Compact and Anonymous Role-Based Authorization Chain

DANFENG YAO
Computer Science Department
Brown University
Providence, RI 02912
dyao@cs.brown.edu

and
ROBERTO TAMASSIA
Computer Science Department
Brown University
Providence, RI 02912
rt@cs.brown.edu

We introduce a decentralized delegation 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 digital 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.

Categories and Subject Descriptors: D.4.6 [**Operating System**]: Security and Protection—*Access Controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms: Security, Algorithms

Additional Key Words and Phrases: Delegation, anonymity, aggregate signature

This work was supported in part by NSF grants CCF–0311510, CNS–0303577 and IIS–0324846. A preliminary version of this paper was published in the Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics Information Assurance Workshop (IAW '06) [Yao and Tamassia 2006].

1. INTRODUCTION

Authorization is an important concept of the resource sharing in open and collaborative environments such as Grid computing [Pallickara et al. 2006] or the Internet. In role-based trust management [Li et al. 2002; Tamassia et al. 2004], privileges are associated with roles and each user is assigned one or more roles. Role members prove their memberships with digital credentials and public-key signatures. 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 [Li et al. 2003; Tamassia et al. 2004], a privilege can be delegated to another role, and then any member of the role can pass that privilege onto other roles. Besides privileges, a role, which represents a collection of privileges, can be delegated as well. We first illustrate in Example 1 a simple multi-step delegation scenario that transfers rights among roles within one administrative domain. Then we show in Example 2 a more complex cross-domain role-based delegation.

EXAMPLE 1. *A hospital has roles* Doctor*,* Nurse*, and* Intern*. The hospital permits all doctors to access a medical storage room. Bob is a doctor and has a doctor role credential issued by the hospital. When Bob is out of town, he authorizes his nurses to access the storage room by issuing the nurses a delegation credential. Alice is Bob's nurse and has a nurse role credential. She has short-term interns who also need to access the storage room. Then Alice passes the access privilege onto her interns by creating another delegation credential. The two-step delegation chain gives the authorization to interns to access the storage room, which consists of the two delegation credentials and Bob and Alice's role credentials. The role credentials show the delegators have the proper roles to issue the delegation.*

Decentralized delegation is to transfer privileges across different administrative domains, which is important to facilitate information and resource sharing in a collaboration. We give a more complex cross-domain role-based delegation in Section 3.

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 authorizer 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 [Winsborough and Li 2004; Yu et al. 2000], 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 [Chaum and van Heijst 1991; Bellare et al. 2003]. The anonymous signing feature of group signatures is particularly suitable for role-based delegation, because what is essential for verifying a delegation credential is the proof of the delegator's role membership, rather than his or her identity. A role-based delegation protocol implemented using group sig-

nature 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 [Li et al. 2003; Tamassia et al. 2004], 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 [Boneh et al. 2003; Lysyanskaya et al. 2004] 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. 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. In this paper, we present an *anonymous-signer aggregate signature* scheme. that satisfies properties of unforgeability, anonymity, traceability, exculpability, unlinkability, collusion-resistance, correctness, and aggregation (See Section 4.3 for definitions). We achieve these properties by designing the signing key such that it is random, yet contains the long-term private key of a role member. Even a role manager cannot sign on behalf of a role member because the manager does not know the long-term private key of that user. 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 [Boneh and Franklin 2001b].

1.2 Our contributions

We present an anonymous-signer aggregate signature scheme. In our scheme, a role member u has a long-term public and private key pair. In addition, u computes a set of one-time secret signing keys from his private key. Then, the public information associated with these one-time signing keys are certified by the role manager. A role manager maintains the role by processing newly joined members and opening signatures (revoking the anonymity of signers) as necessary. The resulting certificates are (one-time) *signing permits*. To sign on behalf of a role, a member u first signs with one of the secret signing keys, then that signature is aggregated with the corresponding signing permit. This operation creates a *role signature* in which the signer is anonymous but can be proven to be a member of a role. We introduce a

simple yet effective 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 [Boneh et al. 2003], 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.

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 and show the proof. The security is based on the security of the aggregate signature scheme [Boneh et al. 2003]. 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 7.

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.

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, u delegates his 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 u, 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 u. Note that u is not the manager of group G. Indeed, u can be anyone outside group G. Our proxy signature scheme for groups is scalable, and is particularly suitable for role-based systems [Sandhu et al. 1996]. 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.

1.3 Organization of the paper

In Section 2, we give an overview of the aggregate signature by Boneh *et al* [Boneh et al. 2003]. A cross-domain role-based delegation example is given in Section 3. The definition and construction of our anonymous-signer aggregate signature scheme are given in Section 4. We prove the security properties in Section 5. In Section 6, we introduce the anonymous role-based cascaded delegation protocol. The analysis of the anonymous role-based cascaded delegation protocol is given in Section 7. Related work is described in Section 8. Conclusions are given in Section 9.

2. PRELIMINARIES

Here, we describe the aggregate signature scheme [Boneh et al. 2003] 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 [Boneh et al. 2003]. It uses bilinear maps [Boneh and Franklin 2001b] 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 [Okamoto and Pointcheval 2001] and are explained further in Section 4.1. The BGLS scheme comprises five algorithms: BGLS_KeyGen, BGLS_Sign, BGLS_Aggregate, BGLS_Verify, and BGLS_Agg-Verify. The first three algorithms are defined the same as in ordinary signature schemes; BGLS_Aggregate merges multiple signatures into one signature of constant length; BGLS_Agg-Verify 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 [Boneh et al. 2003]. 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 [Boneh et al. 2003]. We refer readers to the paper of BGLS scheme [Boneh et al. 2003] for further details.

Our anonymous-signer aggregate signature scheme is constructed based on the aggregate signature scheme [Boneh et al. 2003]. 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. AN EXAMPLE OF CROSS-DOMAIN ROLE-BASED DELEGATION

Example 2 is a multi-step role-based delegation that transfers rights among roles across different administrative domains in a collaboration. Example 2 is conceptually more complex than Example 1 in the introduction.

EXAMPLE 2. [**Scenario**] *Suppose that a hospital has a collaborative project with members of the role* Staff *in a lab. The collaboration requires* Staff *to access certain resources (e.g., medical databases) at the hospital. Also suppose that the lab further subcontracts a part of the project to a company. This subcontract requires a role* Contractor *at the company to also access the resources at the hospital. Therefore, in this example, a two-step delegation is needed to transfer privileges first to the role* Staff *and then to the role* Contractor*. Note that there is no single administrative authority and the three organizations are autonomous.*

Suppose the priviledges (e.g., accessing medical databases) required in the project are all associated with the role Visiting Researcher *at the hospital. Therefore, when the role* Visiting Researcher *is delegated to the role* Staff*, all members of the role* Staff *at the lab are authorized the privileges associated with role* Visiting Researcher *and thus can access the required data. Furthermore, a member of role* Staff *needs to extend the role* Visiting Researcher *to members of role* Contractor*, so that the collaborators at the company can share the resources as well. The rights are transferred by delegation as follows.*

[**Delegation step 1**] *An administrator at the hospital delegates the role* Visiting Researcher *to the lab's role* Staff *in a credential C. This delegation means that a member of* Staff *at the lab is also a member of* Visiting Researcher *at hospital, and can access resources that are associated with the role* Visiting Researcher*. John is a member of the role* Staff *and has the corresponding role credential R. Therefore, John now is delegated the hospital's role* Visiting Researcher*.*

[**Delegation step 2**] *To transfer the access privileges associated with role* Visiting Researcher *to* Contractor *at the company, John (or any authorized* Staff *member) further delegates the role* Visiting Researcher *to the role* Contractor *in a credential C'. This delegation means that a member of role* Contractor *at the company is also a member of* Visiting Researcher *at the hospital.*

[**Delegation chain for** Contractor] *Credentials C, R, and C' constitute the delegation credential that authorizes the role* Contractor*. Note that the role credential R proves that John is indeed a member of* Staff *and thus is entitled to issue delegations.*

[**Accessing resource**] *When a member of* Contractor *at the company requests to access the shared resources at the hospital, he or she presents the delegation chain shown above along with the proof of* Contractor *membership.*

4. 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.

4.1 Assumptions

Similar to the aggregate signature scheme [Boneh et al. 2003], our anonymous-signer aggregate signature scheme uses bilinear maps and works in gap groups [Boneh et al.

2001; Okamoto and Pointcheval 2001], 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 [Boneh et al. 2003], 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 [Boneh and Franklin 2001b] for examples and discussions of groups that admit such pairings.

Reverse Computation Diffie-Hellman (RCDH) Problem: Given a randomly chosen $P \in G_1$, aP, and bP (for unknown randomly chosen $a, b \in \mathbb{Z}_q$), compute cP such that $aP = bcP$.

RCDH problem has been shown to be equivalent to CDH problem by Chen, Zhang, and Kim [Chen et al. 2003], which is shown briefly as follows for completeness. Suppose one can solve CDH problem in G_1 on (P, aP, bP), then one can obtain abP. Let $Q = bP$. Then $P = b^{-1}Q$, $aP = ab^{-1}Q$, and $abP = aQ$. This means that we can obtain aQ from $(Q, b^{-1}Q, ab^{-1}Q)$. Thus solves RCDH problem. Given (P, aP, bP), suppose one can solve RCDH problem in G_1. Then one can first obtain $b^{-1}P$ from (P, bP) because $P = bb^{-1}P$. Then we can solve RCDH problem on $(P, aP, b^{-1}P)$ to obtain abP, as $aP = (ab)b^{-1}P$. This means that we obtain abP and thus solve CDH problem.

Admissible pairings: Following Boneh and Franklin [Boneh and Franklin 2001b], 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 [Boneh and Franklin 2001b; Yao et al. 2004], and most recently in a broadcast encryption scheme with short ciphertexts and private keys [Boneh et al. 2005].

4.2 Operations

An anonymous-signer aggregate signature scheme consists of AA_Setup, AA_Join, AA_Sign, AA_Aggregate, AA_Verify, and AA_Open algorithms.

AA_Setup: On input a security parameter k, a probabilistic algorithm outputs a role public key P_{A_1}. Each entity (role manager and role member) also chooses his

or her public/private keys.

AA_Join: A protocol is run between the role manager A_1 and a user that results in the user becoming a new role member. The outputs of the user are membership certificates and membership secrets.

AA_Sign: An algorithm takes as inputs a role public key, a membership secret, a membership certificate, and a message M_1, and outputs a role signature on M_1.

AA_Aggregate: This deterministic algorithm takes as inputs a number of role signatures and returns one aggregate signature S_{Agg}.

AA_Verify: An algorithm takes as inputs the role public keys $P_{A_1}, \ldots, P_{A_n}$, the aggregate signature S_{Agg}, and the messages $M_1, \ldots, M_n$. n is the number of signatures in the aggregation. P_{A_i} is the role public key of role manager A_i whose member signs message M_i in S_{Agg}, for $i \in [1, n]$. The output is 1 or 0.

AA_Open: The deterministic algorithm takes as inputs the message $M_1, \ldots, M_n$, the signature S_{Agg}, and role manager A_1's secret information to return the identity of the signer on message M_1.

A secure anonymous-signer aggregate signature scheme must satisfy the following properties. We give formal definitions for these security properties in the next section.

Correctness: Signatures produced by a role member using AA_Sign must be accepted by AA_Verify, and the AA_Open recovers the identity of the signer.

Unforgeability: Only valid 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.

We achieve the exculpability property because the manager cannot frame the member. A role member cannot deny his signature to the role manager because the manager possesses a proof that binds the signature to the role member's long-term public key. The signature itself only serves as a partial proof. Only the role manager can revoke the anonymity and this can be done any time without any restriction.

4.3 Formal Definitions of Security Properties

Here, we give formal definitions in game models for properties of unforgeability, anonymity, traceability, and exculpability for an anonymous-signer aggregate signature scheme. We will show later in Section 5 that unlinkability and collusion-resistance are implied by our definitions. Our anonymity and traceability definitions follow the definitions by Bellare, Micciancio, and Warinschi who gave the first formal treatment of group signatures [Bellare et al. 2003]. We do not give game-based security definitions for properties of correctness and aggregation as the definitions in the above section are sufficiently clear.

For unforgeability definition, the challenge key corresponds to the role public key. For anonymity definition, there are two challenge keys that correspond to two users' public keys, and we allow the adversary to adaptively choose both targets. We allow adversary to choose messages and query for opening signatures on the challenge public key(s). Similar to aggregate signature security [Boneh et al. 2003], for a signature aggregated from n role signatures, the adversary is free to choose $n-1$ of the signing keys in all our definitions.

UNFORGEABILITY **Setup:** The challenger chooses a role manager's public key P_{A_1} by random and gives the adversary P_{A_1}. The challenger keeps the corresponding secret key. The challenger also gives the adversary public/private key pairs of all role members.

Join query: The adversary adaptively requests to join the role by asking for membership certificates of users of her choice. The challenger uses role manager's secret key to generate role certificates.

Hash query: The adversary requests the hash of a message of her choice.

Open query: The adversary requests to open anonymous-signer aggregate signatures of her choice.

Unforgeability response: The adversary outputs an anonymous-signer aggregate signature σ along with verification keys $P_{A_1}, P_{A_2}, \ldots, P_{A_n}$, and messages $M_1, \ldots, M_n$. The restrictions are that (1) message M_1 has not been queried and (2) all messages are distinct [1]. $P_{A_1}, \ldots, P_{A_n}$ correspond to the role public keys that are needed to verify the n signatures in the aggregation. The adversary breaks the unforgeability if σ can be verified.

ANONYMITY **Setup:** Same as in the unforgeability definition. In addition, the adversary chooses a message M at this phase.

Join, Hash, Open queries: Same as in the unforgeability definition.

Anonymity challenge: Once the adversary decides the query phase is over, she outputs two users' public keys P_u^0, P_u^1. The challenger picks a random bit $b \in \{0,1\}$ and computes a challenge role signature ρ on M with the secret key corresponding to P_u^b. The adversary's task is to guess which user generates ρ.

The adversary can continue to submit more open queries on signatures other than ρ.

Anonymity response: The adversary outputs a guess b' and wins if $b' = b$.

TRACEABILITY **Setup:** The challenger gives the role manager's public key P_{A_1}

[1] This restriction is inherited from the aggregate signature scheme (See explanation on page 6 of BGLS paper [Boneh et al. 2003])

to the adversary as in unforgeability definition. The challenger gives the public keys of all role members to the adversary.

Key extract query: The adversary obtains private keys of users in a set C of her choice.

Join, Hash, Open queries: Same as in the unforgeability definition.

The order of the above queries is up to the adversary.

Traceability response: The adversary outputs an anonymous-signer aggregate signature τ along with $P_{A_1}, \ldots, P_{A_n}$, and messages $M_1, \ldots, M_n$. The restrictions are that (1) message M_1 has not been queried and (2) all messages are distinct. The adversary wins if τ can be verified and the signer associated with role manager P_{A_1} is opened to $\perp$ or to a user not in set C.

Compared to the "full-traceability" definition by Bellare, *et al* [Bellare et al. 2003], our definition for traceability is weaker as we do not give the adversary the private key of the group manager. In our exculpability definition, an adversary is allowed to have the role manager's secret key, which means that we consider the case of a malicious role manager. The exculpability adversary is given the challenge that is the public key of a target role member.

EXCULPABILITY **Setup:** The challenger chooses a role manager's public key P_{A_1} and a challenge public key P_u by random. P_u is the public key of the target role member that the adversary needs to attack. Let s_{A_1} be the private key corresponding to P_{A_1}. P_{A_1}, s_{A_1}, and P_u are given to the adversary. The challenger keeps the private key of the target user. The challenger also gives the adversary the public/private keys of all the other role members.

Exculpability response: The adversary outputs an anonymous-signer aggregate signature ϕ along with $P_{A_1}, \ldots, P_{A_n}$, and messages $M_1, \ldots, M_n$. The restrictions are that (1) message M_1 has not been queried in the Sign queries and (2) all messages are distinct. The adversary wins if ϕ can be verified and the signer associated with role manager P_{A_1} is opened to the target role member with public key P_u.

Our exculpability definition is restrictive in that it does not allow an adversary to issue queries on the target. Ideally, an adversary may obtain from the challenger signatures of the target on messages of the adversary's choice. Note that the adversary can generate and open signatures of other role members on her own as she is given the private keys of the rest of group.

4.4 Construction

One can construct a naive aggregate group signature scheme from BGLS scheme [Boneh et al. 2003] and one-time signing keys as follows. In a naive scheme, a group member generates a public/private key pair (PK, SK), by running the key generation algorithm of BGLS aggregate signature scheme. The group manager signs (with the group master secret) the public key, and sends the certificate [2] $Cert$back to the group member. To produce a signature on message M, the group member signs M with the private key SK to create signature Sig as in the aggregate signature scheme, and sends $(M, Sig, PK, Cert)$ to the verifier. Signature

[2]This certificate is issued by a group manager for proving group membership of a member. It is different from a CA certificate, which certifies the validity of a public key.

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 secret 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

Table I. Notation for anonymous-signer aggregate signature scheme.

Sig can be aggregated with other signatures of this scheme as in BGLS aggregate signature scheme. However, the above scheme cannot prevent the group manager from misattributing signatures. The group manager first runs the key generation algorithm of BGLS aggregate signature scheme 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.

We overcome the above problem by designing signing keys that are both unlinkable and tied to the *long-term* private key of a signer. In our protocol, a role 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 role manager. The long-term public key of the role member is certified by a Certificate Authority (CA), which serves as a trusted entity. Misattributing a signature to others is impossible, even for the manager, because a role member can prove that the signature does not correspond to his CA-certified public key. The underlying bilinear pairing allows us to achieve this property.

The notation of our anonymous-signer aggregate signature scheme is shown in Table I. The last three items in Table I 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 secret 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 I.

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

—A trusted third party 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 [Bellare and Rogaway 1993].

—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 secure signature scheme. The certification binds a long-term public key with its owner. The public key certificate of a member is used for repudiating misattributed signatures, and is different from the *one-time signing permits* below. The CA can be any trusted third party, but cannot be the same as the role manager. [3]

AA_Join: A role member u obtains one or more *one-time signing permits* from the role manager. Each permit certifies u's one-time signing key information, and is used for issuing role signatures. The following shows how the signing permits are generated.

—u randomly chooses l number of secrets $x_1, \ldots, x_l$. u 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 in a secure channel [4], for all $i \in [1,l]$.

—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. The role manager makes sure that the one-time signing public keys submitted by u and on the manager's record are all unique. This check is necessary for the traceability requirement, otherwise colluding members can submit identical signing keys by manipulating their private keys and signing factors. Finally, the role manager runs BGLS_Sign on inputs s_A and strings *roleinfo* $\parallel K_{u,i}$ to obtain $S_{u,i} = s_A H(\textit{roleinfo} \parallel K_{u,i})$ for all $i \in [1,l]$. $S_{u,i}$ is the i-th *one-time signing permit* for u, and is given to u. The role manager adds tuples $(P_u, X_{u,i}, K_{u,i})$ to its record for all $i \in [1,l]$.

AA_Sign : A role member u first computes a signature S_u on a message M on behalf of the role, by running BGLS_Sign on inputs $s_u x_i$ and M, where $s_u x_i$ is one of his one-time signing secrets. Then, u calls algorithm BGLS_Aggregate to merge signature S_u with his one-time signing permit $S_{u,i}$ corresponding to 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.

—u runs BGLS_Sign with secret key $s_u x_i$ and message M, and obtains a signature $S_u = s_u x_i H(M)$.

—u 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 BGLS_Aggregate, which returns a signature $S_g = S_u + S_{u,i}$. Recall that $S_{u,i} = s_A H(\textit{roleinfo} \parallel K_{u,i})$. S_g is output as the role signature. u also outputs public key P_A and one-time signing public key $K_{u,i}$.

[3] For simplicity, we assume that at a given time each user has only one long-term public key.

[4] $X_{u,i}$ needs to be kept secret because it can be used to identify the signer along with public information P_u and $K_{u,i}$.

AA_Aggregate: Same as in algorithm BGLS_Aggregate. 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 all $k \in [1,n]$ 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.

AA_Verify: This algorithm calls algorithm BGLS_Agg-Verify 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 $k \in [1,n]$. n is the number of signers on the authorization chain.

—For $1 \leq k \leq n$, compute the hash digest $H(M_k)$ of message M_k and $h_k = H(roleinfo_k \parallel K_{u_k,i_k})$ 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.

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

Opening of the signature by the role manager correctly identifies the signer, which is shown next.

AA_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.

THEOREM 1. *The communication complexity of* AA_Join *algorithm is* $O(l)$*, where* l *is the number of one-time signing keys certified. The computational complexity of the* AA_Verify *algorithm is* $O(n)$*, where* n *is the number of signatures aggregated.*

5. SECURITY

We have shown that our anonymous-signer aggregate signature scheme satisfies the correctness requirement. It also clearly satisfies the aggregation property. In this section, we prove the security properties for our anonymous-signer aggregate signature scheme.

THEOREM 2. *Our anonymous-signer aggregate signature scheme is as secure as the BGLS aggregate signature scheme against existential forgery attacks.*

Proof: There are three parties involved in this proof: a challenger, adversary $\mathcal{A}$, and adversary $\mathcal{B}$. If adversary $\mathcal{B}$ has a non-negligible advantage over the unforgeability of our anonymous-signer aggregate signature scheme, then $\mathcal{A}$ uses $\mathcal{B}$ to break the BGLS aggregate signature scheme by answering the challenge posed by the challenger. The challenger chooses a public key P_{A_1} by random and gives P_{A_1} to $\mathcal{A}$ as the BGLS challenge. The challenger keeps the corresponding secret key s_{A_1}. $\mathcal{A}$ then interacts with $\mathcal{B}$ as follows.

Setup: $\mathcal{A}$ gives $\mathcal{B}$ the challenge key P_{A_1}. $\mathcal{A}$ generates a set of public/private key pairs and gives them to $\mathcal{B}$ as the keys of all role members.

Join query: $\mathcal{A}$ answers $\mathcal{B}$'s join query by submitting it to the challenger as follows. Suppose $\mathcal{B}$'s query is to join a user with public key P_u, one-time signing factors X_u, and signing key K_u. $\mathcal{A}$ tests whether $e(P_u, X_u) = e(K_u, \pi)$ holds. If yes, then $\mathcal{A}$ asks the challenger to sign ($roleinfo \parallel K_u$) with secret s_{A_1}. $\mathcal{A}$ passes the signature to $\mathcal{B}$ as the signing permit. $\mathcal{A}$ also keeps the tuple (P_u, K_u, X_u) for the record.

Hash query: $\mathcal{A}$ simply uses a collision-resistant hash function to compute the hash of messages of $\mathcal{B}$'s choice.

Open query: $\mathcal{B}$ requests to open an anonymous-signer aggregate signature of her choice. $\mathcal{A}$ can easily identify the signer's P_u by looking up the signing key K_u in $\mathcal{A}$'s record.

Unforgeability response: $\mathcal{B}$ outputs an anonymous-signer aggregate signature σ along with verification keys $P_{A_1}, P_{A_2}, \ldots, P_{A_n}$, $K_1, \ldots, K_n$, strings $roleinfo_1$, ..., $roleinfo_n$, and messages $M_1, \ldots, M_n$. $P_{A_1}, \ldots, P_{A_n}$ correspond to role public keys that are needed to verify the n signatures in the aggregation. $\mathcal{A}$ makes sure that (1) signing key K_1 has not been queried, (2) $roleinfo_1 \parallel K_1$, ..., $roleinfo_n \parallel K_n$ are distinct, and (3) messages $M_1, \ldots, M_n$ are distinct. $P_{A_1}, \ldots, P_{A_n}$ correspond to the role public keys that are needed to verify the n signatures in the aggregation.

$\mathcal{A}$ passes σ to the challenger, along with keys $P_{A_1}, P_{A_2}, \ldots, P_{A_n}$, $K_1, \ldots, K_n$, and messages $roleinfo_1 \parallel K_1$, ..., $roleinfo_n \parallel K_n$, $M_1, \ldots, M_n$.

If $\mathcal{B}$ breaks the unforgeability, i.e., σ can be verified correctly, with advantage ϵ, then $\mathcal{A}$ breaks BGLS with advantage ϵ. □

Next we show that our signature scheme satisfies the anonymity property.

THEOREM 3. *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups preserves anonymity in the random oracle model.*

Proof: We first design a new game, called random-x game, that is secure based on randomness, i.e., the adversary's advantage over random guessing is negligible. Then we reduce the anonymity game to the random-x game, i.e., breaking anonymity means that breaking the random-x game. Random-x game is as follows.

RANDOM-x GAME

The challenger chooses public parameters: a gap group G_1 of prime order q, a generator π of G_1. The challenger also generates a set of public/private key pairs in the form of $P_u = s_u\pi$ and s_u. These public/private key pairs are given to the adversary.

Query phase: The adversary chooses two public keys P_u^0 and P_u^1, and gives them to the challenger. The challenger picks a random secret x, a random bit b, and computes $K_u^b = s_u^b x\pi$. The adversary is given K_u^b, $x\pi$, and b, so that she learns K_u^b is computed from s_u^b.

When the adversary decides the query phase is over, she outputs two public keys P_u^{*0} and P_u^{*1} to be challenged on.

Challenge phase: The challenger picks a random secret x, a random bit b, and computes $K_u^{*b} = s_u^{*b} x\pi$, where $P_u^{*b} = s_u^{*b}\pi$. The adversary is given K_u^{*b}. The adversary's task is to guess whether b is 0 or 1.

The adversary submits more queries. Finally, the adversary outputs her guess b', and wins if $b' = b$. Because x is randomly chosen, the adversary does not have the advantage over random guessing, thus has negligible advantage in the random-x game. Note that the adversary does not know $x\pi$ of her challenge.

Assume that an adversary $\mathcal{A}$ can break the anonymity of our anonymous-signer aggregate signature scheme. Then an adversary $\mathcal{B}$ can use $\mathcal{A}$ to gain non-negligible advantage in the random-x game as follows. We model hash function H as a random oracle.

Setup: $\mathcal{B}$ first obtains the public parameters from her challenger in the random-x game. $\mathcal{B}$ then chooses a role manager's secret key s_A by random and computes the public key $P_{A_1} = s_A\pi$. $\mathcal{B}$ gives $\mathcal{A}$ P_{A_1}, and keeps s_A. $\mathcal{B}$ also gives $\mathcal{A}$ the public/private key pairs of all role members that $\mathcal{B}$ obtains from her challenger. $\mathcal{A}$ outputs a message M that $\mathcal{A}$ wishes to be challenged on.

Join query: $\mathcal{A}$ adaptively requests to join the role by asking for membership certificates of users of her choice. $\mathcal{B}$ uses s_A to generate role certificates. B makes sure that one-time signing public keys on her record are all unique.

Hash query: For messages other than M, $\mathcal{B}$ picks a random value in $\mathbb{Z}_q$ and returns it as the hash value. For message M, $\mathcal{B}$ picks a random r and returns $r\pi$ as its hash.

Open query: $\mathcal{B}$ also answers $\mathcal{A}$'s requests of opening anonymous-signer aggregate signatures. $\mathcal{B}$ can do this because the signing keys and the associated (long-term) public keys are recorded during the join query phase.

Anonymity challenge: Once $\mathcal{A}$ decides the query phase is over, $\mathcal{A}$ outputs two targets' public keys P_u^{*0}, P_u^{*1} in the random-x game. Then $\mathcal{B}$ outputs these two keys to her challenger as her targets. $\mathcal{B}$'s challenger picks a random bit $b \in \{0, 1\}$ and a random x, and computes $K_u^{*b} = xP_u^{*b} = s_u^{*b} x\pi$. K_u^{*b} is given to $\mathcal{B}$ as the challenge (in the random-x game). Next, $\mathcal{B}$ needs to generate a challenge role signature ρ on M for $\mathcal{A}$ using K_u^{*b} as the signing key even though $\mathcal{B}$ does not know $s_u^{*b} x$. (The trick is in the hash of message M.) $\mathcal{B}$ computes signature $\rho = s_A H(\mathit{roleinfo} \parallel K_u^{*b})$ $+ rK_u^{*b}$. ρ can be correctly verified: $e(\rho, \pi) = e(H(\mathit{roleinfo} \parallel K_u^{*b}), P_A) e(r\pi, K_u^{*b})$ $= e(H(\mathit{roleinfo} \parallel K_u^{*b}), P_A) e(H(M), K_u^{*b})$.

$\mathcal{B}$ answers more open queries from $\mathcal{A}$ on signatures other than ρ as before.

Anonymity response: $\mathcal{A}$ outputs a guess b'. $\mathcal{B}$ outputs b' as her guess in the random-x game. If $\mathcal{A}$ has non-negligible advantage ϵ in breaking the anonymity of our anonymous-signer aggregate signature scheme, then $\mathcal{B}$ has advantage ϵ in breaking the random-x game. □

Anonymity as defined in Section 4.3 naturally implies unlinkability. In fact, they are technically the same property as observed in [Bellare et al. 2003]. We summarize the unlinkability property in the following corollary.

COROLLARY 1. *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups satisfies the unlinkability requirement in the random oracle model.*

Next we show the traceability of our signature scheme. Intuitively, traceability means that the role manager is always able to open a valid signature and identify the signer.

THEOREM 4. *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups satisfies the traceability requirement in the random oracle model under the CDH assumption.*

Proof: We prove traceability by contradiction. We show that if an adversary has non-negligible advantage in the traceability game, then there is a contradiction.

Setup, **Join query**, **Hash query**, and **Open query:** The challenger performs as in the unforgeability proof for Theorem 2. In addition, in **Join query**, the challenger makes sure that the one-time signing public keys are all distinct. At the end of join queries, the challenger has recorded a list of tuples (P_u, K_u, X_u) for the users that have been queried.

Traceability response: The adversary outputs a signature τ along with keys $P_{A_1}, \ldots, P_{A_n}$, $K_1, \ldots, K_n$, strings $roleinfo_1$, ..., $roleinfo_n$, and messages $M_1, \ldots, M_n$. As defined, τ should satisfy the following restriction: (1) signing key K_1 has not been queried, (2) $roleinfo_1 \parallel K_1$, ..., $roleinfo_n \parallel K_n$ are distinct, and (3) messages $M_1, \ldots, M_n$ are distinct.

Suppose the adversary wins. This means that (i) τ can be verified and (ii) the signer associated with role manager P_{A_1} is identified to $\perp$ after opening τ. In our signature scheme, (ii) means that $e(K_1, \pi) \neq e(P_u, X_u)$ for all users on the challenger's record. However, (i) means that τ is correctly formed, in particular, (i) means that τ contains a valid signing permit (in the form of $s_{A_1} H(roleinfo_1 \parallel K_1)$) for signing public key K_1. We have shown in Theorem 2 that the signing permit cannot be forged, which implies that the adversary obtains it from the challenger. Thus the challenger must have a unique tuple corresponding to K_1 on the record. We reach a contradiction. □

Next, we prove the exculpability of our signature scheme [5].

THEOREM 5. *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups satisfies the exculpability requirement in the random oracle model under the CDH assumption.*

[5]Our traceability does not directly imply exculpability as in [Bellare et al. 2003] because the traceability definition does not give the adversary the private key of the role manager.

Proof: There are three parties involved in this proof: a challenger, adversary $\mathcal{A}$, and adversary $\mathcal{B}$. If adversary $\mathcal{B}$ has a non-negligible advantage over our anonymous-signer aggregate signature scheme, then $\mathcal{A}$ uses $\mathcal{B}$ to solve the RCDH problem (defined Section 4.1).

Setup: $\mathcal{A}$'s challenger chooses s and x by random and gives $\mathcal{A}$ (π, $s\pi$, $sx\pi$) as the challenge. Denote $s\pi$ by P_u and $sx\pi$ by K_1. They correspond to the long-term public key and one-time signing public key of a role member u, respectively. Adversary $\mathcal{A}$ also chooses a role manager's private key s_{A_1} by random and computes the corresponding public key $P_{A_1} = s_{A_1}\pi$.

$\mathcal{A}$ gives $\mathcal{B}$ keys P_{A_1}, s_{A_1}, P_u, and K_1. $\mathcal{A}$ also gives $\mathcal{B}$ the public/private keys of all the other role members that $\mathcal{A}$ generates on her own. Adversary $\mathcal{B}$'s task is to use K_1 as one of the signing public keys to create an anonymous-signer aggregate signature misattributing role member u.

Exculpability response: Adversary $\mathcal{B}$ outputs an anonymous-signer aggregate signature ϕ along with $P_{A_1}, \ldots, P_{A_n}$, $K_1, \ldots, K_n$, and messages $M_1, \ldots, M_n$. The restriction is that all messages are distinct. If adversary $\mathcal{B}$ wins then ϕ can be verified and the signer associated with role manager P_{A_1} is opened to the target role member with public key P_u. The latter means that $\mathcal{B}$ can provide the proof X that satisfies $e(s\pi, X) = e(K_1, \pi)$. Because of the bilinearity of e, X must be $x\pi$. Therefore, $\mathcal{A}$ outputs X as her answer and solves the RCDH problem on $sx\pi$ and $x\pi$. □

Our definitions of unforgeability, anonymity, traceability, and exculpability allow the adversary to have the private keys of any number of users except the target(s), therefore our signature scheme is naturally collusion-resistant.

COROLLARY 2. *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups is collusion-resistant in the random oracle model under the CDH assumption.*

6. ANONYMOUS ROLE-BASED CASCADED DELEGATION PROTOCOL

In this section we first briefly introduce the role-based cascaded delegation (RBCD) protocol [Tamassia et al. 2004; Yao et al. 2005]. A large amount of work has been done on trust management and distributed authorization systems [Aringhieri et al. 2005; Aura 1999; Blaze et al. 1998; Clarke et al. 2001; Li et al. 2002]. Among them, role-based cascaded delegation [Tamassia et al. 2004] 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 manager. It enables flexible and dynamic authorization in a decentralized environment. Using predicates and constraints, it is also possible to restrict the scope of the delegation, e.g., prevent further delegation [Yao et al. 2005]. RBCD 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 [Tamassia et al. 2004], a delegation credential includes role member-

ship 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.

Next we 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 [Tamassia et al. 2004], 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 manager, which is equivalent to a role manager in the anonymous-signer aggregate signature scheme.

6.1 Definitions

An anonymous role-based cascaded delegation protocol defines five operations: ARBCD_Initiate, ARBCD_Extend, ARBCD_Prove, ARBCD_Verify, and ARBCD_Open.

ARBCD_Initiate: Same as in RBCD protocol [Tamassia et al. 2004], 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 manager.
ARBCD_Extend: This operation is similar to ARBCD_Initiate, but is run by an intermediate delegator u, who is a member of a role that is delegated a privilege by credential C. The goal is for u 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 u received, and the proof of u's role membership. C' does not disclose the identity of u.

Credential C' may simply be an accumulation of individual certificates. In comparison, our realization using anonymous-signer aggregate signatures is more efficient.
ARBCD_Prove: A requester u with role r produces a proof, which authenticates the delegation chain connecting the resource owner with u. This includes a proof of u's role membership without disclosing the identity, and the delegation credential that delegates the requested privilege to r.
ARBCD_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.
ARBCD_Open: Role manager revokes the anonymity of a delegator by examining signatures on a delegation credential. The identity of the delegator is returned.

6.2 Realization

We give an anonymous RBCD protocol using anonymous-signer aggregate signatures. Compared to the original RBCD protocol [Tamassia et al. 2004], 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.

ARBCD_Setup: A trusted third party runs AA_Setup to set up public parameters *params*, and individuals to choose and certify long-term keys. Then, AA_Join protocol is run between role members and the role manager to set up one-time signing permits. The role manager also keeps a record of signing key information.
ARBCD_Initiate: A resource owner runs the BGLS_Sign to sign a delegation statement that authorizes a certain privilege to a role r. The inputs to BGLS_Sign are the resource owner's private key and the delegation statement that includes the delegated privilege, the role name r, and the role manager's long-term public key. The output is a delegation credential for r.
ARBCD_Extend: Role r is delegated a privilege, and a member u of r wants to further delegate the privilege to a role r'. u first runs algorithm AA_Sign to generate a role signature for r'. The inputs to AA_Sign are u's one-time signing secret key $s_u x_i$, a delegation statement, and u's one-time signing permit S_i corresponding to $s_u x_i$. The delegation statement includes the following information: role name r', the long-term public key of r''s manager, delegated privilege, AA_Sign returns a role signature *Sig*. Then the public signing key $s_u x_i \pi$ is appended to the delegation statement. Finally, delegator u calls AA_Aggregate to aggregate *Sig* with the signature on the delegation credential issued to role r. The resulting aggregate signature S_{Agg} and delegation statements are given to members of role r' as the delegation credential.
ARBCD_Prove: A requester u of role r first runs AA_Sign that uses a one-time signing key to sign a random challenge message T chosen by verifier. The random challenge is to prevent replay attacks and ensure that u possesses the secret signing key. Then, u calls AA_Aggregate to merge the output signature with the signature on the delegation credential issued to role r. The outputs are returned.
ARBCD_Verify: AA_Verify is run to verify the aggregate signatures submitted by the requester against the delegation statements. The request is granted if the signature is accepted, and rejected if otherwise. Note that the delegation statements include the following public keys required to verify the aggregate signature: (1) public signing keys of intermediate delegators, and (2) long-term public keys of role managers. (1) are for verifying the signatures created in ARBCD_Extend operations; and (2) are for verifying one-time signing permits of intermediate delegators. We assume that the verifier knows the public key of the resource owner, which is needed to verify the signature generated in ARBCD_Initiate.
ARBCD_Open: A role manager runs algorithm AA_Open 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 manager

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 [Tamassia et al. 2004]. The time required for signing and verification is the same as in the original RBCD protocol [Tamassia et al. 2004]. In anonymous RBCD, role managers 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 [Barreto et al. 2002]). As described in Section 1.2, the above protocol gives rise to a proxy signature scheme for groups. Details (definitions, construction, and proof of security) of the proxy signature scheme are omitted in this paper.

7. ANALYSIS

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 [Ateniese et al. 2000], and five times shorter than the one generated in BBS scheme [Boneh et al. 2004]. 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 manager, 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 [Juels 1999] and the secret handshakes protocol [Balfanz et al. 2003]. 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 aggre-

gate 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, 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. Note that there is a conceptual difference between the aggregate algorithm in BGLS scheme [Boneh et al. 2003] 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.

Revocation Role membership revocation before the expiration can be handled by maintaining a revocation service, which can be efficiently achieved using authenticated dictionaries (e.g. [Goodrich et al. 2003; Naor and Nissim 1998]). Authenticated dictionary is a system for distributing data and supporting authenticated responses to queries. One-time signing public keys of revoked members are put on the repository of revocation service by a role manager. Before verifying a role signature, the revocation service is queried to ensure that the signature is not generated by a revoked signing key.

Anonymity of Role Manager In our schemes, the public key of the role manager is required to verify role signatures, and therefore known to the public. Nevertheless, it does not mean that the role manager cannot sign messages anonymously. On the contrary, a role manager can run the protocols to certify a set of secret signing keys of his choice and use them as signing keys without disclosing the identity. Therefore, our schemes provide the same signing ability to every role member including the role manager.

8. RELATED WORK

Our signature scheme has properties that are related to group signature schemes. Group signatures, introduced by Chaum and van Heijst [Chaum and van Heijst 1991], 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 [Chaum and van Heijst 1991], 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 [Camenisch and Stadler 1997], and followed by a number of improvements [Ateniese et al. 2000; Boneh et al. 2004]. Until recently, group signature constructions (e.g., [Ateniese et al. 2000; Camenisch and Lysyanskaya 2002]) were mostly based on the strong-RSA assumption, and a group signature typically comprised of multiple elements of RSA-signature length. Recently, bilinear pairing [Boneh and Franklin 2001b] has been used to construct group signature schemes [Boneh et al. 2004;

Camenisch and Lysyanskaya 2004; Chen et al. 2003], whose security is based on variants of Diffie-Hellman assumptions. The group signature scheme by Boneh, Boyen, and Shacham [Boneh et al. 2004] significantly shortens the signature length, compared to the RSA-based state-of-the-art group signature scheme by Ateniese *et al.* [Ateniese et al. 2000]. An identity-based group signature scheme was proposed by Chen, Zhang, and Kim [Chen et al. 2003], 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 [Boneh and Franklin 2001b]). Bellare, Micciancio, and Warinschi gave the first formal treatment of group signatures by introducing strong, formal definitions for the core requirements of anonymity and traceability [Bellare et al. 2003]. They also developed a construction of a group signature scheme achieving the requirements based only on the existence of trapdoor permutations.

Most recently, Chase and Lysyanskaya gave an abstract construction of anonymous delegatable credentials based on their construction of signatures of knowledge [Chase and Lysyanskaya 2006]. Our anonymous role-based cascaded delegation can be implemented using their anonymous delegatable credential system, which allows one to issue delegation credential without revealing his or her identity and the delegation can be further extended to others anonymously. In comparison, we focus on the functionality of signature aggregation in addition to anonymous delegation in our construction.

There has been extensive research on access control models in the past decade [Ferraiolo and Kuhn 1992; Sandhu 1993; Sandhu et al. 1996]. The concept of role-based access control [Ferraiolo and Kuhn 1992; Sandhu et al. 1996] 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 [Blaze et al. 1998], delegation certificates [Aura 1999], SPKI [Clarke et al. 2001], Delegation Logic (DL) [Li et al. 2003], proof-carrying authorization (PCA) [Appel and Felten 1999], *RT* framework [Li et al. 2002], and role-based cascaded delegation [Tamassia et al. 2004]. 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 [Holt et al. 2003] 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) [Boneh and Franklin 2001a; Cocks 2001; Shamir 1984] 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.* [Bradshaw et al. 2004] 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.* [Frikken et al. 2004] give a scheme that

hides both credentials and policies. Most recently, Frikken *et al.* [Frikken et al. 2006] 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.

Anonymous credential systems have been developed [Camenisch and Lysyanskaya 2001; Camenisch and Van Herreweghen 2002; Chaum 1985; Chaum and Evertse 1987] 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.

9. CONCLUSION

We have proposed an anonymous role-based cascaded delegation (RBCD) protocol that protects sensitive role-membership information of delegators. Although the anonymous RBCD model can use any group signature scheme to realize, we have shown that there is a performance advantage using our anonymous-signer aggregate signature scheme. Anonymous-signer aggregate signature scheme is suitable for sensitive applications where a large number of signatures are produced and the role or group membership of a signer (instead of the identity of the signer) is needed for verification.

10. ACKNOWLEDGEMENTS

The authors would like to thank Anna Lysyanskaya and anonymous reviewers for their helpful comments and suggestions.

REFERENCES

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

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

Ateniese, G., Camenisch, J., Joye, M., and Tsudik, G. 2000. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — CRYPTO '00*. LNCS, vol. 1880. Springer Verlag, 255–270.

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

Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., and Wong, H. 2003. Secret handshakes from pairing-based key agreements. In *2003 IEEE Symposium on Security and Privacy*. IEEE Press, 180–196.

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

Bellare, M., Micciancio, D., and Warinschi, B. 2003. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology - EUROCRYPT*. Lecture Notes in Computer Science, vol. 2656. 614–629.

Bellare, M. and Rogaway, P. 1993. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*. ACM, 62–73.

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

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

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

Boneh, D. and Franklin, M. K. 2001b. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto '01*. LNCS, vol. 2139. Springer-Verlag, 213–229.

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

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

Boneh, D., Lynn, B., and Shacham, H. 2001. Short signatures from the Weil pairing. In *Advances in Cryptology — Asiacrypt '01*. LNCS, vol. 2248. Springer-Verlag, 514–532.

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

Camenisch, J. and Lysyanskaya, A. 2001. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Advances in Cryptology — EUROCRYPT 2001*, B. Pfitzmann, Ed. Lecture Notes in Computer Science, vol. 2045. Springer Verlag, 93–118.

Camenisch, J. and Lysyanskaya, A. 2002. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology — Crypto '02*. LNCS, vol. 2442. 61–76.

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

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

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

Chase, M. and Lysyanskaya, A. 2006. On signatures of knowledge. In *Advances in Cryptology - CRYPTO*. Lecture Notes in Computer Science, vol. 4117. Springer, 78–96.

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

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

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

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

Clarke, D., Elien, J.-E., Ellison, C., Fredette, M., Morcos, A., and Rivest, R. L. 2001. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security 9(4)*, 285–322.

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

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

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

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

Goodrich, M. T., Shin, M., Tamassia, R., and Winsborough, W. H. 2003. Authenticated dictionaries for fresh attribute credentials. In *Proc. Trust Management Conference*. LNCS, vol. 2692. Springer, 332–347.

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

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

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

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

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

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

Naor, M. and Nissim, K. 1998. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium*. 217–228.

Okamoto, T. and Pointcheval, D. 2001. The gap-problems: a new class of problems for the security of cryptographic schemes. In *PKC '01*. LNCS, vol. 1992. Springer-Verlag, 104–118.

Pallickara, S. L., Plale, B., Fang, L., and Gannon, D. 2006. End-to-end trustworthy data access in data-oriented scientific computing. In *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006)*. 395–400.

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

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. 1996. Role-based access control models. *IEEE Computer 29, Number 2*, 38–47.

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

Tamassia, R., Yao, D., and Winsborough, W. H. 2004. Role-based cascaded delegation. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*. ACM Press, 146 – 155.

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

Yao, D., Fazio, N., Dodis, Y., and Lysyanskaya, A. 2004. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. ACM Press, 354 – 363.

Yao, D. and Tamassia, R. 2006. Cascaded authorization with anonymous-signer aggregate signatures. In *Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics Information Assurance Workshop (IAW '06)*.

Yao, D., Tamassia, R., and Proctor, S. 2005. 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)*. IEEE Press, 157–168.

Yu, T., Ma, X., and Winslett, M. 2000. 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)*. 210–219.