the initial group public key $\mathcal{Y}$. Group members and manager also choose public/private keys.
JOIN: A protocol between the group manager and a user that results in the user becoming a new group member. The output of the user is membership certificates and membership secrets.
SIGN: An algorithm that on input a group public key, a membership secret, a membership certificate, and a message M outputs the group signature *Sig* of M.
VERIFY: An algorithm takes as inputs the group public key $\mathcal{Y}$, the group signature *Sig*, and the message M. Output is 1 or 0.
OPEN: The deterministic algorithm takes as inputs the message M, the signature *Sig*, and group manager's secret information to return the identity of the signer.

A secure group signature scheme must satisfy the following properties:

Correctness: Signatures produced by a group member using SIGN must be accepted by VERIFY.
Unforgeability: Only group members can sign messages on behalf of the group.
Anonymity: Given a valid signature, it is computationally hard to identify the signer for anyone except the group manager.
Unlinkability: Deciding whether two different valid signatures were computed by the same group member is computationally hard for anyone except the group manager.
Traceability: The group manager is always able to open a valid signature and identify the signer.
Exculpability: Even if the group manager and members collude, they cannot sign on behalf of a non-involved member.

Coalition-resistance: A colluding subset of group members cannot produce a valid signature that the group manager cannot open.

3.3 Construction of A Group Signature Scheme

The construction of our group signature scheme is based on the pairing-based aggregate signature scheme [8] and group signature scheme by Chen *et al.* [13]. It comprises five algorithms: SETUP, JOIN, SIGN, VERIFY, and OPEN. The **Signing**, **Verification**, and **Aggregate Verification** algorithms of Boneh *et al.*'s aggregate signature scheme [8] are used in constructing our scheme.

SETUP: This operation outputs the system parameters and public/private keys of users that will be used in the system.

- Root of system chooses a set of public parameters $params = (G_1, G_2, \hat{e}, \pi, H)$, where G_1, G_2 are groups of a large prime order q, G_1 is a gap group, $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear map, π is a generator of G_1, and $H : \{0,1\}^* \rightarrow G_1$ is a collision-resistant hash function, viewed as a random oracle.
- Each group member chooses a secret s_u as his private key, and computes the product $s_u\pi$ as its public key P_u. Similarly, the group manager chooses his secret key s_A, and computes the public key $P_A = s_A\pi$.

JOIN: A group member E obtains two types of certificates from the group manager. One is *long-term group membership certificate*, which certifies E's public key information. The other is *one-time signing permit*, which certifies E's one-time signing key information. The latter is used for issuing group signatures. The details of obtaining the certificates are as follows.

1. A group member E obtains a long-term group membership certificate *Cert* from the group manager. The manager runs algorithm **Signing** of aggregate signature scheme [8] on inputs s_A and statement $groupinfo \| s_u\pi$. **Signing** returns the product $s_A H(groupinfo \| s_u\pi)$, where s_A is the private key of the group manager, $s_u\pi$ is E's public key, and *groupinfo* contains information such as group name and membership expiration date. *Cert* is given to E.
2. A group member E also obtains one-time signing permits from the group manager. E randomly chooses a number of secrets $x_1, \ldots, x_l$, and computes one-time signing secret keys $x_1\pi, \ldots, x_l\pi$ and one-time signing public keys $s_u x_1\pi, \ldots, s_u x_l\pi$. Keys $s_u\pi, x_i\pi$, and $s_u x_i\pi$ are sent to the group manager, for all $i \in [1, l]$. E also sends *Cert* to the group manager.
3. The group manager first checks if E with public key $s_u\pi$ is a valid group member. This is done by verifying the signature *Cert* using algorithm **Verification** of aggregate signature scheme [8] with the group manager's public key P_A and statement $groupinfo \| s_u\pi$. The protocol terminates if $\hat{e}(Cert, \pi) \neq \hat{e}(P_A, H(groupinfo \| s_u\pi))$.
   Then the group manager tests if $e(s_u x_i\pi, \pi) = e(s_u\pi, x_i\pi)$ for all $i \in [1, l]$. If the test fails, the protocol terminates. Otherwise, the group manager runs algorithm **Signing** of aggregate signature scheme on inputs s_A and strings $groupinfo \| s_u x_i\pi$ to obtain $S_i = s_A H(groupinfo \| s_u x_i\pi)$ for all $i \in [1, l]$. S_i

is an one-time signing permit and is given to E. The group manager adds $(s_u\pi, x_i\pi, s_u x_i\pi)$ to its record for all $i \in [1, l]$.

Note that after E joins the group, step 2 and 3 can be run to generate more one-time signing permits, and E can obtain multiple such certificates each time.

SIGN: A group member E first produces a signature S_u on a message M on behalf of the group, by running algorithm **Signing** of aggregate signature scheme [8] on inputs $s_u x_i$ and M, where $s_u x_i$ is one of his one-time signing secrets. Then, E calls algorithm **Aggregation** of aggregate signature scheme to merge signature S_u with his one-time signing permit S_i associated with the secret $s_u x_i$. This gives the group signature, which is returned with the public key P_A of the group manager and the key $s_u x_i \pi$. The details are as follows.

- E runs algorithm **Signing** of aggregate signature scheme with secret key $s_u x_i$ and message M, and obtains a signature $S_u = s_u x_i H(M)$.
- E aggregates the signature S_u with one-time signing permit S_i associated with secret $s_u x_i$. This is done by running **Aggregation** of aggregate signature scheme, which returns a signature $S_g = S_u + S_i$. Recall that $S_i = s_A H(\mathit{groupinfo} \| s_u x_i \pi)$. S_g is output as the group signature. E also outputs public key P_A and one-time signing public key $s_u x_i \pi$.

VERIFY: This algorithm calls algorithm **Aggregate Verification** of aggregate signature scheme [8] with the following inputs: a group signature S_g, and public keys P_A and $s_u x_i \pi$ needed to verify the signature. The algorithm computes hash digest $H(M)$ of message M and the hash digest $h = H(\mathit{groupinfo} \| s_u x_i \pi)$ of one-time signing permit. S_g is accepted if $\hat{e}(S_g, \pi) = \hat{e}(P_A, h)\hat{e}(s_u x_i \pi, H(M))$; rejected if otherwise.

OPEN: Given a group signature S_g and its associated public information P_A and $s_u x_i \pi$, a group manager first verifies S_g. If S_g is valid, the group manager can identify a group member's public key $s_u \pi$ from the $s_u x_i \pi$ value, by consulting the group record. Furthermore, the group member cannot deny his signature because the group manager can provide a proof that the signature is associated with the member's public key $s_u \pi$ by showing: $\hat{e}(s_u x_i \pi, \pi) = \hat{e}(s_u \pi, x_i \pi)$. Similar to what is shown in the group signature scheme of Chen *et al.* [13], the group manager cannot misattribute a signature to frame the member unless he can compute $b\pi$ given q, $a\pi$, and $c\pi$ that satisfies: $a \equiv bc \mod q$. This problem was shown to be equivalent to CDH problem [13] in group G_1, and is called the reversion of computation Diffie-Hellman (RCDH) Problem. Therefore, it is impossible for the group manager to forge a signature to frame a group member. The security of our scheme is further analyzed in Section 3.4.

3.4 Security of Group Signature Scheme

We analyze the security of our group signature scheme by proving the following theorems. The first theorem states it is infeasible for an adversary to forge either a long-term membership certificate, an one-time signing permit, or a group

signature. The second theorem shows that in our group signature scheme, a verifier can verify the membership of a signer without gaining any other knowledge about the signer. Finally, the last theorem shows that our group signature scheme satisfies the properties listed in Section 3.2.

Theorem 1. *Our group signature scheme is as secure as the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham against existential forgery attacks.*

Proof. Long-term group membership certificates, one-time signing permits, and members' signatures using one-time secret signing keys, are generated by running algorithm **Signing** of the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham. Therefore, if an adversary can forge any of these signatures, she can also forge signatures in the aggregate signature scheme of Boneh *et al.* Note that a signature computed with one-time secret signing key is in the form of $sxH(M)$, rather than $sH(M)$ as in the aggregate signature scheme [8]. It can be easily shown that if an adversary can forge a signature in a form of $sxH(M)$, then she can forge a signature in the form of $sH(M)$ (by setting x to 1 in the reduction). This is omitted here.

Furthermore, our group signature is computed by running algorithm **Aggregate** of the aggregate signature scheme with an one-time signing permit and a member's signature. Therefore, we conclude that our group signature scheme is as secure as the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham against existential forgery attacks.

Theorem 2. *A valid group signature in our scheme contains a proof of group membership without revealing the identity of the signer.*

Proof. A valid group signature S_g is aggregated from an one-time signing permit of a group member E and E's signature using the corresponding one-time signing key. That is $S_g = S_i + S_u$, where $S_i = s_A H(groupinfo \| s_u x_i \pi)$ and $S_u = s_u x_i H(M)$. Because of the definition of aggregate signatures [8], a valid signature S_g implies that S_i is valid. This proves that the holder of key $s_u x_i \pi$ is a certified member of the group. A valid S_g also means that S_u is valid, therefore S_u is generated with the secret key $s_u x_i$. Thus, S_g contains a proof of group membership. Because the signing key $s_u x_i$ is an one-time signing key, and x_i is chosen randomly by the group member, the identity of the signer is not revealed.

Next, we show that our scheme satisfies the security requirements of an aggregate group signature scheme, which is captured in the Section 3.2.

Theorem 3. *Our group signature scheme from bilinear pairings in gap groups is a secure group signature scheme.*

Proof. Correctness: This is trivial. *Unforgeability:* This is proved in Theorem 1. *Anonymity:* This is proved in Theorem 2. *Unlinkability:* Given one-time secret signing key $sx_i\pi$ and $sx_j\pi$, it is computationally hard to decide whether they correspond to the same $s\pi$, without knowing $x_i\pi$ and $x_j\pi$. *Traceability* and *Exculpability:* These are shown in the description of OPEN algorithm in Section 3.3. *Coalition-resistance:* From Theorem 1 and 2 this can be deduced.

4 Aggregate Group Signature Scheme

In this section, we first define an aggregate group signature scheme, and then give a construction of this scheme based on our group signatures.

Definitions: An aggregate group signature scheme consists of six algorithms: SETUP, JOIN, SIGN, VERIFY, AGGREGATE, and OPEN. Algorithm AGGREGATE is defined below, and the rest of the algorithms have the same definitions as in the group signature scheme in Section 3.

AGGREGATE: This deterministic algorithm takes as inputs a number of group signatures and returns one aggregate signature.

Correctness, anonymity, unlinkability, traceability, exculpability, and *coalition-resistance* of aggregate group signature scheme are defined the same as in a group signature scheme in Section 3. Besides, an aggregate group signature scheme has stricter requirement for unforgeability, and a new requirement for aggregatability.

Unforgeability: Only group members can sign messages on behalf of the group. In particular, for an aggregate group signature S that is aggregated from n individual group signatures, even if an adversary knows $n-1$ of them, she cannot successfully forge S.

Aggregatability: Multiple signatures signed on different messages by different signers can be aggregated into one signature of constant length, and the aggregation can be performed by anyone.

4.1 Construction of An Aggregate Group Signature Scheme

We use our group signature scheme as a building block to construct an aggregate group signature scheme. The scheme also makes use of the **Aggregate Verification** algorithm of aggregate signature scheme [8].

SETUP: Same as in our group signature scheme.

JOIN: Same as in our group signature scheme.

SIGN: Same as in our group signature scheme.

AGGREGATE: Same as in algorithm **Aggregation** in the aggregation signature scheme [8]. It takes as inputs n number of group signatures Sig_k and corresponding values P_{A_k} and $s_{u_k}x_{i_k}\pi$ for all $k \in [1,n]$. Set $S_{Agg} = \sum_{k=1}^{n} Sig_k$. S_{Agg} is output as the aggregate group signature. The associated keys P_{A_k} and $s_{u_k}x_{i_k}\pi$ for $k \in [1,n]$ are also returned.

VERIFY: This algorithm calls algorithm **Aggregate Verification** of aggregate signature scheme [8] with the following inputs: an aggregate group signature S_{Agg}, public key P_{A_k}, and one-time signing public key $s_{u_k}x_{i_k}\pi$ for all $k \in [1,n]$.

- For $1 \le k \le n$, compute the hash digest $H(M_k)$ of message M_k and $h_k = H(groupinfo_k \| s_{u_k}x_{i_k}\pi)$ of the statement on one-time signing permit.
- S_{Agg} is accepted, if $\hat{e}(S_{Agg}, \pi) = \Pi_{k=1}^{n}\hat{e}(P_{A_k}, h_k)\hat{e}(s_{u_k}x_{i_k}\pi, H(M_k))$; rejected if otherwise.

OPEN: Given an aggregate group signature S_{Agg} and its associated public information including P_{A_k} and $s_{u_k} x_{i_k} \pi$ for $k \in [1, n]$, a group manager first verifies S_{Agg}. If it is valid, a group manager can easily identify a group member's public key $s_{u_k}\pi$ from $s_{u_k} x_{i_k} \pi$, by consulting the group record. Furthermore, the group member cannot deny his signature because the group manager can provide a proof that the signature is associated with the member's public key, as in our group signature scheme.

Theorem 4. *Our aggregate group signature scheme is as secure as the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham against existential forgery attacks.*

Proof. Individual group signatures are shown in Theorem 1 to be as secure as the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham against existential forgery attacks. In our aggregate group signature scheme, group signatures are aggregated using **Aggregation** algorithm of aggregate signature scheme of Boneh *et al.* Therefore, we conclude that the aggregate group signature scheme is as secure as the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham against existential forgery attacks.

Theorem 5. *Our aggregate group signature scheme from bilinear pairings in gap groups is a secure aggregate group signature scheme.*

Proof. Correctness: This is trivial. *Unforgeability:* This is proved in Theorem 4. *Anonymity:* This is proved in Theorem 2. *Unlinkability:* Same as in the proof of Theorem 3. *Traceability* and *Exculpability:* These are shown in the description of OPEN algorithm in Section 4.1. *Coalition-resistance:* From Theorem 4 and 2 this can be deduced. *Aggregatability:* This is trivial.

Next, we describe how aggregate group signatures are used to implement an anonymous role-based cascaded delegation protocol, which leads to a proxy signature scheme for groups.

5 Anonymous Role-Based Cascaded Delegation Protocol

We first define anonymous role-based cascaded delegation (RBCD) and then describe how it can be implemented using aggregate group signatures. Similar to definitions in RBCD [25], a privilege represents a role membership or a permission for an action such as accessing a database. A role defines a group of entities who are members of this role. Role members are managed by a role administrator. In this protocol, a role administrator is equivalent to a group manager in aggregate group signature scheme.

5.1 Definitions for Anonymous RBCD

An anonymous RBCD protocol defines five operations: **Initiate**, **Extend**, **Prove**, **Verify**, and **Open**.

Initiate: Same as in RBCD protocol [25], this operation is run by a resource owner to delegate a privilege to a role. It initiates a delegation chain for the privilege. The delegation certificate is signed using the private key of the resource owner on a statement, which includes the delegated privilege, the name of the role, and the public key of the role administrator. This operation can also be used by a role administrator to generate long-term role certificates, by replacing the public key of a role administrator with the public key of a role member in the statement.
Extend: This operation is similar to **Initiate**, but is run by an intermediate delegator E, who is a member of a role that is delegated a privilege by credential C. The goal is to generate a credential C' that extends the privilege to members of another role r. Delegation credential C' includes information of the delegated privilege, the name of role r, and the public key of role r's administrator. In addition, C' also contains the delegation credential C that E received, and the proof of E's role membership. C' does not disclose the identity of E.

C' may simply be an accumulation of individual certificates. In comparison, our implementation using aggregate group signatures is more efficient.
Prove: A requester E with role r produces a proof, which authenticates the delegation chain connecting the resource owner with E. This includes a proof of E's role membership without disclosing the identity, and the delegation credential that delegates the requested privilege to role r.
Verify: This is performed by the resource owner to verify that a proof produced by a requester correctly authenticates the delegation chain of a privilege.
Open: Role administrator revokes the anonymity of a delegator by examining signatures on a delegation credential. The identity of the delegator is returned.

5.2 Protocol Implementation

We give our implementation of an anonymous RBCD protocol using aggregate group signatures. Compared to RBCD [25], an one-time signing secret key instead of the delegator's private key is used to sign a credential, and an one-time signing permit instead of a role credential is used to prove role membership.

Setup: SETUP algorithm of aggregate group signature scheme is run to allow the root of system to set up public parameters *params*, and individuals to choose public and private keys. Then, JOIN protocol is run between role members and the role administrator to set up role membership certificates and one-time signing permits. The role administrator also keeps a record of signing key information.
Initiate: A resource owner runs **Signing** of aggregate signature scheme [8] to sign a delegation credential that authorizes a certain privilege to a role.
Extend: To further delegate a privilege to a role r', a member E of role r first runs algorithm SIGN of aggregate group signature scheme to sign a delegation statement. The inputs to algorithm SIGN are *params*, E's one-time signing secret key $s_u x_i$, his one-time signing permit S_i corresponding to $s_u x_i$, and a delegation statement. SIGN returns a group signature *Sig*, and appends the public signing key $s_u x_i \pi$ to the delegation statement. Then, the delegator E calls AGGREGATE

of aggregate group signature with *Sig* and the signature on the delegation credential issued to role r. The aggregated signature S_{Agg} and delegation statements are returned as the delegation credential to role r'.
Prove: A requester E of role r first runs SIGN algorithm of aggregate group signature, which uses an one-time signing key to sign a random message T chosen by verifier. Then, E calls AGGREGATE to merge the output signature with the signature on the delegation credential issued to role r. The outputs are returned.
Verify: VERIFY algorithm of aggregate group signature is run to verify the aggregate signatures submitted by the requester. The request is granted if the signature is accepted, and rejected if otherwise.
Open: A role administrator runs algorithm OPEN of aggregate group signature with a delegation credential, a target signing key $s_u x_i \pi$, and the signing keys record. This returns the public key $s_u \pi$, which identifies the signer.

The security of the above protocol is directly based on the security of the aggregate group signature scheme. This implies that it is infeasible to forge any valid delegation credential even under collusion. Anonymous RBCD satisfies traceability and exculpability requirements, i.e. a role administrator can revoke the anonymity of a role member as an intermediate delegator, but cannot frame a role member.

Our implementation of anonymous RBCD supports delegator anonymity without affecting the performance. It has similar efficiency as in RBCD protocol [25]. The time required for signing and verification is the same as in RBCD. In anonymous RBCD, role administrators need to sign multiple one-time signing permits for role members, which is not required in RBCD protocol. Nevertheless, signing a single signature is quite fast (3.57 ms on a 1 GHz Pentium III, compared to 7.90 ms for a RSA signature with 1007-bit private key on the same machine [5]). As described in Section 1.4, the above protocol gives rise to a proxy signature scheme for groups. Due to page limit, details (definitions, construction, and proof of security) of the proxy signature scheme are omitted in this paper.

6 Discussion

Properties	**Proposed**	**ACJT** [1]	**BBS** [6]	**CZK** [13]
Assumption	CDHP	Strong RSA DDHP	SDHP	CDHP
Aggregatability	Yes	No	No	No
Length of signature*	510 bits	$>$ 3072 bits	1533 bits	850 bits
Number of certificates	Many	One	One	Many
Length of group public key	Constant	Constant	Constant	Constant
System	CA-based	CA-based	CA-based	ID-based

Table 1. Comparison of our group signature scheme and three previous schemes. Strong Diffie-Hellman (SDH) problem [6] is a variant of DH problem. *Length of signature with security approximately the same as 1024-bit RSA signature.

We compare properties of our group signature scheme with several existing group signature schemes in Table 1, and discuss in details one-time signing keys, length of signatures, and revocation issues in our schemes.

One-time signing keys Group signing keys and signing permits in our signature schemes are not reusable, similar to the group signature scheme of Chen, Zhang, and Kim [13]. However, group members can obtain multiple signing permits $S_1, \ldots, S_n$ at a time, by asking the group manager to certify multiple signing keys $s_u x_1 \pi, \ldots, s_u x_n \pi$. Therefore, communications between the group manager and members can be largely reduced. Similar concepts can be found in the trustee tokens scheme [17] and the secret handshakes protocol [3].

Signature length An anonymous RBCD protocol can be implemented using any group signature scheme. However, the implementation based on our aggregate group signatures is more efficient. Public keys and signatures (individual or aggregate) are single elements in G_1, and can be as short as 170 bits, with security similar to 1024 bits RSA signatures [9]. To verify a group signature, the public key of group manager and an one-time signing public key are required. Therefore, the size of information needed for verification is 510 bits, shorter than the short group signatures recently proposed [6]. For long delegation chains, our implementation provides aggregation ability that can significantly shorten the overall size of delegation credentials. Therefore, anonymous RBCD protocol using aggregate group signatures achieve efficient storage and transmission.

Revocation Group membership revocation before expiration can be handled by maintaining a revocation service, which can be efficiently achieved using authenticated dictionaries [16, 22]. Authenticated dictionary is a system for distributing data and supporting authenticated responses to queries. One-time signing public keys of revoked members are put on the repository of revocation service by group managers. Before verifying a group signature, the revocation service is queried to ensure that the signature is not generated with a revoked signing key.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — Crypto '00*, volume 1880 of *LNCS*, pages 255–270.
2. G. Ateniese, D. X. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *Financial Cryptography '02*, volume 2357 of *LNCS*, pages 183–197.
3. D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong. Secret handshakes from pairing-based key agreements. In *2003 IEEE Symposium on Security and Privacy*, pages 180–196.
4. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — Eurocrypt '97*, LNCS.
5. P. S. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology — Crypto '02*, LNCS.
6. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — Crypto '04*, LNCS.
7. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto '01*, volume 2139 of *LNCS*, pages 213–229.

8. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology — Eurocrypt '03*, volume 2656 of *LNCS*, pages 416–432.
9. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology — Asiacrypt '01*, volume 2248 of *LNCS*, pages 514–532.
10. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology — Crypto '02*, volume 2442 of *LNCS*, pages 61–76.
11. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — Crypto '04*, LNCS.
12. D. Chaum and E. van Heijst. Group signatures. In *Advances in Cryptology — Eurocrypt '91*, pages 257–265.
13. X. Chen, F. Zhang, and K. Kim. A new ID-based group signature scheme from bilinear pairings. In *Proceedings of WISA 2003*, pages 585–592, August 2003.
14. D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.
15. C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Advances in Cryptology — Eurocrypt '03*, volume 2656 of *LNCS*, pages 272–293.
16. M. T. Goodrich, M. Shin, R. Tamassia, and W. H. Winsborough. Authenticated dictionaries for fresh attribute credentials. In *Proc. Trust Management Conference*, volume 2692 of *LNCS*, pages 332–347, 2003.
17. A. Juels. Trustee tokens: Simple and practical tracing of anonymous digital cash. In *Financial Cryptography '99*, volume 1648 of *LNCS*, pages 33–43.
18. A. Kiayias and M. Yung. Extracting group signatures from traitor tracing schemes. In E. Biham, editor, *Advances in Cryptology – Eurocrypt '03*, volume 2656 of *LNCS*, pages 630–648.
19. A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, 2004.
20. N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, 2003.
21. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology — Eurocrypt '04*.
22. M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium*, pages 217–228, 1998.
23. T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *PKC '01*, volume 1992 of *LNCS*.
24. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29, Number 2:38–47, 1996.
25. R. Tamassia, D. Yao, and W. H. Winsborough. Role-based cascaded delegation. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, pages 146 – 155.
26. G. Tsudik and S. Xu. Accumulating composites and improved group signing. In *Advances in Cryptology — Asiacrypt '03*, volume 2894 of *LNCS*, pages 269–286.
27. W. Winsborough and N. Li. Safety in automated trust negotiation. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 147–160.
28. T. Yu, X. Ma, and M. Winslett. PRUNES: An efficient and complete strategy for automated trust negotiation over the internet. In *Proceedings of CCS*, 2000.