Cascaded Authorization with Anonymous-Signer Aggregate 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

May 16, 2006

Abstract

We introduce a decentralized trust management model called anonymous role-based cascaded delegation. In this model, a delegator can issue authorizations on behalf of her role without revealing her identity. This type of delegation protects the sensitive membership information of a delegator and hides the internal structure of an organization. To provide an efficient storage and transmission mechanism for credentials used in anonymous role-based cascaded delegation, we present a new signature scheme that supports both signer anonymity and signature aggregation. Our scheme has compact role signatures that make it especially suitable for ubiquitous computing environments, where users may have mobile computing devices with narrow communication bandwidth and small storage units.

Keywords: Delegation, anonymity, aggregate signature

1 Introduction

Authorization is an important concept of the resource sharing in open and collaborative environments such as Grid systems or the Internet. In role-based trust management [31, 39], privileges are associated with roles and each user is assigned one or more roles. Role members prove their memberships with public keys and digital credentials. Role-based delegation is important in decentralized role-based trust management for transferring privileges and sharing resources among role members that are initially unknown to each other. A delegation credential is a digital certificate signed by a delegator on a statement that gives authorizations to delegatees. In role-based delegation models [32, 39], 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.

For example, in a collaborative environment, hospital H delegates its role *guest* to the role *consultant* at company P in a credential C. John is a member of the role *consultant* at company P and has the corresponding role credential R. John may further delegate the role *guest* at H

*This work was supported in part by NSF grants CCF–0311510, CNS–0303577 and IIS–0324846.

to the role *professor* at university U in a credential C'. Credentials C, R, and C' constitute the delegation credential for the role *professor* at U, and prove the validity of the authorzation.

For privacy concerns, the identity of a user or an authorizer may be sensitive information in e-commerce, e-medicine, or peer-to-peer file-sharing (e.g., Kazaa) applications. An authorizor may not want to reveal his or her identity and role membership at each authorization or authentication. There has been a significant amount of work on trust negotiation frameworks [40, 43], 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, an *anonymous role-based delegation* protocol can be implemented with group signatures, in which a signature proves the membership of a signer without revealing the identity [21]. 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 the delegator's role membership, rather than the identity. A role-based delegation protocol implemented using group signature schemes is not only scalable due to the use of roles, but also has strong privacy protection provided by the group signature schemes.

A practical concern about group signatures is their efficiency in a distributed environment. Next, we introduce the technique of aggregate signatures and explain the need for a signature scheme that supports both anonymous signing and signature aggregation.

1.1 Credential size and aggregate signatures

Lengthy digital credentials are inefficient to transmit and store. In decentralized trust management systems [32, 39], 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. 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 [11, 33] 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 obtain an aggregate signature scheme that supports anonymous signing in role-based authorization. Desirable properties of such a signature scheme include anonymity, unlinkability, and revocability, and aggregation. In on-line banking applications for example, certain transaction can be approved only if it is signed sequentially by a member of the role *cashier*, a member of the role *accountant*, and a member of the role *manager*. Each signature can be generated without disclosing the signer's identity for privacy protection, and then be aggregated to existing ones.

Existing group signatures do not support signature aggregation. A naive group signature scheme supporting aggregation can be derived from the pairing-based aggregate signatures [11] and one-time signing keys. However, it does not have the exculpability or non-framing property. Exculpability means 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 [11]. 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 produce a signature on message

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

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 [11].

However, the above scheme cannot prevent the group manager from misattributing signatures. The group manager first runs the key generation algorithm of the aggregate signature scheme [11] 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. The innocent group member does not have any proof that can be used to deny the signature.

In this paper, we present an *anonymous-signer aggregate signature* scheme that supports both anonymity and aggregation by designing a special kind of cryptographic key pair. The unique feature of the key is that it contains the identity information of a role member, yet it is unlinkable. In other words, the seemingly random key used for signing cannot be forged by a role manager, as it contains the long term secret key of a role member, which is not known to the manager. We are able to achieve this by leveraging properties of a bilinear map, which was first used in the identity-based encryption scheme of Boneh and Franklin [10].

In the following, we briefly introduce the role-based cascaded delegation (RBCD) protocol [39, 42]. RBCD protocol is extended in this paper to support anonymity using the anonymous-signer aggregate signature scheme.

1.2 Role-based cascaded delegation

A large amount of work has been done on trust management and distributed authorization systems [2, 4, 7, 23, 31]. Among them, role-based cascaded delegation [39] is an efficient role-based trust management model that supports administrator-free delegation. Administrator-free delegation allows an individual role member to issue delegations without the participation of a role administrator. It enables flexible and dynamic authorization in a decentralized environment. 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 the access based on the proof.

In RBCD [39], a delegation credential includes 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 signature scheme to extend the original RBCD protocol to support the anonymity of delegators. The contributions of this paper are summarized next.

1.3 Our contributions

We present an anonymous-signer aggregate signature scheme. In our scheme, a role member E has a long-term public and private key pair. In addition, E computes a set of secret signing keys

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

from his private key. Then, the public information associated with these secret signing keys are certified by the role manager. The resulting certificates are called *signing permits*. To generate a role 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 the role signature of E. We introduce a key blinding mechanism that integrates the long-term private key of a signer with a random blinding factor. Using this special signing key, a role member cannot deny a signature when revoked anonymity; yet, the role manager cannot misattribute a signature to any role member. By leveraging signature aggregation [11], the length of a role signature can be as short as 170 bits with security equivalent to a 1024-bit RSA signature. A role signature along with the public information needed for verification is only 510 bits or 64 bytes long. Role members can join and leave at any time, without requiring existing members to perform any update.

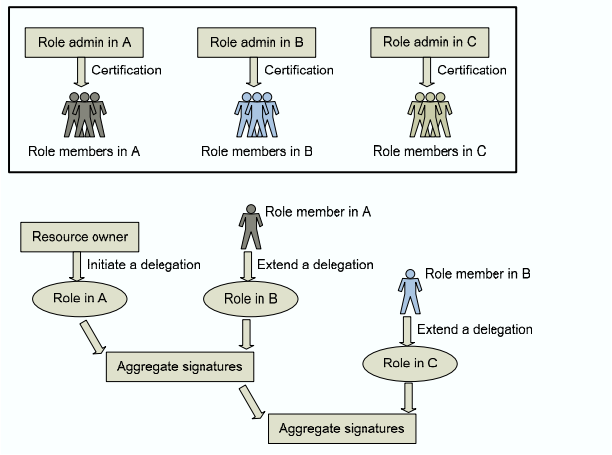

Figure 1: A schematic drawing of the RBCD protocol using aggregate signatures. Role administrators first certify role members' one-time signing keys, as shown in the upper rectangle. The delegation is initiated and extended sequentially, and signatures from these operations are aggregated.

In an anonymous-signer aggregate signature scheme, individual role signatures that may be generated by members of different roles can be aggregated into one signature of constant length. Even if a signature is aggregated with other signatures, a role manager can trace the signer by showing a proof. The security is based on the security of the aggregate signature scheme [11]. Because of one-time public keys, the asymptotic growth of our signatures is still linear in the number of individual signatures. Nevertheless, signature aggregation can significantly reduce the length of multiple signatures. A discussion on the efficiency of the scheme is given in Section 5.

We describe how anonymous-signer aggregate signatures can be used to realize an anonymous and efficient role-based authorization 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 our anonymous-signer aggregate signature scheme allows the compression of delegation credentials and significantly improves the efficiency. Delegation certificates in RBCD are issued to roles, rather than individual role members. For example, a privilege is delegated to the role *doctor* at a hospital.

Note that the RBCD protocol does *not* require a hierarchical generalization of our signature scheme, and does *not* require the (expensive) hierarchical certification of one-time signing keys. We illustrate the anonymous role-based authorization in Figure 1.

Finally, we point out that anonymous role-based delegation implemented with anonymous-signer aggregate 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 [37]. 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. The ability to aggregate multiple such proxy signatures into one make this scheme efficient in pervasive computing environment. 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.

1.4 Outline of the paper

In Section 2, we give an overview of the aggregate signature scheme by Boneh *et al.* [11]. The definition and construction of our anonymous-signer aggregate signature scheme are given in Section 3. In Section 4, we introduce the anonymous role-based cascaded delegation protocol. The analysis of the anonymous role-based cascaded delegation protocol is given in Section 5. Related work is described in Section 6.

2 Preliminaries

Here, we describe the aggregate signature scheme [11] that is used to construct our signature schemes. The aggregate signature scheme by Boneh, Gentry, Lynn, and Shacham (BGLS scheme for short) supports aggregation of multiple signatures on distinct messages from distinct users into one signature [11]. It uses bilinear maps [10] 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 [35] 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 algorithms are defined the same 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 [11]. 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 [11]. We refer readers to the paper of BGLS scheme [11] for further details.

Our anonymous-signer aggregate signature scheme is constructed based on the aggregate signature scheme [11]. We do not claim our scheme as a general group signature scheme, although it has the key properties of a group signature scheme. To distinguish from the naming conventions of group signatures, we use *role*, *role member*, *role manager*, and *role signature* in our scheme, which are equivalent to *group*, *group member*, *group manager*, and *group signature* in a group signature scheme, respectively. A role represents a number of individuals having certain attributes, each of

them being a role member. The role is administrated by the role manager. A role signature is a signature signed by a role member on behalf of a role.

3 Anonymous-signer aggregate signature scheme

We present our anonymous-signer aggregate 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 [11], our anonymous-signer aggregate signature scheme uses bilinear maps and works in gap groups [13, 35], 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 the DDH problem can be solved in polynomial time but no probabilistic algorithm can solve the CDH problem with non-negligible advantage within polynomial time. As observed in the aggregate signature scheme [11], 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 [10] for examples and discussions of groups that admit such pairings.
Admissible pairings: Following Boneh and Franklin [10], 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:

1. Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$ and all $a, b \in \mathbb{Z}$.
2. Non-degenerate: The map does not send all pairs in $G_1 \times G_1$ to the identity in G_2.
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

Admissible pairing has been used to construct a number of encryption and signature schemes [10], and most recently in a broadcast encryption scheme with short ciphertexts and private keys [12].

3.2 Definitions

An anonymous-signer aggregate signature scheme consists of Setup, Join, Sign, Aggregate, Verify, and Open algorithms.

Setup: On input a security parameter k, a probabilistic algorithm outputs the initial group public key $\mathcal{Y}$. Each entity (role manager and role member) also chooses his or her public/private keys.
Join: A protocol between the role manager and a user that results in the user becoming a new role member. The output of the user is membership certificates and membership secrets.
Aggregate: This deterministic algorithm takes as inputs a number of role signatures and returns one aggregate signature.

Sign: An algorithm that on input a role public key, a membership secret, a membership certificate, and a message M outputs the role signature *Sig* of M.
Verify: An algorithm takes as inputs the role public key $\mathcal{Y}$, the role signature *Sig*, and the message M. The output is 1 or 0.
Open: The deterministic algorithm takes as inputs the message M, the signature *Sig*, and role manager's secret information to return the identity of the signer.

A secure anonymous-signer aggregate signature scheme must satisfy the following properties:

Correctness: Signatures produced by a role member using Sign must be accepted by Verify.
Unforgeability: Only role members can sign messages on behalf of the role. In particular, for an anonymous-signer aggregate signature S that is aggregated from n individual role signatures, even if an adversary knows $n-1$ of them, she cannot successfully forge S.
Anonymity: Given a valid signature, it is computationally hard to identify the signer for anyone except the role manager.
Unlinkability: Deciding whether two different valid signatures were computed by the same role member is computationally hard for anyone except the role manager.
Traceability: The role manager is always able to open a valid signature and identify the signer.
Exculpability: Even if the role manager and members collude, they cannot sign on behalf of a non-involved member.
Coalition-resistance: A colluding subset of role members cannot produce a valid signature that the role manager cannot open.
Aggregation: 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.

3.3 Construction

A naive anonymous-signer aggregate signature scheme can be developed based on one-time keys and aggregate signature scheme. However, such a scheme does not satisfy the exculpability requirement. We overcome this problem by designing signing keys that are both unlikable and tied to the *long-term* private key of a signer. Although this sounds like an oxymoron, we are able to achieve this and prove all of the required properties of the signature scheme.

The construction of our signature scheme is based on the pairing-based aggregate signature scheme [11] and the group signature scheme by Chen *et al.* [22]. It comprises six algorithms: Setup, Join, Sign, Aggregate, Verify, and Open. The *Signing*, *Verification*, and *Aggregate Verification* algorithms of BGLS scheme [11] are used in our scheme. The notation of our anonymous-signer aggregate signature scheme is shown in Table 1. The last three items in Table 1 refer to the k-th signature in an (aggregate) signature aggregated from n ($k \leq n$) individual signatures.

Notation: For a role member u, s_u represents his private key, P_u represents his long-term public key, $K_{u,i}$ represents his i-th signing public key, and $X_{u,i}$ represents the corresponding i-th (public) signing factor. For a role manager A, s_A represents his private key, P_A represents his long-term public key. $S_{u,i}$ is the i-th signing permit generated by the role manager for member u. When referring to an aggregate signature, K_{u_k,i_k} represents the signing public key associated with the k-th signature in the aggregate signature. Similarly, P_{u_k} and X_{u_k,i_k} represent the long-term public key and the secret signing factor of the k-signer in an aggregate signature, respectively. See also Table 1.

Table 1: Notation for anonymous-signer aggregate signature scheme.

u	A role member
s_u	Private key of u
P_u	Long-term public key of u
$K_{u,i}$	i-th signing key
$X_{u,i}$	i-th public signing factor
$S_{u,i}$	i-th signing permit of u
A	A role manager
s_A	Private key of A
P_A	Long-term public key of A
K_{u_k,i_k}	The signing public key for the k-th signature
P_{u_k}	Long-term public key of k-th signer
X_{u_k,i_k}	Secret signing factor of k-th signer

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

- E randomly chooses a number of secrets $x_1, \ldots, x_l$, and computes one-time signing factors $X_{u,1} = x_1\pi, \ldots, X_{u,l} = x_l\pi$ and one-time signing public keys $K_{u,1} = s_u x_1\pi, \ldots, K_{u,l} = s_u x_l\pi$. Keys $P_u, X_{u,i}$, and $K_{u,i}$ are sent to the role manager, for all $i \in [1, l]$. E also sends *Cert* to the role manager.

- The role manager tests if $e(K_{u,i}, \pi) = e(P_u, X_{u,i})$ for all $i \in [1, l]$. Recall $K_{u,i} = s_u x_i \pi$, $P_u = s_u\pi$, and $X_{u,i} = x_i\pi$. If the test fails, the protocol terminates. Otherwise, the role manager runs algorithm *Signing* of BGLS scheme on inputs s_A and strings $roleinfo \| K_{u,i}$ to obtain $S_{u,i} = s_A H(roleinfo \| K_{u,i})$ for all $i \in [1, l]$. $S_{u,i}$ is the i-th *one-time signing permit* for E, and is given to E. The role manager adds tuples $(P_u, X_{u,i}, K_{u,i})$ to its record for all $i \in [1, l]$.

Sign: A role member E first computes a signature S_u on a message M on behalf of the role, by running algorithm *Signing* of BGLS scheme [11] 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 BGLS scheme to merge signature S_u

with his one-time signing permit $S_{u,i}$ associated with the secret $s_u x_i$. This gives the role signature, which is returned with the public key P_A of the role manager and the key $K_{u,i}$. The details are as follows.

- E runs algorithm *Signing* of BGLS 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 $S_{u,i}$ associated with secret $s_u x_i$. This is done by running *Aggregation* of BGLS scheme, which returns a signature $S_g = S_u + S_{u,i}$. Recall that $S_{u,i} = s_A H(roleinfo \| K_{u,i})$. S_g is output as the role signature. E also outputs public key P_A and one-time signing public key $K_{u,i}$.

Aggregate: Same as in algorithm *Aggregation* in the BGLS scheme [11]. It takes as inputs n number of role signatures S_{g_k} and corresponding values P_{A_k} and K_{u_k,i_k} for all $k \in [1,n]$. Set $S_{Agg} = \sum_{k=1}^{n} \mathrm{S}_{g_k}$. S_{Agg} is output as the anonymous-signer aggregate signature. The associated keys P_{A_k} and $K_{u_k,i_k} = s_{u_k} x_{i_k} \pi$ for $k \in [1,n]$ are also returned.

Note that k-th role manager's public key P_{A_k} for $k \in [1,n]$ does not need to be the same. In other words, signatures from roles of different organizations can be aggregated.

Verify: This algorithm calls algorithm *Aggregate Verification* of BGLS scheme [11] with the following inputs: an anonymous-signer aggregate signature S_{Agg}, public key P_{A_k}, and one-time signing public key K_{u_k,i_k} for all $k \in [1,n]$.

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

The correctness of the verification algorithm in our anonymous-signer aggregate signature scheme is shown as follows.

$$
\begin{aligned}
\hat{e}(S_{Agg}, \pi) &= \hat{e}(\sum_{k=1}^{n} S_{g_k}, \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(S_{g_k}, \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(S_{u_k} + S_{u_k,i_k}, \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(S_{u_k}, \pi) \hat{e}(S_{u_k,i_k}, \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(s_{u_k} x_{i_k} H(M_k), \pi) \hat{e}(s_{A_k} H(roleinfo_k \| K_{u_k,i_k}), \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(H(M_k), s_{u_k} x_{i_k} \pi) \hat{e}(h_k, s_{A_k} \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(H(M_k), K_{u_k,i_k}) \hat{e}(h_k, P_{A_k})
\end{aligned}
$$

Open: Given an anonymous-signer aggregate signature S_{Agg} and its associated public information including P_{A_k} and K_{u_k,i_k} for $k \in [1,n]$, a role manager first verifies signature S_{Agg}. If it is valid, a role manager can easily identify a role member's public key P_{u_k} from K_{u_k,i_k}, by consulting the role record. The role member cannot deny his signature because the role manager can provide a proof

(i.e. by showing $\hat{e}(K_{u_k,i_k}, \pi) = \hat{e}(P_{u_k}, X_{u_k,i_k})$) that the signature is associated with the member's public key. 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, and (3) forging a valid signature with a correctly formed signing key is hard.

Theorem 1 *The communication complexity of* Join *algorithm is $O(l)$, where l is the number of one-time signing keys certified. The computation complexity of* Verify *algorithm is $O(n)$, where n is the number of signatures aggregated.*

3.4 Security

Here we analyze the security of our anonymous-signer aggregate signature scheme. *Adversarial model:* In our scheme, a probabilistic polynomial-time adversary is allowed to do the following. (1) An adversary can intercept any signatures in both the aggregated and individual form; (2) an adversary can steal valid signing certificates of role members; and (3) an adversary can collude with the role manager and role members, and attempt to sign on behalf of a non-involving role member.

We analyze the security of our anonymous-signer aggregate signature scheme by proving the following three theorems. The first theorem states it is infeasible for an adversary to forge either a one-time signing permit or a role signature. The second theorem shows that in our 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 signature scheme satisfies the properties listed in Section 3.2.

Theorem 2 *Our anonymous-signer aggregate signature scheme is as secure as the BGLS aggregate signature scheme 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 BGLS scheme. Therefore, if an adversary can forge any of these signatures, she can also forge signatures in the BGLS scheme. 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 [11]. 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 role signature is computed by running algorithm *Aggregate* of the BGLS scheme with a one-time signing permit and a member's signature. Therefore, we conclude that our anonymous-signer aggregate signature scheme is as secure as the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham against existential forgery attacks.

Individual role signatures are shown in Theorem 2 to be as secure as the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham against existential forgery attacks. In our anonymous-signer aggregate signature scheme, role signatures are aggregated using the *Aggregation* algorithm of BGLS scheme. Therefore, the anonymous-signer aggregate signature scheme is as secure as the aggregate signature scheme of Boneh, Gentry, Lynn, and Shacham against existential forgery attacks. □

Theorem 3 *A valid anonymous-signer aggregate signature in our scheme contains proofs of role memberships without revealing the identities of signers.*

Proof: A valid role signature S_g is aggregated from a one-time signing permit of a role member E and E's signature using the corresponding one-time signing key. That is $S_g = S_u + S_{u,i}$, where $S_{u,i} = s_A H(roleinfo \| K_{u,i})$, $S_u = s_u x_i H(M)$, and $K_{u,i} = s_u x_i \pi$. Because of the definition of aggregate signatures [11], a valid signature S_g implies that both $S_{u,i}$ and S_u are valid. This proves that the holder of key $s_u x_i \pi$ is a certified member of the role, and S_u is a valid signature signed with the secret key $s_u x_i$. Thus, signature S_g contains a proof of role membership. Furthermore, because the signing key $s_u x_i$ is a one-time signing key and x_i is chosen randomly by the role member, the identity of the signer is not revealed. □

Theorem 4 *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups satisfies the following requirements: correctness, unforgeability, anonymity, unlinkability, traceability, coalition-resistance, and aggregation in the random oracle model under the CDH assumption.*

Proof: The proofs of correctness and aggregation are straightforward. *Unforgeability:* This is proved in Theorem 2. *Anonymity:* This is proved in Theorem 3. *Unlinkability:* Given one-time signing public key of a role member $K_{u,i} = s_u x_i \pi$ and $K_{u,j} = s_u x_j \pi$, it is computationally hard to decide whether they correspond to the same long-term public key $P_u = s_u \pi$, without knowing the random component $x_i \pi$ 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 role manager to misattribute a public key certificate to a role member. In a naive attack, a role manager or any collusion of members chooses random x^* and s^*, signs a message with key $X = s^* x^*$, and misattributes the signature to a role member. A role member with long-term public key P_u repudiates the signature by showing that the signing key does not correspond to P_u, i.e. $\hat{e}(s^* x^* \pi, \pi) \neq \hat{e}(P_u, x^* \pi)$. Furthermore, the role 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 the CDH problem [22] in group G_1, and is called the reversion of computation Diffie-Hellman (RCDH) Problem. Therefore, it is impossible for the role manager to forge a signature with a correctly formed signing key. to frame a role member. *Coalition-resistance:* From Theorem 2 and 3 this can be deduced. □

4 Anonymous role-based cascaded delegation protocol

We first define anonymous role-based cascaded delegation and then describe how it is realized using anonymous-signer aggregate 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 [39], a privilege represents a role membership or a permission for an action such as accessing a database. Anonymous role-based cascaded delegation allows any role member to authorize on behalf of the role without disclosing the individual identity. Recall that a role defines a group of entities having certain qualifications. Role members are managed by a role administrator, which is equivalent to a role manager in the anonymous-signer aggregate signature scheme.

4.1 Definitions

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

INITIATE: Same as in RBCD protocol [39], 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, credential 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.

Credential C' may simply be an accumulation of individual certificates. In comparison, our realization using anonymous-signer aggregate 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.

4.2 Realization

We give an anonymous RBCD protocol using anonymous-signer aggregate signatures. Compared to the original RBCD protocol [39], a one-time signing secret key instead of the delegator's private key is used to sign a credential, and a one-time signing permit instead of a role credential is used to prove role membership.

SETUP: Setup in anonymous-signer aggregate 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 BGLS aggregate signature scheme [11] 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 anonymous-signer aggregate 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 role signature *Sig*, and appends public signing key $s_u x_i \pi$ to the delegation statement. Then, delegator E calls Aggregate of anonymous-signer aggregate signature with *Sig* and the signature on the delegation credential issued to role r. The resulting aggregate 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 anonymous-signer aggregate signature, which uses a 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 anonymous-signer aggregate signature is run to verify the aggregate signatures submitted by the requester. The request is granted if the signature is accepted, and rejected if

Table 2. Comparison of signature schemes. l is the number of one-time keys.

Properties	**Proposed**	**ACJT** [3]	**BBS** [8]	**NS** [34] (2nd scheme)
Assumption	CDH	Strong RSA DDH	Strong DH	Strong DH DBDH, DL
Aggregate	Yes	No	No	No
Communications	$O(l)$	$O(1)$	$O(1)$	$O(1)$
Length of group/role public key	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Number of certificates	$O(l)$	1	1	1
Length of signature	170 bits	8696 bits	1533 bits	3072 bits

otherwise.

OPEN: A role administrator runs algorithm Open of anonymous-signer aggregate 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 anonymous-signer aggregate 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 [39]. The time required for signing and verification is the same as in the original RBCD protocol [39]. 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 [6]). As described in Section 1.3, 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.

5 Analysis

We compare our anonymous-signer aggregate signature scheme with several existing group signature schemes in Table 2. The strong Diffie-Hellman (DH) and DBDH problems are variants of the DH problem. In Table 2, the length of a 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 its paper [3]. Communications refer to the communication costs between a user and the manager to generate n grouprole signatures.

For a delegation chain of length n, a delegation credential using our anonymous-signer aggregate signatures can be twenty times shorter than the one using ACJT scheme [3], and five times shorter than the one generated in BBS scheme [8]. For a delegation chain of length twenty, the size of our credential is 1.4 KB, and 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, and the one using BBS takes 2.1 seconds.

In the anonymous RBCD protocol, a delegation credential generated by INITIATE operation contains a signature, delegator's public key, delegatee's role, the public key of delegatee's role administrator, and the delegated privilege. Similarly, we can derive the contents of a delegation

credential after $n-1$ extensions. Assume a role or privilege name is expressed in 100 bits and let security requirement equivalent to 1024-bit RSA signature. Using ACJT group signatures, the size of a credential of length n is at least $10944n$ bits. Using BBS group signatures, the size is $2073n$ bits. Using our signature scheme, the size is $540n + 170$. The improvement in credential size is more significant as the length of delegation chain increases.

One-time keys. A major drawback of our anonymous-signer aggregate signature scheme is that signing keys and signing permits are not reusable. To reduce communications between the role manager and members, role members can obtain multiple signing permits $S_1, \ldots, S_n$ at a time, by asking the role manager to certify multiple signing keys $K_{u,1}, \ldots, K_{u,n}$. Similar concepts can be found in the trustee tokens scheme [29] and the secret handshakes protocol [5]. A role 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 role members is large. For example, for a role 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.

Remark: Our anonymous RBCD protocol does *not* use the anonymous-signer aggregate signatures in a hierarchical fashion, where a role member in one organization is the role manager of another organization and so on. Instead, signatures to be aggregated are generated by role members belonging to *independent* roles (or organizations), and role members have their signing keys certified independently by their role managers.

Therefore, we conclude that the 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.

Finally, we would like to point out that there is a conceptual difference between the aggregate algorithm in BGLS scheme [11] and the one in this paper. In their aggregate signature scheme, a verifier is given a signature along with the identities of the parties involved and their respective messages. The verifier can obtain the public keys from CA, and thus in aggregate signatures, the size of public information is reduced. Instead, in our proposed scheme, the verifier can not obtain the one-time signing public keys from a certified directory.

Finally, we have used our signature scheme to construct an aggregate proxy signature scheme for groups, and we have discussed its applications.

6 Related Work

Our signature scheme has properties that are related to group signature schemes. Group signatures, introduced by Chaum and van Heijst [21], 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. In early group signature schemes [21], group public keys grew with the size of the group and were inefficient.

A group signature scheme with constant-sized public keys was first given in [17], and followed by a number of improvements [3, 8]. Until recently, group signature constructions (e.g., [3, 15]) were mostly based on the strong-RSA assumption, and a group signature typically comprised of multiple

elements of RSA-signature length. Recently, bilinear pairing [10] has been used to construct group signature schemes [8, 16, 22], whose security is based on variants of Diffie-Hellman assumptions. The group signature scheme by Boneh, Boyen, and Shacham [8] significantly shortens the signature length, compared to the RSA-based state-of-the-art group signature scheme by Ateniese *et al.* [3]. An identity-based group signature scheme was proposed by Chen, Zhang, and Kim [22], where a group signature cannot be forged even if the private key of a user is known by a third party (i.e., the Private Key Generator in the ID-based systems [10]).

There has been extensive research on access control models in the past decade [25, 36, 37]. The concept of role-based access control [25, 37] is widely deployed to improve the scalability and efficiency of management. Trust management models are developed for the authorization in distributed systems. A number of such systems have been proposed, for example KeyNote [7], delegation certificates [4], SPKI [23], Delegation Logic (DL) [30], proof-carrying authorization (PCA) [1], *RT* framework [31], and role-based cascaded delegation [39]. The anonymous role-based delegation protocol and implementation presented in this paper are privacy-enhancing techniques general for any role-based trust management systems.

Hidden credentials system [28] has been proposed to protect sensitive credentials and policies. The main idea of that paper is that when a signature derived from an identity based encryption scheme (IBE) [9, 24, 38] is used to sign a credential, the credential content can be used as a public encryption key such that the signature is the corresponding decryption key. Hidden credentials can be used in such a way that they are never shown to anyone, thus the sensitive credentials are protected. The Hidden Credentials system also protects sensitive policies by not specifying which credentials can be used to decrypt the encrypted resource. Bradshaw *et al.* [14] extended the hidden credentials system to support complex access policies expressed as monotonic Boolean functions. They applied a secret splitting system to conceal the structure of such policies. The extended hidden credentials system protects the structure of Bob's polices. Frikken *et al.* [27] give a scheme that hides both credentials and policies. Most recently, Frikken *et al.* [26] proposed a protocol that allows the client and the servers to have policies for their credentials (to mitigate probing attacks) and hide these policies and the credentials. The above-mentioned work is in the conventional access control settings, where the server and authorized clients have established trust when hidden credentials are issued. Our point-based trust management model is suitable for more general contexts where the server and clients are initially unknown to each other.

Anonymous credential systems have been developed [18, 19, 20] to allow anonymous, yet authenticated and accountable, transactions between users and service providers. These systems give a technique for protecting the users' privacy when conducting Internet transactions. Our work presented in this paper is for anonymous role-based authorization, and can potentially be used as an anonymous credential system, where a user authenticates herself to be a valid member of a role. This can be achieved by generating a role signature, which is verified by a resource owner (verifier). The disadvantage of such an anonymous credential system compared to the state-of-the-art is that the credential is only one-time use instead of multi-use.

References

[1] A. W. Appel and E. W. Felten. Proof-carrying authentication. In *ACM Conference on Computer and Communications Security*, pages 52–62, 1999.

[2] R. Aringhieri, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Assessing efficiency of trust management in peer-to-peer systems. In *1st International Workshop on Collaborative Peer-to-Peer Information Systems (COPS '05)*, 2005.

[3] 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.

[4] T. Aura. Distributed access-rights management with delegation certificates. In *Secure Internet Programming – Security Issues for Distributed and Mobile Objects*, volume 1603 of *LNCS*, pages 211–235. Springer, 1999.

[5] 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.

[6] 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.

[7] M. Blaze, J. Feigenbaum, and A. D. Keromytis. KeyNote: Trust management for public-key infrastructures. In *Proceedings of Security Protocols International Workshop*, 1998.

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

[9] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Proceedings of Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[10] 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.

[11] 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.

[12] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology – Crypto '05*, 2005.

[13] 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.

[14] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of 11th ACM Conference on Computer and Communications Security (CCS)*, Oct. 2004.

[15] 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.

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

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

[18] J. Camenisch and E. Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 21–30, 2002.

[19] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[20] D. Chaum and J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Proceedings of Advances in cryptology—CRYPTO '86*, pages 118–167, January 1987.

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

[22] 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.

[23] 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.

[24] C. Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA International Conference on Cryptography and Coding*, volume 2260, pages 360–363. Springer, Dec. 2001.

[25] D. Ferraiolo and R. Kuhn. Role-based access control. In *Proceedings of the 15th National Computer Security Conference*, 1992.

[26] K. Frikken, J. Li, and M. Atallah. Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS)*, 2006.

[27] K. B. Frikken, M. J. Atallah, and J. Li. Hidden access control policies with hidden credentials. In *Proceedings of the 3nd ACM Workshop on Privacy in the Electronic Society (WPES)*, Oct. 2004.

[28] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society (WPES)*, Oct. 2003.

[29] 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.

[30] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security (TISSEC)*, Feb. 2003. To appear.

[31] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[37] 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.

[38] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[39] 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.

[40] W. Winsborough and N. Li. Safety in automated trust negotiation. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 147–160. IEEE Press, May 2004.

[41] D. Yao and R. Tamassia. Cascaded authorization with anonymous-signer aggregate signatures. Brown University Technical Report. April 2006. `http://www.cs.brown.edu/people/dyao/anonymity-full.pdf`.

[42] D. Yao, R. Tamassia, and S. Proctor. On improving the performance of role-based cascaded delegation in ubiquitous computing. In *Proceedings of IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm '05)*, pages 157–168. IEEE Press, September 2005.

[43] T. Yu, X. Ma, and M. Winslett. PRUNES: An efficient and complete strategy for automated trust negotiation over the internet. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 210–219, November 2000.