Anonymous Role-based Delegation With Group Signatures

Danfeng Yao
Computer Science Department
Brown University
Providence, RI 02912
dyao@cs.brown.edu

Roberto Tamassia
Computer Science Department
Brown University
Providence, RI 02912
rt@cs.brown.edu

Abstract

We propose a decentralized trust management model called anonymous role-based cascaded delegation. In this model, a delegator can issue authorizations on behalf of his role without revealing his own identity. Anonymous delegation protects sensitive membership information of a delegator and hides the internal structure of an organization. Certificates used in anonymous role-based cascaded delegation model can be signed using any type of group signature scheme. However, in ubiquitous computing, users may have mobile computing devices with narrow communication bandwidth and small storage units. To make credentials efficient to store and transmit, we construct a group signature scheme that supports aggregation. We explain why an aggregate group signature scheme cannot be trivially obtained using existing aggregate signature scheme and one-time signing keys. Improvements in the credential size brought by using signature aggregation are compared with existing group signature schemes.

Keywords: Role-based Access Control, Decentralized trust management, Delegation, Group signatures, Aggregate signatures

1 Introduction

In Role-Based Access Control (RBAC) systems [23], 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 [15, 19], 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 role-based delegation models [19, 24], a member E of role r is allowed to delegate privileges associated with r to other roles by issuing delegation credentials. In order to delegate, the member E has to show his role membership by revealing his role credential.

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. 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 [13]. The anonymous signing feature of group signatures is 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 their efficiency in a decentralized environment. Next, we introduce the technique of aggregate signatures and explain the need for an *aggregate group signature* scheme in an efficient anonymous role-based delegation protocol.

1.1 Credential Size and Aggregate Signatures

Lengthy digital credentials are inefficient to transmit and store. In decentralized trust management systems [19, 24], a delegation chain represents how trust or a delegated privilege is transferred from one user to another. The chain contains a sequence of delegation credentials that 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 State Bank to authorize the right of signing e-coins. A member of role *clerk* at State Bank delegates this right to a role at County Bank by issuing another delegation credential C_2. Credentials 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. Credentials associated with a delegation chain need to be compact, because mobile devices may have limited storage units and bandwidth.

Aggregate signatures [8, 20] are an effective solution for shortening credential size. Namely, multiple signatures on different messages can be aggregated into one signature of constant size. 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 revocability, but is also efficient.

In this paper, we present an efficient 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 [24].

1.2 Group signatures

Group signatures, introduced by Chaum and van Heijst [13], 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. A group signature scheme with constant-sized public keys was first given in [12], and followed by a number of improvements [1, 2, 6, 17, 18]. Until recently, group signature constructions [1, 10, 25] were mostly based on the strong-RSA assumption [4], and a group signature typically comprised of multiple elements of RSA-signature length. Recently, bilinear pairing [7] was used to construct group signature schemes [6, 11, 14, 21], whose security is based on variants of Diffie-Hellman assumptions. The group signature scheme by Boneh, Boyen, and Shacham [6] significantly shortens the signature length, compared to the RSA-based state-of-the-art group signature scheme by Ateniese *et al.* [1].

Existing group signatures do not support signature aggregation. A naive aggregate group signature scheme can be derived from the pairing-based aggregate signatures [8] and one-time signing keys. However, it fails to satisfy the exculpability or non-framing requirement of group signatures. Exculpability is that even if the group manager and members collude, they cannot sign on behalf of a non-involved member. In the naive scheme, a group member generates k public keys $(PK_1, \ldots, PK_k)$, by running k times the key generation algorithm of the aggregate signature scheme [8]. The group manager signs (with the group master secret) each of the public keys separately, and sends the k certificates [1] $Cert_1, \ldots, Cert_k$ back to the group member. To

[1] These certificates are issued by a group manager for proving group membership of a member. It is different from

produce a signature on message M, the group member signs M with the private key corresponding to PK_i $(1 \leq i \leq k)$ to create signature Sig as in the aggregate signature scheme, and sends $(M, Sig, PK_i, Cert_i)$ to the verifier. Group signature Sig can be aggregated with other signatures of this scheme as in the aggregate signature scheme [8].

However, the above scheme does not satisfy the exculpability (non-framing) requirement of a group signature, which is shown as follows. A group manager first runs the key generation algorithm of the aggregate signature scheme [8] to obtain a key pairs (PK, SK). He signs public key PK using the group master secret and generates a certificate $Cert$ for PK. The group manager can then sign a message with private key SK, and misattribute the signature to *any* group member. Unfortunately, the innocent group member does not have any proof for denying the signature.

In this paper, we present an efficient aggregate group signature scheme that solves the above problem. A group member generates a one-time signing key based on both a *long-term* private key of and a *short-term* secret. The signing keys are then certified by the group manager. The long-term public key of a group member is certified by a Certificate Authority (CA). Misattributing a signature to others is infeasible, even for the group manager, because a group member can prove that a signing key is not his and does not correspond to his CA-certified public key. The use of bilinear pairing allows us to achieve this property. Aggregate group signatures are useful in role-based cascaded delegation, which is introduced next.

1.3 Role-based Cascaded Delegation (RBCD)

Role-based Cascaded Delegation (RBCD) [24] 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 the RBCD protocol [24], 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 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 $2n$. 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.

1. In order to construct aggregate group signatures, we present the first group signature scheme that supports aggregation. Our scheme is based on 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 long-term 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

a CA certificate, which certifies the validity of a public key.

with one of the secret signing keys, then the signature is aggregated with the corresponding signing permit. This gives a group signature from E. The length of a group signature can be as short as 170 bits with security equivalent to a 1024-bit RSA signature. A group signature along with the public information needed for verification is only 510 bits or 64 bytes long. Group members can join and leave at any time, without requiring existing members to perform any update. Furthermore, we are able to solve the framing problem that exists in the naive group signature scheme described earlier. A group member cannot deny a signature if his anonymity is revoked by the group manager, and in the meantime, the group manager cannot misattribute a signature to any group member.

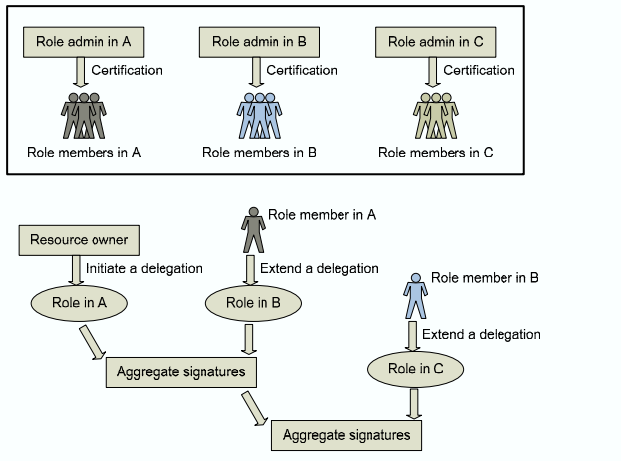

Figure 1: A schematic drawing of the RBCD protocol using aggregate signatures. Role administrators first certify role members' one-time signing keys, as shown in the upper rectangle. The delegation is initiated and extended sequentially, and signatures from these operations are aggregated.

2. We define and construct the first aggregate group signature scheme based on our group signatures. Individual group signatures that may be generated by members of different groups can be aggregated into one signature of constant length. 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]. For a delegation chain of length n, a delegation credential using our aggregate group signatures is twenty times shorter than the one using ACJT scheme [1], and five times shorter than the one generated in BBS scheme [6]. Because of one-time public keys, the asymptotic growth of our aggregate group signatures is still linear in the number of individual signatures. Nevertheless, the aggregatability along with short signatures and public keys can significantly reduce the length of multiple signatures. A discussion on the practical efficiency of the scheme is given in Section 6.
3. We describe how aggregate group signatures can be used to realize 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 with any group signature scheme, using aggregate group signatures allows the compression of delegation credentials and significantly improves efficiency. Delegation certificates in RBCD are issued to roles, rather than individual role members. For example, a privilege is delegated to role *doctor* at a hospital. Note that the RBCD protocol does *not* use aggregate group signature in a hierarchical fashion, where a group member of one group is the group manager of another group and so on. Instead, signatures to be aggregated are generated by group members (or role members) belonging to *independent* groups (or organizations), and role members have their signing keys certified independently by their role managers. The RBCD

protocol does *not* require a hierarchical generalization of aggregate group signatures, which involves the hierarchical certification of one-time signing keys. We illustrate our RBCD protocol using aggregate group signatures in Figure 1.

2 Preliminaries

Here, we describe the aggregate signature scheme [8] that is used to construct our signature schemes. The aggregate signature scheme by Boneh *et al.* [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 [22] 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, 22], 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 (DDH) 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 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 [22]. 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

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

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

As explained earlier, a naive group signature scheme based on one-time keys cannot satisfy the exculpability requirement. To overcome this problem, our signing keys are computed from the *long-term* private key of a signer. The corresponding long-term public key is certified by a Certificate Authority (CA). We prove that the non-framing property is achieved.

The construction of our group signature scheme is based on the pairing-based aggregate signature scheme [8] and the group signature scheme by Chen *et al.* [14]. 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 our scheme.

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

- The 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 $P_A = s_A\pi$. These are the *long-term* public keys, and are certified by a Certificate Authority (CA) using any signature scheme. The public key certificate of a member is used for repudiating misattributed signatures, and is different from the *one-time signing permits* below.

JOIN: A group member E obtains one or more *one-time signing permits* from the group manager. The permits certify E's one-time signing key information, and are used for issuing group signatures. The following shows how the signing permits are generated.

1. 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 $s_ux_1\pi, \ldots, s_ux_l\pi$. Keys $s_u\pi, x_i\pi$, and $s_ux_i\pi$ are sent to the group manager, for all $i \in [1, l]$. E also sends *Cert* to the group manager.
2. The group manager tests if $e(s_ux_i\pi, \pi) = e(s_u\pi, x_i\pi)$ 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 $groupinfo\|s_ux_i\pi$ to obtain $S_i = s_AH(groupinfo\|s_ux_i\pi)$ for all $i \in [1, l]$. S_i is an one-time signing permit and is given to E. The group manager adds $(s_u\pi, x_i\pi, s_ux_i\pi)$ to its record for all $i \in [1, l]$.

SIGN: A group member E first computes 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_ux_i and M, where s_ux_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_ux_i. This gives the group signature, which is returned with the public key P_A of the group manager and the key $s_ux_i\pi$. The details are as follows.

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

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 $s_ux_i\pi$ needed to verify the signature. The algorithm computes hash digest $H(M)$ of message M and the hash digest $h = H(groupinfo\|s_ux_i\pi)$ of one-time signing permit. S_g is accepted if $\hat{e}(S_g, \pi) = \hat{e}(P_A, h)\hat{e}(s_ux_i\pi, H(M))$; rejected if otherwise.

OPEN: Given a group signature S_g and its associated public information P_A and $s_ux_i\pi$, 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 $s_ux_i\pi$ 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: $\hat{e}(s_ux_i\pi, \pi) = \hat{e}(s_u\pi, x_i\pi)$. The exculpability requirement is satisfied. Intuitively, this is because (1) long-term public keys are certified by CA (2) signing keys are computed from long-term private keys (3) forging a valid signature with a correctly formed signing key is hard. We give the proof for exculpability in Theorem 3.

3.4 Security

We analyze the security of our group signature scheme by proving the following three theorems. The first theorem states it is infeasible for an adversary to forge either 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. 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). 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 both S_i and S_u are valid. This proves that the holder of key $s_u x_i \pi$ is a certified member of the group, and S_u is a valid signature signed with secret key $s_u x_i$. Thus, signature S_g contains a proof of group membership. Furthermore, 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:* This is shown in the description of OPEN algorithm in Section 3.3. *Exculpability:* In our proof, we assume that Certificate Authority (CA) does not collude with a group manager to misattribute a public key certificate to a group member. In a naive attack, a group manager or any collusion of members chooses random x^* and s^*, signs a message with $s^*x^*\pi$, and misattributes the signature to a group member. A group member with long-term public key $s_u\pi$ repudiates the signature by showing that the signing key does not correspond to $s_u\pi$, i.e. $\hat{e}(s^*x^*\pi, \pi) \neq \hat{e}(s_u\pi, x^*\pi)$. Furthermore, 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 [14] 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 with a correctly formed signing key. to frame a group member. *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.

4.1 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.

4.2 Construction

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, JOIN, and SIGN algorithms are the same as in our group signature scheme and are not repeated here.

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 (i.e. showing $\hat{e}(s_{u_k} x_{i_k} \pi, \pi) = \hat{e}(s_{u_k} \pi, x_{i_k} \pi)$) that the signature is associated with the member's public key.

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, 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.*

The proof of the Theorem 5 is similar to Theorem 3 in Section 3.4, and is not shown in this paper. Next, we describe how aggregate group signatures are used to implement an anonymous role-based cascaded delegation protocol.

5 Anonymous Role-Based Cascaded Delegation Protocol

We first define anonymous role-based cascaded delegation and then describe how it is realized using aggregate group signatures. Delegation credentials generated in our signature scheme are efficient to store and transmit, which is important in ubiquitous computing. Similar to definitions in the original RBCD protocol [24], 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. A role administrator in this protocol is equivalent to a group manager in the aggregate group signature scheme.

5.1 Definitions

An anonymous role-based cascaded delegation protocol defines five operations: **Initiate**, **Extend**, **Prove**, **Verify**, and **Open**.

Initiate: Same as in RBCD protocol [24], 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.

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 realization 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 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 Realization

We give an anonymous RBCD protocol using aggregate group signatures. Compared to the original RBCD protocol [24], 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 in aggregate group signature scheme is run to allow the root of system to set up public parameters *params*, and individuals to choose and certify long-term keys. Then, JOIN protocol is run between role members and the role administrator to set up one-time signing permits. The role administrator also keeps a record of signing key information.
Initiate: A resource owner runs the **Signing** algorithm 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. 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 public signing key $s_u x_i \pi$ to the delegation statement. Then, delegator E calls AGGREGATE of aggregate group signature with *Sig* and the signature on the delegation credential issued to role r. The resulting aggregated signature S_{Agg} and delegation statements are returned to r' as the delegation credential.
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 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. The anonymous RBCD protocol 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 realization of anonymous RBCD supports delegator anonymity without affecting the performance. It has similar efficiency as in the original RBCD protocol [24]. The time required for signing and verification is the same as in the original RBCD protocol [24]. In anonymous RBCD, role administrators need to sign multiple one-time signing permits for role members, which is not required in RBCD. Nevertheless, 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]).

6 Discussion

We compare our group signature scheme with several existing group signature schemes in Table 1.

Properties	**Proposed**	**ACJT** [1]	**BBS** [6]	**NS** [21] (2nd scheme)
Assumption	CDH	Strong RSA DDH	Strong DH	Strong DH DBDH, DL
Aggregatability	Yes	No	No	No
Length of signature*	170 bits	8696 bits [21]	1533 bits [6]	3072 bits [21]
Number of certificates	Many	One	One	One
Length of group public key	Constant	Constant	Constant	Constant
System	CA-based	CA-based	CA-based	CA-based

Table 1: Comparison of group signature schemes. Strong Diffie-Hellman (DH) and DBDH problems [6, 21] are variants of DH problem. *Length of signature with security equivalent to a 1024-bit RSA signature, and does not include public keys and parameters. The length of ACJT signature is from the instance given in [21].

For a delegation chain of length n, a delegation credential using our aggregate group signatures is twenty times shorter than the one using ACJT scheme [1], and five times shorter than the one generated in BBS scheme [6]. For a delegation chain of length twenty, the size of our credential is 1.4 KB, while the one in BBS scheme is 5.2 KB; for a 20 Kbits per second connection, our credential can be transmitted within 0.5 seconds, while the one using BBS takes 2.1 seconds. Note that this improvement is significant for small mobile devices with limited communication bandwidth and storage unit. For example, smart cards with a microprocessor typically have 32 KB EEPROM storage. Since RBCD protocols are aimed to be used for resource sharing in dynamic and distributed collaboration environment, users with small computing devices are not unusual. Therefore, our approach has the advantage over existing schemes, in terms of storage and transmission efficiency. The calculation of credential sizes is omitted due to the page limit.

One-time keys A major drawback of our aggregate group signature scheme is that group signing keys and signing permits in our signature schemes are not reusable. To reduce communications between the group manager and members, 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 $s_u x_1 \pi, \ldots, s_u x_n \pi$. Similar concepts can be found in the trustee tokens scheme [16] and the secret handshakes protocol [3]. A group manager needs to keep a file for storing one-time signing public keys. However, this does not significantly affect his performance, even though the number of group members is large. For example, for a group that has 100,000 members who obtain 100 one-time signing keys each year for ten years, the total storage space for all the one-time signing keys takes about 6.4 GB and can be easily stored on hard disks. Although file I/O in general can be relatively slow, appending new keys to the file is done off-line and does not affect the performance of users. If a database is used to maintain the keys, operations such as searching a signing key can be very fast as the keys can be indexed.

Therefore, we conclude that anonymous RBCD model using our signature scheme supports anonymity, exculpability (non-framing), and aggregation, without incurring significant overhead from the use of one-time signing keys.

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. Springer Verlag, 2000.

[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. IEEE Press, 2003.

[4] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer Verlag, 1997.

[5] P. S. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology — Crypto '02*, volume 2442 of *LNCS*, pages 354–368. Springer-Verlag, 2002.

[6] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — Crypto '04*, LNCS, 2004.

[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. Springer-Verlag, 2001.

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology — Eurocrypt '03*, pages 416–432, 2003.

[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. Springer-Verlag, 2001.

[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, 2002.

[11] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO '04*, 2004.

[12] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, volume 1296 of *LNCS*, pages 410–424. Springer-Verlag, 1997.

[13] D. Chaum and E. van Heijst. Group signatures. In *Advances in Cryptology — Eurocrypt '91*, pages 257–265. Springer-Verlag, 1991.

[14] X. Chen, F. Zhang, and K. Kim. A new ID-based group signature scheme from bilinear pairings. In K. Chae and M. Yung, editors, *Proceedings of International Workshop on Information Security Applications (WISA) 2003*, volume 2908 of *LNCS*, pages 585–592. Springer, August 2003.

[15] 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.

[16] A. Juels. Trustee tokens: Simple and practical tracing of anonymous digital cash. In *Financial Cryptography '99*, volume 1648 of *LNCS*, pages 33–43. Springer-Verlag, 1999.

[17] A. Kiayias and M. Yung. Extracting group signatures from traitor tracing schemes. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 630–648, Warsaw, Poland, 2003. Springer.

[18] A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/`.

[19] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, February 2003.

[20] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology — Eurocrypt '04*, volume 3027 of *LNCS*, pages 74–90. Springer-Verlag, 2004.

[21] L. Nguyen and R. Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *Advances in Cryptology – Asiacrypt '04*, volume 3329 of *LNCS*, pages 372–386. Springer, 2004.

[22] 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*, pages 104–118. Springer-Verlag, 2001.

[23] 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.

[24] 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. ACM Press, June 2004.

[25] G. Tsudik and S. Xu. Accumulating composites and improved group signing. In *Advances in Cryptology — Asiacrypt '03*, volume 2894 of *LNCS*, pages 269–286.