Independently-Verifiable Decentralized Role-Based Delegation

Roberto Tamassia, *Member, IEEE,* Danfeng Yao, *Member, IEEE,* William H. Winsborough, *Member, IEEE*

***Abstract*— In open systems such as Grid computing and Internet, delegation transfers privileges among users across different administrative domains and facilitates information sharing. We present an independently-verifiable delegation mechanism, where a delegation credential can be verified without the participation of domain administrators. Our protocol, called *role-based cascaded delegation* (RBCD), supports simple and efficient cross-domain delegation of authority. RBCD enables a role member to create delegations based on the dynamic needs of collaboration, yet in the meantime a delegation chain can be verified by anyone without the participation of role administrators. We also describe an efficient realization of role-based cascaded delegation using aggregate signatures, where the authentication information for an arbitrarily long role-based delegation chain is captured by one short signature of constant size.**

***Index Terms*— decentralized delegation, digital credentials.**

I. Introduction

Trust management is an approach to access control in environments where entities that are not in the same security domain need to share resources. Several trust management systems have been proposed in recent years, e.g., PolicyMaker [4], KeyNote [3], SPKI/SDSI [9], and the *RT* framework [22]. The notion of delegation is essential in transferring trust and authorization in trust management systems. It facilitates information and resource sharing in distributed collaborative environment such as Grid computing and peer-to-peer networks. Delegation chains trace sequences of entities, starting from the resource owner and including entities authorized by (though possibly unknown to) the owner. These entities play a central part in authorization by providing the credentials that represent their own delegation acts, which enable the delegation chain to be verified.

In role-based delegation, delegated privileges are issued to a role rather than to an individual. The abstraction of roles makes delegation scalable as one delegation benefits all members. Although the concept of role-based delegation is not new in access control and trust management literature, multi-step role-based delegation chain and its verification have not been much studied. Most prior work that addresses the problem of determining whether credentials prove an entity's resource request is authorized [3], [4], [9] assumes that all potentially relevant credentials are available in one central storage. However, this assumption is not valid in decentralized environment where there is no central authority. In fact, the verification cost may be quite expensive in a typical trust management system implementation. Collecting and verifying delegation credentials incurs communicational and computational costs, as does checking that together the credentials provide proof that a given user is authorized. In this paper, we present techniques that can be used to significantly reduce these costs. Next, we illustrate in Example 1.1 a simple multi-step delegation scenario that transfers rights among roles within one administrative domain.

Example 1.1: 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. When an intern, say Carl, requests for access, the delegation credentials and role credentials are verified. In addition, Carl's *Intern* role credential is also verified to ensure he is indeed an intern.

Example 1.1 only involves one administrative domain, namely the hospital. Therefore, the credentials and the public keys of delegators (Bob and Alice) can be reasonably assumed to be available to the verifier that is the hospital server. However, trust management is to facilitate information sharing across different administrative domains. Delegation is usually decentralized and typically involves users and roles from multiple organizations. The verification process is much more complex, because there is no single trusted authority and public keys for credentials may not be known or trusted. We show in Section IV a more complex cross-domain role-based delegation. In this paper, we study the role-based delegation in the general setting. We propose a *role-based cascaded delegation* protocol that supports assured information sharing in a decentralized fashion.

For efficient transmission and storage, compact digital credentials are desirable. Multi-step delegation credentials may be lengthy because the verification of a delegation chain requires checking a number of signatures linear in the length of the chain, where the length is defined as the number of delegations on the chain. Conventional signature schemes,

A preliminary version of the paper was presented at the 2004 *ACM Symposium on Access Control Models and Technologies* [32]. Work supported in part by NSF grants CCF-0311510, CCF-0524010, CCR-0325951, and IIS–0324846, and by a research gift from Sun Microsystems.

Roberto Tamassia and Danfeng Yao are with Brown University; William H. Winsborough is with University of Texas at San Antonio.

such as RSA [28] and DSA [12], produce relatively long signatures compared to the security they provide. For a 1024-bit modulus (security level), RSA signatures are 1024 bits long and standard DSA signatures are 320 bits long. The number of signatures required to authenticate a role-based delegation chain of length n is about $2n$, because in addition to verifying each of the delegation transactions, one must verify that the intermediate delegators are members of the required roles. Among the signatures associated with a delegation chain, the signature on a role credential is generated by the administrator of that role independently from the rest of the signatures.

Unfortunately, it is not known how to aggregate individually generated signatures from different signers on different messages in conventional cryptosystems, such as RSA [7], [24]. This means that the entire set of signatures has to be stored by delegated entities, and transmitted across networks at each delegation and verification. Because intermediate delegators in our model may be entities who have limited computational power and communicational bandwidth, the implementation of role-based cascaded delegation using conventional credentials is inefficient. We overcome these problems by realizing the role-based cascaded delegation with short aggregate signatures [6], [8].

A. Our Contributions

Existing delegation models assume the delegation is issued by administrators. However, to enable flexible resource sharing, the decision of introducing new role members into a collaboration needs to be dynamically made by members of existing roles, without the involvement of administrators. In the meantime, the shared information needs to be adequately protected against unauthorized or unqualified users. These goals drives us to reexamine conventional assumptions in role-based delegation, and define our model from a different perspective. We summarize our contributions next.

- We present a role-based delegation mechanism that supports the efficient verification of multi-step delegation chains. It has two main features: **(1)** flexible delegation where a delegation can be issued by a valid member of a role, not just by the administrator; and **(2)** simple verification where a delegation credential is self-contained and the verification does not require the participation of any role administrators. Our delegation mechanism takes a simple accumulation approach, where each intermediate delegator passes down the relevant digital credential to the delegated entity for later verification.
- We give a detailed protocol specification for public key signing and management that ensures the integrity of shared resources. The significance of our RBCD protocol is that we do not assume the existence of a public key infrastructure (PKI), which may be expensive to adopt widely. This feature makes our protocol general enough for many decentralized and open system environments such as peer-to-peer networks, where there are no central authorities and PKI is usually not available. Another important feature of our delegation protocol is that a delegation is issued to a role, yet we are able to support access accountability. That is, in case of misbehaving individuals, the resource owner can identify their identities, who are authorized through our role-based delegation.
- We also present a concrete realization of RBCD that gives compact delegation credentials. Traditionally, the number of signatures required for the verification of a delegation chain is linear in the number of entities of the chain. Our implementation needs only one aggregate signature, which is a significant improvement in efficiency over the existing delegation chain protocols.

Although our delegation model is role-based, it can be simplified to support individual delegation, i.e., a role member further extends his or her delegated privileges to another individual. Because role-based delegation is more general and scalable, it is the focus of our presentation in this paper.

B. Organization of the paper

The rest of this paper is organized as follows. We define the terminology and notations used in our role-based cascaded delegation in Section II. The necessary cryptographic knowledge is also described. The overview of our role-based cascaded delegation mechanism is given in Section III. In Section IV, an example of role-based cascaded delegation is presented. Our delegation protocol is described in Section V. A realization of RBCD with aggregate signatures is presented in Section VI. In Section VII, we address the issues of revocation, security, scalability, and efficiency for our model and implementation. A comparison of role-based cascaded delegation with existing trust management approaches is given in Section VIII. Section IX is the conclusion.

II. Definitions and Preliminaries

In this section, we give our definitions for affiliated and delegated roles, and define our terminology and notations. We also give our definition for independently-verifiable decentralized role-based delegation. Finally we describe the necessary cryptographic knowledge.

A. Roles and their scopes

In our model, we define the *administrator* of a role as the organization that creates and manages the role. If a role credential of an entity D is signed and issued by the administrator of the role, that role is said to be an *affiliated role* of D. (This type of role is usually obtained through the affiliation with an organization, and thus the name.) If a role credential of D is instead issued through delegation and signed by entities other than the administrator of the role, that role is called a *delegated role* of D.

The following example illustrates the difference between affiliated and delegated roles. Bob is a full-time professor at University U. He has a credential signed by university U for the role *professor* at U, denoted U*.professor*. Thus, role U*.professor* is an affiliated role of Bob. Alice is not the university's employee, but she is Bob's collaborator. Bob delegates Alice the role U*.professor* to allow her to access university's resources. However, Alice does not have a credential signed by U for *U.professor*. Thus, *U.professor* is a delegated role of Alice.

The reason of making this distinction is to protect sensitive resources and to provide easy management for resource owners. An affiliated role and a delegated role have different access scopes. Delegations to a role r of an organization only apply to those entities who have r as an affiliated role. In the above example, if a privilege is delegated by a third-party to role *U.professor*, then Bob is entitled to this privilege, whereas Alice is not. This is because *U.professor* is Alice's delegated role. The new privileges delegated to role *U.professor* do not automatically propagate to her. A real-life analogy to this distinction is professor vs. visiting professor. In a university, a full-time professor is appointed by the university, whereas a visiting professor position is temporary and typically approved by a full-time professor. A visiting professor has fewer privileges than a full-time professor in terms of access rights.

Our delegation model for roles is different from conventional delegation models, where delegations to a role *automatically* propagate to *all* the entities that are delegated the role. However, it is important to make this distinction and our definitions provide higher assurance to the security of shared resources. For example, if hospital H delegates the right of reading a patient's medical record to the role *U.professor*, Alice would be entitled to this privilege in conventional delegation models, but not in our model. For sensitive data such as medical records, the automatic propagation of delegations to unknown roles may not always be desired by the resource owner. In comparison, our delegation model allows easy management of delegations for resource owners.

To support flexible decentralized delegation, we give to both role types (affiliated and delegated) the capability of delegating the role to other roles. Thus, in the above example, both Bob and Alice are able to delegate role *U.professor* to other roles.

Even though our delegation mechanism is privilege-oriented, it is more efficient than the capability-list style delegations [3] because of the role abstraction. Delegations in our model may be issued to roles, as well as to individuals, which benefit from the efficiency, scalability, and simplicity brought by the role-based delegation. The delegated privileges are role assignments, therefore, role-based cascaded delegation approach is more efficient than the capability-lists.

B. Terminology

As in the *RT* framework [23], we define an *entity* to be either an organization or an individual. An entity may issue credentials and make requests. Also, an entity may have one or more affiliated roles or delegated roles, which are authenticated by role credentials. An *affiliated role credential* is the credential for an affiliated role, and is signed by the administrator of the role. Similarly, a *delegated role credential* is the credential for proving a delegated role. Both credentials are issued using our delegation protocol. An affiliated role can be viewed as delegated directly by the administrator of the role. A *privilege* can be a role assignment or an action on a resource. The *original issuer* or *original delegator* of privilege P is the first entity on a delegation chain, and is the owner of the resources associated with privilege P. A *delegation chain* of privilege P is the path that shows the delegation sequence of P between entities. The chain connects a delegated entity to the original issuer of P. Given an entity on a delegation chain, the preceding entities on the chain are the *ancestor* entities.

An *extension credential* is generated and signed by a delegator on delegation transaction information, such as identities of the delegator and delegatee, and the delegated privilege. An *extension signature* is the signature on an extension credential. A *role signature* of an entity is the signature on an affiliated role credential of the entity. The *identity signature* of an entity is a signature computed by the entity using her private key. A *complete delegation credential* includes the identity signature of the requester, extension signatures, and role signatures. A *partial delegation credential* is a delegation credential issued to a role. It cannot be used by an individual for proving authorization, as it lacks the identity and role signatures of the requester.

We give the definition for independently-verifiable decentralized role-based delegation as follows.

Definition 2.1: A delegation is an independently-verifiable decentralized role-based delegation if and only if the following requirements are satisfied:

1) **[Decentralized]** *Domain*$_1$ and *Domain*$_2$ are two independent administrative domains. Let *Domain*$_1.r$ be a role administrated by *Domain*$_1$.
2) **[Role-Based]** The delegation is issued by a member of role $Domain_1.r$ to delegate privileges associated with r to members of $Domain_2$.
3) **[Independently-verifiable]** Given the public key associated with *Domain*$_1$, the delegation credential can be verified by any third-party without the participation of the administrator of *Domain*$_1$.

We will illustrate further this above definition in later sections.

C. Notations

We use a simple notation to express delegation credential. We allow the delegation of role memberships, and delegation to roles. A role r administered by entity A is denoted as $A.r$. Entity A is the administrator of role $A.r$. A role defines a group of entities who are members of this role. An affiliated role $A.r_a$ defines a subset of a role $A.r$ that contains a group of entities whose role credentials are directly issued by A. Similarly, a delegated role $A.r_d$ defines a subset of the role $A.r$ that contains entities whose role credentials are not directly issued by A. Role $A.r_a$ and $A.r_d$ define $A.r$, i.e. $A.r = A.r_a \cup A.r_d$. If an entity D has an affiliated role $A.r$, his *role credential* is denoted by $A \xrightarrow{A.r} D$, which indicates that D is assigned role $A.r$ by the role administrator A. Entity D can further delegate role $A.r$ to a role $B.s$ (administered by B) by issuing an *extension credential*, which is denoted by $D \xrightarrow{A.r} B.s$. Similarly, any member entity E of role $B.s$ can further delegate role $A.r$ to a role $C.t$ (administered by C). The corresponding extension credential is denoted by $E \xrightarrow{A.r} C.t$.

D. HCBE Preliminaries

Here, we give a brief overview of the necessary cryptographic knowledge.

The *Hierarchical Certificate-based Encryption* (HCBE) scheme [15] is a public key cryptosystem, where messages are encrypted with public keys and decrypted with corresponding private keys. What is unique about HCBE is that it makes the decryption ability of a keyholder contingent on that keyholder's acquisition of a hierarchy of signatures from certificate authorities. To decrypt a message, a keyholder needs both his private key and the public key certificates (signatures) that are respectively signed by a chain of CAs. The CA hierarchy consists of a root CA and lower-level CAs. Higher-level CA certifies the public key of the next-level CAs, and the CAs at the bottom (leaf positions) of the hierarchy certify the public keys of individual users.

HCBE is based on the aggregate signature scheme [6], [8], which supports aggregation of multiple signatures on distinct messages from distinct users into one short signature. The HCBE scheme [15] has six algorithms, HCBE_SETUP, HCBE_CERT_OF_CA, HCBE_CERT_OF_BOB, HCBE_AGGREGATE, HCBE_ENCRYPTION, and HCBE_DECRYPTION. The second and the third algorithms are essentially the same; HCBE_CERT_OF_CA is for certifying the public keys of CAs, and HCBE_CERT_OF_BOB is for certifying the public key of an individual. The API of the algorithms are given below.

HCBE_SETUP: A set of system parameters *params* is generated. Among other parameters, *params* contain two cryptographic hash functions H and H', a bilinear map $\hat{e}$, and a constant π with certain properties. A bilinear map [5] is a mapping function $\hat{e}(x, y)$ that takes two inputs x and y, and outputs a value. Each entity D chooses his private key s_D, computes and publishes his public key $s_D\pi$.

HCBE_CERT_OF_CA$(s_i, \mathit{info}_{i+1})$: CA at i-th level runs this algorithm to certify the public key of the CA at level $i+1$ by computing a signature. The first input is the private key of CA$_i$, and the second input is a string info_{i+1} that contains the public key $s_i\pi$ of the signer and the public key $s_{i+1}\pi$ of CA$_{i+1}$. The string info_{i+1} may also include information such as the expiration date, etc.

HCBE_CERT_OF_BOB$(s_{n-1}, \mathit{info}_n)$: CA$_{n-1}$ runs this algorithm to certify the public key of Bob. The first input is the private key of CA$_{n-1}$, and the second input is a string info_n that contains the public key $s_{n-1}\pi$ of the signer and the public key $s_n\pi$ of Bob.

HCBE_AGGREGATE$(s_n, \mathit{info}', \mathit{sig}_2, \ldots, \mathit{sig}_n)$: This algorithm is run by Bob, who uses his private key s_n and the public key certificates on his chain to compute an aggregate signature, which will be used as his decryption key. The inputs to this algorithm are Bob's private key s_n, the string info' that contains the information of Bob, and a number of signatures [1] that contains the public key certificate signatures associated with his certification chain.

[1]HCBE_AGGREGATE can take any number of signatures.

HCBE_ENCRYPTION$(M, \mathit{info}_1, \ldots, \mathit{info}_n, \mathit{info}')$: Alice computes the ciphertext to send to Bob. The inputs are a message M, string info_i of the certification at level i on Bob's chain for $1 \leq i \leq n$, and string info' that Bob signs in HCBE_AGGREGATE.

HCBE_DECRYPTION(C, S_{Agg}): Bob decrypts the ciphertext C to retrieve the message using his aggregate signature S_{Agg}.

The security of HCBE assures that a ciphertext for an individual can only be correctly decrypted using both the receiver's private key and his public key certificate obtained from the hierarchy of CAs. We slightly modify the encryption and decryption schemes in HCBE scheme for our verification of delegation chain. In our protocol in Section VI, the requester computes an aggregate signature, and gives it to the verifier. The verifier encrypts a message with the delegation chain information, and attempts to decrypt the ciphertext with the aggregate signature. Successful decryption verifies the delegation chain.

III. OVERVIEW OF ROLE-BASED CASCADED DELEGATION

We propose a model for the delegation of authority in role-based trust management systems, called *role-based cascaded delegation*. The main goal of this model is to allow flexible transfer of privileges and sharing of resources in decentralized environments. Our model allows a role member to delegate his or her privileges to users who may belong to different organizations, as opposed to restricting this delegation ability to role administrators in traditional access control models. In addition, our role-based cascaded delegation model allows a delegatee to further extend the delegated privileges to other collaborators. The challenge arise in realizing this goal in a decentralized environment is that the public key of an intermediate delegator may not be known by a verifier or the resource owner. Therefore, the delegation credential signed by that delegator may not be trusted by the verifier.

To solve this problem, we borrow the concept of cascaded delegation from distributed systems literature [25], [31]. The distributed cascaded delegation problem is essentially to design a delegation mechanism that efficiently verifies a hierarchical delegation chain. In the cascaded delegation model, a delegation recipient E may further extend the delegated privilege to another entity E', and the delegation credentials of E are passed to entity E' along with the delegation certificate signed by E as the issuer. The public key of the next delegatee is encoded in the delegation credential, which naturally forms a chain of trust. Therefore, trusting the original delegator means that the delegatees' public keys are authorized by the delegation. In addition, the authorization chain is stored in delegation credentials and does not have to be dynamically discovered. However, previous cascaded delegation protocols support neither multiple administrative domains nor the use of roles in the delegation. We give support to both in our role-based cascaded delegation model.

In our role-based cascaded delegation, given a privilege, two types of entities can delegate the privilege to others. One is the resource owner of the privilege. The other is a member of a role who is delegated the privilege. A role r

is delegated a privilege by receiving a delegation credential C that explicitly assigns the privilege to role r. Members of the role r are allowed to further delegate the privilege to another role r' as follows. A member D of the role r uses the delegation credential C to generate a delegation credential C'. C' comprises multiple component credentials, which include the credential of the current delegation authorization, the credential C from the preceding delegation, and the role membership credential of the delegator D. The verifier can make the authorization decision based on delegation credential C' and the role membership credential of the requester. The verification can be done by any party without the participation of any role administrators, which is called by us as independent verifiability (See also Section II).

The length of a delegation chain in role-based cascaded delegation refers to the number of delegators involved. A privilege P is delegated by an entity E to a role r_1. A member D of role r_1 further delegates the same privilege P to role r_2. The delegation chain of privilege P involves entity E, role r_1, entity D, and role r_2. Role r_2 receives the privilege P as the result of the delegation chain. The length of the chain is two.

Decentralized role-based delegation allows users from administratively independent domains to be dynamically joined according to the needs of the tasks. We have also explored the applications of RBCD for efficient and flexible trust establishment in decentralized and pervasive environments in [34].

IV. RBCD Example

In this section, we describe a delegation example for the role-based cascaded delegation model. Suppose a collaboration project is established between a hospital H and a medical school M. To facilitate the collaboration, the hospital initiates a delegation chain and delegates its role H.*guest* to the affiliated role M.*professor* at the medical school. Hospital H is the administrator of the role H.*guest*. The delegation is expressed in the partial delegation credential (1), using the notation described in Section II.

$$H \xrightarrow{H.\mathit{guest}} M.\mathit{professor} \tag{1}$$

In credential (1), hospital H is the original issuer, H.*guest* is the delegated privilege, and M.*professor* is the role that receives the delegation.

The hospital H allows members of the role M.*professor* to further delegate H.*guest* role to whomever they deem necessary to accomplish the project. Bob is a professor at M and has an affiliated role credential (2).

$$M \xrightarrow{M.\mathit{professor}} \text{Bob} \tag{2}$$

For a task in the collaboration project, Bob subcontracts to a lab L. Lab L is independent from school M and is unknown to the hospital H. Lab L defines a research assistant role L.*assistant*. In order for members of the role L.*assistant* to work on the task and utilize the resources of the hospital H, Bob delegates the role H.*guest* to the affiliated role L.*assistant*. In our role-based cascaded delegation model, Bob issues a partial delegation credential (3) by extending the delegation credential (1) to role L.*assistant*.

$$(H \xrightarrow{H.\mathit{guest}} M.\mathit{professor}), (M \xrightarrow{M.\mathit{professor}} \text{Bob}),$$
$$(\text{Bob} \xrightarrow{H.\mathit{guest}} L.\mathit{assistant}) \tag{3}$$

Credential (3) also includes Bob's role credential (2) for proving that he is allowed to delegate H.*guest*. (3) is a partial delegation credential for role L.*assistant*.

Recall that (3) is different from the linked role in *RT* framework [22], as the role H.*guest* is delegated, not M.*professor*. Alice is a research assistant in lab L, and has an affiliated role credential (4) issued by lab L to prove this role membership.

$$L \xrightarrow{L.\mathit{assistant}} \text{Alice} \tag{4}$$

(4) is equivalent to the role membership representation below, as in *RT* framework. (5) is read as Alice has a role of L.*assistant*.

$$L.\mathit{assistant} \leftarrow \text{Alice} \tag{5}$$

Because (4) is issued by the lab L, the role L.*assistant* is Alice's affiliated role. To prove that she has the hospital's delegated *guest* role, Alice obtains the delegation credential (3) for role L.*assistant* from a credential server, and aggregates it with her affiliated role credential (4). This delegation generates credential (6).

$$(H \xrightarrow{H.\mathit{guest}} M.\mathit{professor}), \quad (M \xrightarrow{M.\mathit{professor}} \text{Bob}),$$
$$(\text{Bob} \xrightarrow{H.\mathit{guest}} L.\mathit{assistant}), \quad (L \xrightarrow{L.\mathit{assistant}} \text{Alice}) \tag{6}$$

Credential (6) and the identity signature of Alice yield a complete delegation credential for Alice. For verification, the hospital H does not need to discover the delegation chain that connects Alice with role H.*guest*, because this information is contained in credential (6). Furthermore, the lab administrator does not have to participate in the verification of Alice's role membership as this information is also in (6). The hospital makes the authorization decision by verifying each component of credential (6) and Alice's identity signature. Note that the hospital does not need to have prior knowledge of or trust relationship with lab L. This independent verifiability enables a cross-domain authorization chain to be easily verified.

We allow actions to be delegated, as well as roles. For example, the hospital may delegate the read access of a database *db* (*Read db*) to role M.*professor*, which is expressed in (7).

$$H \xrightarrow{(\mathit{Read}\ \mathit{db})} M.\mathit{professor} \tag{7}$$

V. Role-based Cascaded Delegation Protocol

In this section, we first describe the role-based cascaded delegation protocol and then show an efficient realization of this protocol using the HCBE scheme [15]. In what follows, a role r represents an affiliated role.

A. Protocol

The role-based cascaded delegation protocol defines four operations: RBCD_Initiate, RBCD_Extend, RBCD_PROVE, and RBCD_VERIFY. In our protocol description, delegation credentials once issued are stored in public credential servers that can be queried by anyone. The credential servers (See also Section VII-B) may be simple LDAP servers. Because of our security guarantees (See Section VII-A, adversaries cannot use the credentials on the servers to forge authorization.

- RBCD_INITIATE(P_{D_0}, s_{D_0}, $D_0.priv$, $A_1.r_1$, P_{A_1}): This operation is run by the administrator D_0 of a privilege $D_0.priv$ to delegate $D_0.priv$ to an affiliated role $A_1.r_1$. This operation initiates a delegation chain for privilege $D_0.priv$. Inputs are the public key P_{D_0} of entity D_0, the corresponding private key s_{D_0}, the delegated privilege $D_0.priv$, the role name $A_1.r_1$, and the public key P_{A_1} of role administrator A_1. Recall that only affiliated roles can receive delegations, as discussed in Section II-A. The output is a partial delegation credential C_1 for the role $A_1.r_1$, represented as
$$D_0 \xrightarrow{D_0.priv} A_1.r_1.$$
The statement of C_1 includes the public key P_{D_0}, the privilege $D_0.priv$, and information about the role $A_1.r_1$ such as the role name and the public key of the administrator A_1. The delegation certificate is signed using the private key s_{D_0}. D_0 stores C_1 on a credential server.
Note that if the last argument is the public key of an individual, this operation can also be used for generating role certificates. Role certificate is given to the corresponding role member.
- RBCD_EXTEND ($s_{D_n}, D_0.priv, C_n, R_{D_n}$, $A_{n+1}.r_{n+1}$, $P_{A_{n+1}}$):
This operation is run by an intermediate delegator D_n, who is a member of an affiliated role $A_n.r_n$, to extend the delegation of privilege $D_0.priv$ to the role $A_{n+1}.r_{n+1}$. The inputs are the private key s_{D_n} of the delegator D_n, the delegated privilege $D_0.priv$, the partial delegation credential C_n that delegates the privilege $D_0.priv$ to the role $A_n.r_n$, the role credential R_{D_n} of the delegator D_n, the role name $A_{n+1}.r_{n+1}$, and the public key $P_{A_{n+1}}$ of role administrator A_{n+1}. Credential C_n is retrieved from a credential server. The partial delegation credential C_n is a function of the preceding extension and role credentials, which are denoted as:
$$\begin{aligned} &(D_0 \xrightarrow{D_0.priv} A_1.r_1), \\ (A_1 \xrightarrow{A_1.r_1} D_1), \quad &(D_1 \xrightarrow{D_0.priv} A_2.r_2), \\ &\cdots \\ (A_{n-1} \xrightarrow{A_{n-1}.r_{n-1}} D_{n-1}), \quad &(D_{n-1} \xrightarrow{D_0.priv} A_n.r_n) \end{aligned}$$
where D_0 represents the resource owner, and $A_i.r_i$ is the role that is delegated the privilege $D_0.priv$ by an entity D_{i-1} who has the affiliated role $A_{i-1}.r_{i-1}$, for $i \in [1, n]$.
An extension credential denoted by $D_n \xrightarrow{D_0.priv} A_{n+1}.r_{n+1}$ is generated as an intermediate product of the operation RBCD_EXTEND. Its statement contains information about the delegated privilege $D_0.priv$ and the role $A_{n+1}.r_{n+1}$. It is signed with the private key s_{D_n}. The final output of this operation is a partial delegation credential C_{n+1}, which is a function of the credential C_n, the role credential R_{D_n} denoted by $A_n \xrightarrow{A_n.r_n} D_n$, and the extension credential described above.
Credential C_{n+1} may simply be delegation credential C_n together with two individual credentials. Alternatively, D_n can compute a delegation credential for the role $A_{n+1}.r_{n+1}$ as in existing cascaded delegation protocols [11], [27], and also passes down his role credential to members of the role $A_{n+1}.r_{n+1}$. In comparison, our realization using HCBE [15] scheme provides a more efficient approach.
- RBCD_PROVE($s_{D_n}, D_0.priv, R_{D_n}, C_n$):
This operation is performed by the requester D_n who wants to exercise privilege $D_0.priv$. D_n is a member of the affiliated role $A_n.r_n$. The requester D_n uses the partial delegation credential C_n and D_n's affiliated role credential R_{D_n}, denoted by $A_n \xrightarrow{A_n.r_n} D_n$, to prove that he is authorized the privilege $D_0.priv$. The inputs are the private key s_{D_n} of the requester D_n, the privilege $D_0.priv$, the affiliated role credential R_{D_n} of the requester, and the delegation credential C_n. Credential C_n is retrieved by the requester from a credential server. The operation produces a proof F, which contains delegation statements and corresponding signatures for verification. The private key s_{D_n} is for proving the authenticity of the public key P_{D_n} that appears on the role credential R_{D_n} of the requester.
- RBCD_VERIFY(F):
This operation is performed by the resource owner D_0 to verify that the proof F produced by the requester D_n correctly authenticates the delegation chain of privilege $D_0.priv$. D_n is a member of the role $A_n.r_n$. The input is a proof F that is computed by the requester D_n. F contains signatures and a string tuple $[D_0.priv, P_{D_0}, A_1.r_1, P_{A_1}, P_{D_1}, \ldots, P_{D_{n-1}}, A_n.r_n$, P_{A_n}, $P_{D_n}]$ that consists of the components of a delegation chain for requester D_n. In the string tuple, $D_0.priv$ is the delegated privilege, for $i \in [1, n]$ $P_{D_{i-1}}$ is the public key for the delegator D_{i-1} whose affiliated role is $A_{i-1}.r_{i-1}$, $A_i.r_i$ is the role that receives the delegation from D_{i-1}, P_{A_i} is the public key of role administrator A_i, and P_{D_n} is the public key of the requester. The verifier checks whether the signatures in F correctly authenticates the delegation chain. This process includes the authentication of each delegation extension $D_{i-1} \xrightarrow{D_0.priv} A_i.r_i$, and entity D_i's affiliated role membership $A_i \xrightarrow{A_i.r_i} D_i$, for all $i \in [1, n]$. F also contains the proof of possession of private key s_{D_n} that corresponds to public key P_{D_n}. D_n is granted $D_0.priv$ if the verification is successful, and denied if otherwise.

Our role-based cascaded delegation model supports independently-verifiable decentralized role-based delegation. Recall that independently-verifiable decentralized role-based delegation is defined in Section II as the ability for a member of role r to delegate r to other roles or entities, and in addition the delegation credential can be independently verified by any third-party without the participation of the administrator of role r. In RBCD, RBCD_EXTEND is performed by a valid member of role r to delegate r to others. The partial delegation credential generated contains the role credentials of all delegators on the delegation chain. Therefore, the verification of the delegation credentials does not require any role administrators, and can be performed by anyone.

Affiliated role credentials can be issued using RBCD_INITIATE operation by the administrator of a role. RBCD_EXTEND operation is used to issue delegated role credentials. The delegation chain of a privilege grows at each delegation extension. The verifier may perform revocation checking at the RBCD_VERIFY operation. Delegation revocation is discussed in Section VII. In the next section, we describe a realization of cascaded delegation using the Hierarchical Certificate-based Encryption [15], which allows aggregation of multiple credentials into one credential.

VI. REALIZATION

Role-based cascaded delegation can be implemented in a straightforward manner using the RSA signature scheme [28]. At each delegation, the delegator D computes an RSA signature on the delegation statement, and issues it to delegatees along with D's role signature (also an RSA signature). The delegation chain verification consists of verifying each of the above signatures.

We present a more efficient realization of role-based cascaded delegation using the Hierarchical Certificate-based Encryption (HCBE) [15] scheme. In HCBE, each entity has a public/private key pair generated on his own. A member of an affiliated role has an affiliated role credential, which contains a signature signed by the administrator of the role. The delegation credential in this protocol consists of an aggregate signature and a string tuple.

In RBCD, a delegator issues a partial delegation credential to a role, which is not valid until a member of the affiliated role adds in his role credential and identity information. The complete delegation credential of an entity is computed by the entity, using the partial delegation credential obtained through credential servers, his role credential, and his secret personal information. Each member of an affiliated role has a *unique* complete delegation credential, however, the delegator only needs to generate *one* partial delegation credential, which does not require the knowledge of the members of that affiliated role. This feature makes our protocol scalable. Any member of that affiliated role can further delegate the privilege to other affiliated roles, without any assistance from administrators. The public information of intermediate delegators is traceable. The affiliated role membership of all the delegators on a delegation chain can be proved, however, the signatures on their role credentials are not revealed to anyone.

A delegation credential of an entity corresponds to a delegation chain, and has two components: one aggregate signature of constant size and a string tuple. The string tuple defines the delegation chain, and its size is linear in the length of the chain. The signature is used for authentication of the chain. The aggregate signature [6] in the HCBE scheme is an ordinary sized signature that is the aggregation of multiple signatures, which may include signatures from delegators, role administrators, and the requester. To request a service, the requester uses his private key to sign a statement which is chosen by the verifier, and aggregates this signature with signatures from his role credential and the partial delegation credential obtained from a credential server. To verify the delegation chain, one simply verifies that aggregate signature submitted by the requester.

Our role-based cascaded delegation protocol has five operations, which make use of the algorithms in the HCBE scheme [15]. Alternatively, one can use operations in the aggregate signature scheme [6] for generating and verifying delegation credentials. We choose to use HCBE for the presentation, because its operations have intuitive meanings that are similar to our needs.

RBCD_SETUP: This operation outputs the system parameters, public/private keys, and role credentials that will be used in the system.

- The root of the system calls HCBE_SETUP and obtains a set of public parameters denoted as *params*. Among other parameters in *params*, including collision-resistant hash functions H and H', a special constant π, and a bilinear map $\hat{e}$ [5].
- Each entity (organization or individual) D chooses a secret s_D as his private key, and computes the product $s_D\pi$ as its public key P_D.
- An organization A with the private key s_A certifies entities who have $A.r$ as an affiliated role. For each entity D who has the affiliated role $A.r$ and the public key P_D, organization A computes a role signature R_D by running HCBE_CERT_OF_CA($s_A, P_D \| A.r$), where $\|$ denotes string concatenation. The output signature, representing the role assignment $A \xrightarrow{A.r} D$, is given to entity D for proving the affiliated role membership.

RBCD_INITIATE: Resource owner D_0 delegates the privilege $D_0.priv$ to members of an affiliated role $A_1.r_1$. The private key s_{D_0} corresponds to the public key P_{D_0} of entity D_0. Entity D_0 does the following.

- Set the string $\textit{info}_1 = P_{D_0} \| D_0.priv \| A_1.r_1 \| P_{A_1}$. String $\textit{info}_1$ contains the public key P_{D_0} of the owner of the delegated privilege, the delegated privilege $D_0.priv$, the role $A_1.r_1$ that receives the privilege, and the public key P_{A_1} of the administrator of the role $A_1.r_1$. Run HCBE_CERT_OF_CA($s_{D_0}, \textit{info}_1$), which outputs an extension signature X_1. Define a string tuple $\textit{chain}_1$ as $[D_0.priv,\ P_{D_0},\ A_1.r_1,\ P_{A_1}]$. Set the partial delegation credential C_1 for the role $A_1.r_1$ as $(X_1, \textit{chain}_1)$. Credential C_1 is put on a credential server.

RBCD_EXTEND: An entity D_i, whose role is $A_i.r_i$, further delegates $D_0.priv$ to role $A_{i+1}.r_{i+1}$. D_i uses his private key s_{D_i}, his role signature R_{D_i}, and the delegation credential C_i of the role $A_i.r_i$ to compute a partial delegation credential C_{i+1}. Entity D_i does the following.

- Parse the credential C_i as $(S_{Agg}, \textit{chain}_i)$, where S_{Agg} is the aggregate signature of credential C_i and $\textit{chain}_i$

is the corresponding string tuple. Signature S_{Agg} is a function of preceding extension and role signatures on the delegation chain. String tuple chain_i contains the components of the delegation chain. Set the string $\mathit{info}_{i+1} = P_{D_0} \| D_0.priv \| A_{i+1}.r_{i+1} \| P_{A_{i+1}}$, where P_{D_0} is the public key of the resource owner of the delegated privilege, $D_0.priv$ is the delegated privilege, $A_{i+1}.r_{i+1}$ is the role that receives the privilege, and the public key $P_{A_{i+1}}$ of the role administrator A_{i+1}. Run HCBE_AGGREGATE$(s_{D_i}, \mathit{info}_{i+1}, R_{D_i}, S_{Agg})$, which outputs an aggregate signature S'_{Agg}.

- Define the string tuple chain_{i+1} of credential C_{i+1} as the string tuple chain_i appended with public key P_{D_i}, the role name $A_{i+1}.r_{i+1}$, and the public key $P_{A_{i+1}}$. Set credential C_{i+1} as $(S'_{Agg}, \mathit{chain}_{i+1})$. The partial delegation credential C_{i+1} for the role $A_{i+1}.r_{i+1}$ is put on a credential server.

RBCD_PROVE: The requester D_n with the role signature R_{D_n} and delegation credential C_n wants to use the delegated privilege $D_0.priv$. D_n is given a random message T by the verifier D_0. The message T contains some random information to prevent a replay attack. D_n does the following.

- Parse the credential C_n as $(S_{Agg}, \mathit{chain}_n)$, where S_{Agg} is the aggregate signature of C_n and chain_n is the string tuple. Run HCBE_AGGREGATE$(s_{D_n}, T, R_{D_n}, S_{Agg})$, where s_{D_n} is the private key of D_n. HCBE_AGGREGATE outputs an aggregate signature S'_{Agg}. Set the string tuple chain'_n to be chain_n appended with the public key P_{D_n} of D_n. Set the proof F to be $(S'_{Agg}, \mathit{chain}'_n, T)$, which is sent to the verifier D_0.

RBCD_VERIFY: The verifier D_0 verifies the proof F submitted by the requester D_n as follows.

- Parse F as $(S'_{Agg}, \mathit{chain}'_n, T)$, where S'_{Agg} is an aggregate signature, chain'_n is a string tuple, and T is a message. Parse the string tuple chain'_n as $[D_0.priv,\ P_{D_0},\ A_1.r_1,\ P_{A_1}, \ldots, A_n.r_n,\ P_{A_n},\ P_{D_n}]$, where for $i \in [0, n-1]$ P_{D_i} is the public key of delegator D_i whose affiliated role is $A_i.r_i$, $A_{i+1}.r_{i+1}$ is the role receiving the delegation from D_i, $P_{A_{i+1}}$ is the public key of role administrator A_{i+1}, and P_{D_n} is the public key of the requester.
- Encrypt a message M in HCBE as follows. Choose a random number r. Set the ciphertext $\mathit{Ciphertext} = [r\pi, V]$, where π is one of the public parameters, $V = M \oplus H'(g^r)$, where $g = g_1 g_2 g_3$ is a product of the following: $g_1 = \hat{e}(P_{D_n}, H(T))$, $g_2 = \Pi_{i=1}^{n} \hat{e}(P_{A_i}, H(P_{D_i} \| A_i.r_i))$, $g_3 = \Pi_{i=0}^{n-1} \hat{e}(P_{D_i}, H(P_{D_0} \| D_0.priv \| A_{i+1}.r_{i+1} \| P_{A_{i+1}}))$. The value g is the product of multiple bilinear map functions [5] whose inputs are the public key of a signer and the hash digest of the signed message. H and H' are the two hash functions in the system parameters $params$. $\oplus$ denotes bit-wise XOR operation. T is the message that D_n signs in RBCD_PROVE.
- Run HCBE_DECRYPTION$(\mathit{Ciphertext}, S'_{Agg})$ to decrypt ciphertext $\mathit{Ciphertext}$ using S'_{Agg}. Compare the output M' of the decryption with the original message M. The request is granted if $M = M'$, denied if otherwise.

The correctness of the protocol can be directly deduced from the correctness of HCBE and is not shown here.

A delegation to the intersection of roles [22], for example $A_1.r_1 \cap A_2.r_2$, may be realized by extending one delegation to a string that represents an intersection of roles, rather than one role. To extend or prove such a delegation, an entity needs to aggregate two, rather than one, role signatures into a delegation credential. Additional fields can be added by the delegator to a delegation credential to increase the expressiveness, one of them being the expiration date of a delegation. Given a delegation chain defined by the credential, the expiration date of a delegation should be no later than any of the expiration dates of preceding delegations. The delegator may also set restrictions on the level of a delegation, which specifies whether or not the privilege can be further delegated and for how many times, i. e., the length of a delegation chain. This constraint helps improve the accountability, and gives the delegator a tight control over the delegated privileges. The verifier or the delegation receiver should check if all the constraints are satisfied before accepting a credential. Supporting the RBCD model with predicates and constraints was recently presented in [34].

We discuss the security, efficiency, and scalability of role-based cascaded delegation protocol in the next section.

VII. ANALYSIS

We now analyze the security, efficiency, scalability, and revocation of role-based cascaded delegation.

A. Security

In this section, we first analyze the security of our role-based cascaded delegation model, and then describe the security of the RBCD realization with aggregate signatures.

The security property of the RBCD model is defined as follows: unauthorized entities cannot access protected resources, and unauthorized entities cannot issue valid delegations. We allow adversaries to do the following: (1) adversaries can observe communications between delegation participants and between resource owners and requester; (2) adversaries can forge delegation credentials or role credentials; and (3) adversaries can submit access requests. We assume the existence of a signature scheme that is secure against forgery attacks by (probabilistic) polynomial-time adversaries.

Then in RBCD, given a partial delegation credential for a role r, a polynomial-time adversary cannot forge a valid delegation chain that authorizes the role r to any role or individual. The analysis is as follows. The partial delegation credential is generated by INITIATE or EXTEND operations. A partial delegation credential is issued to roles, rather than to individuals. To use the partial delegation credential for role r to request for access, one needs to have a valid role credential R_r of role r and the private-key corresponding to the public-key stated in R_r. The latter is for signing the challenge nonce from the resource owner. Given any secure signature scheme against polynomial-time adversaries, an adversary cannot forge role credential and the signature on the nonce. Therefore, she cannot use the partial delegation credential to authorize the role r to herself. In addition, an adversary cannot forge valid extension credentials to extend role r, because she is unable to forge a valid role credential of role r that is required in EXTEND operation.

The RBCD realization with aggregate signatures provides strong protection of sensitive signatures because individual signatures can be verified without being disclosed. To extend a delegation, an intermediate delegator aggregates two signatures. One is his role signature signed by a role administrator, and the other is the extension signature signed by the delegator himself. Once the role signature and the extension signature are aggregated with the signature from the previous delegation (the order does not matter), it is impossible for others to find out what the role signature or the extension signature is. Similarly, for a requester, the role signature and the signature on a challenge statement are also protected. This is not achievable in conventional signature schemes, such as RSA [7].

Furthermore, the security of the aggregate signature and HCBE schemes guarantees that an attacker cannot forge a valid aggregate signature consisting of n individual signatures, even if he possesses $n - 1$ of the required private keys [6]. In our delegation model, this implies that one cannot forge any valid delegation credential from existing credentials. Although signature verification can be performed by anyone, an adversary cannot derive any signature nor secret key of the preceding delegators from the aggregate signature that is issued to him. HCBE also guarantees that collusions between users do not give them any information more than what they have already known.

B. *Scalability*

The abstraction of roles in role-based cascaded delegation greatly reduces the potential for a large number of delegation credentials, and makes the model scalable. Because the partial delegation credentials issued by the delegators cannot be directly used for accessing resources, they may be stored at credential servers so that members of a role can query the server to retrieve the partial credential. Thus, our implementation scales up to a large number of credential receivers. Also, the delegation is decentralized. Individuals, who have qualified roles, can make delegations of the roles without the assistance of administrators. In collaboration environments where coalitions are formed dynamically, this feature greatly facilitates resource sharing. Note that our model does not require the existence of public-key infrastructure.

C. *Efficiency*

We analyze the efficiency of RBCD model, and compare its realizations with RSA and aggregate signatures. The size of a partial or complete delegation credential is linear in the length of a delegation chain, which is the number of delegation transactions associated with the delegation credential. This complexity is because at each delegation transaction, one extension credential and one role credential are accumulated to existing delegation credentials.

Although the asymptotic sizes of delegation credentials in different RBCD realizations are the same, the implementation using HCBE and aggregate signatures can have significant advantages in delegation efficiency, compared to an implementation using conventional credentials. We compare our HCBE-based realization with the realization using the RSA signature scheme [28] described at the beginning of Section VI. We consider a 1024-bit modulus RSA scheme, in which the size of the public key is slightly larger than 1024 bits and the size of a signature is 1024 bits long.

For the same level of security as 1024-bit modulus RSA, the signatures and public keys in the aggregate signature scheme can be as short as 170-bit long [8]. Observe that at each delegation extension of RBCD, the following information needs to be added to the delegation credential: the public key of the delegator, the role name of recipients, the public key of the role administrator, the signature on the role credential of the delegator, and the extension signature generated by the issuer. The analysis also applies to the AGGREGATE operation performed by the requester. Therefore, to authenticate a delegation chain of length n (i.e. having n delegations), the information required by the verifier includes the delegated privilege, the public keys of n delegators and n role administrators, n role names, the public key of the requester, along with $2n + 1$ digital signatures.

Suppose the length of a role name is 100 bits and the delegated privilege has the same size as a role name. The total size of the credential in our HCBE realization is $170 + 170(2n + 1) + 100(n + 1) = 440n + 440$ bits. For the RSA signature scheme, such a delegation credential contains $2n$ additional signatures, and the total size is at least $1024(2n + 1) + 1024(2n + 1) + 100(n + 1) = 4196n + 2148$ bits.

For example, consider a delegation chain of length 20. The size of the delegation credential in RSA is more than 86 Kbits, while in the HCBE realization it is about 9.2 Kbits. Smart cards with a microprocessor typically have 32 KBytes (256 Kbits) EEPROM storage. Thus, our approach has a clear advantage in terms of the number of credentials that can be stored by smart cards and similar devices. For small mobile devices with limited communication bandwidth, this saving in the credential size allows the information to be transmitted faster. The above analysis also applies to the EXTEND operation.

For a 20 Kbits per second connection and a delegation chain of length 20, the time for transmitting the entire RSA credentials to the verifier in the PROVE operation takes $(4196 \times 20 + 2148)/20000 = 4.30$ seconds. The time in our HCBE realization takes $(440 \times 20 + 440)/20000 = 0.46$ seconds. In addition, generating a signature in HCBE scheme requires only 3.57 ms on a 1 GHz Pentium III, and is faster than generating a signature in the RSA scheme, which requires 7.90 ms for a 1007-bit private key on the same machine [1].

The running time for verifying an aggregate signature associated with a delegation chain is linear in the number of single signatures aggregated, i.e., the length of the chain. The verification of a signature in the HCBE scheme is slow (about 50 ms on a 1 GHz Pentium III) compared to RSA signature verification (0.40 ms on the same machine for a 1007 bits private key) [1]. Nevertheless, in our protocol only the servers of resource owners, which are typically powerful, have to performs delegation chain verifications.

Table I summarizes the analysis above.

D. Delegation renewal and revocation

At each delegation extension, the issuer can set an expiration date for the delegation, which may be earlier than the expiration dates of preceding delegations on the chain. For a delegation credential to be considered valid, none of the expiration dates has passed. Intermediate delegators may issue delegations with a short validity period, and then periodically renew them. Delegation renewal can be done in a hierarchical fashion as follows. To renew a delegation, a delegator E puts the renewed partial delegation credential on credential servers. Intermediate delegators that succeed to E may retrieve the renewed credential and update the corresponding delegations that are issued by them.

Delegation revocation before expiration can be handled by maintaining a revocation service, which can be efficiently achieved using the authenticated dictionary technique (see, e.g., [10], [17], [18], [26]). An authenticated dictionary is a system for distributing data and supporting authenticated responses to queries about the data. The data originates at a secure central site (the repository) and is distributed to servers scattered around the network (responders). The responders answer queries about the data made by clients on behalf of the repository and provide a proof of the answer.

The roles or public keys whose delegated privileges are revoked are put on the repository of the revocation service by the resource owner. Before verifying the credential signatures in the VERIFY operation, the resource owner queries the revocation service to ensure that no public key whose delegated privileges are revoked appears on the delegation credential. Similarly, the revocation of affiliated role memberships can also be supported using a revocation service, which the verifier queries in the VERIFY operation to ensure the validity of the affiliated role memberships of intermediate delegators.

VIII. Related Work

The PolicyMaker [4] and KeyNote [3] are the first trust management systems that authorize decentralized access by checking a proof of compliance. The system puts all the policy and credential information into signed certificates that are programmable. Certificates in PolicyMaker are generalized as they consist of programs written in a general programming language. SPKI/SDSI (Simple Public Key Infrastructure/Simple Distributed Security Infrastructure) is a public-key infrastructure emphasizing decentralized name space and flexible authorization [9], [13]. The owner of each public key can create a local name space relative to that key. These name spaces can be linked together to enable chains of authorization and define groups of authorized principals. To access a protected resource, a client must show a proof that takes the form of a certificate chain proving that the client's public key is one of the groups on the resource's ACL, or that the client's public key has been delegated authority from a key in one of the groups on the resource's ACL. Due to the flexible naming and delegation capabilities of SPKI/SDSI certificates, finding such a chain can be nontrivial.

Compared to RBCD, PolicyMaker, KeyNote, and SPKI/SDSI do not define role abstractions, and thus delegations can only be issued to individuals. The use of roles makes authorization scalable, and in the meantime, the role-based delegation mechanism is more complex as demonstrated in this paper. In addition, these systems assume that all certificates are centrally stored, which may not be realistic in decentralized environments. In comparison, we address this issue with a simple accumulation approach by having delegators to pass down relevant credentials.

The *RT* framework is a family of Role-based Trust management languages for representing policies and credentials in decentralized authorization [22]. Compared to our work, the work of *RT* focuses on the high-level expressiveness aspect of trust management, and does not address the cryptographic verification problem of authorization chains as studied in this paper. Our delegation mechanism is general and can be incorporated into existing role-based trust management systems such as *RT* to instantiate a concrete delegate mechanism. Details of how this incorporation is done is out of the scope of this paper.

As we said earlier in the introduction, our role-based delegation can be simplified to support individual delegation, i.e., a role member further extends his or her delegated privileges to another individual. Therefore, TM systems such KeyNote, PolicyMaker, and SPKI/SDSI can also utilize our protocol to instantiate their delegation mechanisms.

QCM [19] and SD3 [20] are two trust-management systems that consider distributed storage of credentials. A limitation of the approach in QCM and SD3 is assuming that issuers initially store all the credentials, which may be impractical for some applications. This limitation was addressed by Li *et al.* [23], who presented goal-directed credential chain discovery algorithms that support a more flexible distributed storage scheme in which credentials may be stored by their issuer, their recipient (also called their "subject"), or both. The algorithms dynamically search for relevant credentials from remote servers to build a proof of authorization. While storing credentials with their issuers or recipients supports flexible delegation models, in many cases such flexibility is unnecessarily costly. The discovery algorithms require delegation issuers or their responders (credential servers) to participate in the computation. Role-based cascaded delegation can be integrated with the credential chain discovery algorithms to reduce the communicational and computational costs to a certain degree [34]. This is because part of the target authorization chain is already captured in RBCD's delegation credentials and does not need to be discovered.

There are several cryptographic cascaded delegation [31] schemes for the proxy authentication and authorization, including nested signature schemes [33], delegation keys [27], and a combined approach [11]. These schemes do not provide the support for delegations to roles, and the delegation credentials are not as compact as ours, as is explained in the following. Nested signatures define the order of delegations on a delegation path. They are used to prevent the attacker to

Chain length $n = 20$	Credential size	Transmission (20 Kbit/s)	Signing [1]	Verifying [1]
RBCD using RSA	86 Kbits	4.3s	7.9ms	0.4ms
RBCD using Agg. Sig.	9.2 Kbits	0.46s	3.57ms	50ms

TABLE I

EFFICIENCY COMPARISONS BETWEEN RBCD REALIZATIONS USING RSA SIGNATURES AND BILINEAR-MAP BASED AGGREGATE SIGNATURES

construct another delegation path using one of the delegation credential [33]. The size of delegation credential is linear to the number of entities on a delegation chain, and verification of signatures is done sequentially. Cascaded delegation is also implemented by binding two delegation credentials using delegation keys [27]. Applying this scheme to role-based delegation means sharing secret group key among members of a role, which may cause accountability problem. The hierarchical delegation protocol by Ding *et al.* [11] combines the nested signature scheme and delegation public/private key approach. It is based on Schnorr signature scheme [29], self-certified public keys [16], and the concept of hierarchical key generation [14]. Compared to our realization using HCBE, their delegation and verification algorithms require more computations. In their scheme, to verify one hierarchical delegation credential of length l, a verifier has to compute and verify l public delegation keys (different from public keys in conventional PKI). In addition, at each delegation the delegation receiver is required to perform a number of computations. In our scheme, a delegated entity is not required to perform any computation.

The security framework for Java-based computing environment in [31] uses roles in chained delegations to simplify the management of privileges. However, their delegations are made to individuals rather than to roles. The framework does not support tracing the delegation credentials of intermediate entities on the delegation chain, therefore does not support the verification of delegation chains. Their term cascaded delegation has different meanings from ours, and refers to delegations where all the privileges of preceding entities on the chain are inherited by the delegatee. In our model, only the specified privilege is delegated throughout a delegation chain.

Permission-based delegation model (PBDM) built on RBAC supports user-to-user, role-to-role delegations [35]. A delegator creates one or more temporary delegation roles and assigns delegatees to particular roles. Delegations in PBDM requires changes of role hierarchies by the proper authority, for example, a project leader who has write access to the role assignment and access policies. PBDM does not address decentralized delegation, which is the main focus of this paper.

X-GTRBAC Admin [2] is an administration model for policy administration within a large enterprise. It specifies the user-to-role and permission-to-role assignments in the XML-based Generalized Temporal Role Based Access Control (X-GTRBAC) framework [21]. X-GTRBAC Admin supports decentralized administration by distributing assignment tasks to multiple domains within the enterprise while enforcing temporal constraints. In comparison, our RBCD models aim at the decentralized trust management among members of independent organizations. Therefore, X-GTRBAC Admin is complementary to RBCD models.

Shehab, Bertino, and Ghafoor recently propose a distributed secure interoperability framework for mediator-free collaboration environments [30]. They define secure access paths for dynamic collaboration environment, and also give a path authentication technique for proving path authenticity. Their idea of exploring trust paths in multi-domain environment is similar to the authentication of delegation chains in RBCD. The main difference of their work from ours is that they focus on the domain-level authentication, as opposed to authentication of individual role members.

IX. CONCLUSIONS

We have studied cross-domain role-based delegation problem for information sharing where there is no central administrator. The main challenge addressed in this paper is the verification of role-based authorization chains in decentralized environments, which has not been much studied in existing literatures. We have presented a role-based cascaded delegation model and its associated cryptographic operations for the purpose of convenient verification of delegation chains. RBCD enables a role member to create delegations based on the need of collaboration, yet in the meantime a delegation chain can be verified by anyone without the participation of role administrators. Our protocol is general and can be realized by any signature scheme. We have described a specific realization with hierarchical certificate-based encryption scheme that gives delegation compact credentials.

REFERENCES

[1] 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.

[2] R. Bhatti, J. Joshi, E. Bertino, and A. Ghafoor. X-GTRBAC admin: a decentralized administration model for enterprise wide access control. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, pages 78–86, 2004.

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

[4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.

[5] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO '01*, volume 2139 of *LNCS*. Springer-Verlag, 2001.

[6] 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.

[7] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. A survey of two signature aggregation techniques. *CryptoBytes*, 6(2), 2003.

[8] 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.

[9] 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.

[10] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine. Authentic third-party data publication. *Journal of Computer Security*, 11(3), 2003.

[11] Y. Ding, P. Horster, and H. Petersen. A new approach for delegation using hierarchical delegation tokens. In *2nd Int. Conference on Computer and Communications Security*, pages 128 – 143. Chapman and Hall, 1996.

[12] FIPS 186-2 Digital signature standard, 2000.

[13] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Yloenen. Simple public key certificate. `http://www.ietf.org/rfc/rfc2693.txt`.

[14] C. Günther. An identity-based key exchange protocol. In *Advances in Cryptology - Eurocrypt '89*, volume 434 of *LNCS*, pages 29–37. Springer-Verlag, 1989.

[15] C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Advances in Cryptology — Eurocrypt '03*, volume 2656 of *LNCS*, pages 272–293, 2003.

[16] M. Girault. Self-certified public keys. In *Advances in Cryptology — Eurocrypt '91*, volume 547 of *LNCS*, pages 490–497. Springer, 1991.

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

[18] M. T. Goodrich, R. Tamassia, N. Triandopoulos, and R. Cohen. Authenticated data structures for graph and geometric searching. In *Proc. RSA Conference—Cryptographers' Track*, volume 2612 of *LNCS*, pages 295–313. Springer, 2003.

[19] C. A. Gunter and T. Jim. Policy-directed certificate retrieval. *Software: Practice and Experience*, 30:1609–1640, September 2000.

[20] T. Jim. SD3: A trust management system with certified evaluation. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 106–115. IEEE Computer Society Press, May 2001.

[21] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.

[22] 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.

[23] 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.

[24] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology – Eurocrypt '04*. Available at `http://eprint.iacr.org/2003/091/`.

[25] N. Nagaratnam and D. Lea. Secure delegation for distributed object environments. In *Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)*, April 1998.

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

[27] B. C. Neuman. Proxy-based authentication and accounting for distributed systems. In *International Conference on Distributed Computing Systems*, pages 283–291, 1993.

[28] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Commun. ACM*, 21:120–126, 1978.

[29] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[30] M. Shehab, E. Bertino, and A. Ghafoor. Secure collaboration in mediator-free environments. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS '05)*, November 2005.

[31] K. R. Sollins. Cascaded authentication. In *Proceedings of 1988 IEEE Symposium on Security and Privacy*, pages 156–163, April 1988.

[32] 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.

[33] V. Varadharajan, P. Allen, and S. Black. An analysis of the proxy problem in distributed systems. In *Proceedings of 1991 IEEE Symposium on Security and Privacy*, pages 255–275, 1991.

[34] 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.

[35] X. Zhang, S. Oh, and R. Sandhu. PBDM: A flexible delegation model in RBAC. In *Proceedings of the ACM Symposium on Access Control Models and Technologies*, pages 149 – 157. ACM Press, 2003.