Role Signatures for Access Control in Open Distributed Systems

Jason Crampton and Hoon Wei Lim

Abstract Implementing access control efficiently and effectively in an open and distributed system is a challenging problem. One reason for this is that users requesting access to remote resources may be unknown to the authorization service that controls access to the requested resources. Hence, it seems inevitable that predefined mappings of principals in one domain to those in the domain containing the resources are needed. In addition, verifying the authenticity of user credentials or attributes can be difficult. In this paper, we propose the concept of *role signatures* to solve these problems by exploiting the hierarchical namespaces that exist in many distributed systems. Our approach makes use of a hierarchical identity-based signature scheme: verification keys are based on generic role identifiers defined within a hierarchical namespace. The verification of a role signature serves to prove that the signer is an authorized user and is assigned to one or more roles. Individual member organizations of a virtual organization are not required to agree on principal mappings beforehand to enforce access control to resources. Moreover, user authentication and credential verification is unified in our approach and can be achieved through a single role signature.

1 Introduction

The most problematic issue for an authorization service in any open distributed computing environment is that access requests may be received from a user that is not known to the authorization service. It is certainly possible to use signed assertions and a public key infrastructure (PKI) to determine that the user has been previously

Jason Crampton
Information Security Group, Royal Holloway, University of London, e-mail: jason.crampton@rhul.ac.uk

Hoon Wei Lim
SAP Research, France, e-mail: hoon.wei.lim@sap.com

Please use the following format when citing this chapter:

Crampton, J. and Lim, H.W., 2008, in IFIP International Federation for Information Processing, Volume 278; *Proceedings of the IFIP TC 11 23rd International Information Security Conference*; Sushil Jajodia, Pierangela Samarati, Stelvio Cimato; (Boston: Springer), pp. 205–219.

authenticated by some security domain D_1, even one not previously known to the security domain D_2 to which the request was directed. It may even be possible to use similar types of assertions to determine that a user has a particular attribute, role r, say, in D_1. However, there still remains the difficult problem of interpreting r in the context of D_2's authorization policy. It seems inevitable that there must be some prior agreement between D_1 and D_2 about what r should mean to D_2. This presupposes that D_2 is aware of the roles defined in D_1's security policy, which also means that D_1 is prepared to reveal role names and their authorization semantics (within D_1) to D_2.

In short, it seems inevitable that pre-defined mappings will need to be specified between principals in one security domain and those in another. It is fair to say, therefore, that authorization is considerably more difficult than authentication in open distributed systems. Indeed, it seems practically impossible to evaluate an access request from a user that is not previously known to the authorization service, unless there exists some *a priori* agreement between the domain containing the authorization service and the requester's domain.

In addition to the problem of principal mapping, we also note that all of the above approaches and existing authorization frameworks that we know of for open distributed computing environments, such as KeyNote [3], SPKI/SDSI [7] and RBTM [12], rely on some form of certificate-based PKI. Essentially these frameworks rely on signed statements or assertions, attesting to the user or the associated public key having a particular attribute. A set of such attributes is used to map the user to principals in the relevant authorization policy. The richer the policy language, the more complex the recovery of these assertions and subsequent computation of an authorization decision becomes. A considerable amount of research effort has been devoted to *credential chain discovery algorithms* in both SPKI/SDSI [5] and RBTM [13], for example. In essence, existing approaches require the processing, particularly verification, of a large number of digitally signed credentials.

In this paper, we consider the problems of *inter-domain principal mapping* and *verification of user credentials* that make authorization so difficult in open distributed environments. We believe the nature of a hierarchical structure within a virtual organization (VO) or a federation offers some opportunities to reduce the impact of the difficulties posed by principal mapping, credential verification and credential chain discovery.[1] Typically, a VO will have a hierarchical structure, enabling member organizations (MOs) and principals within those organizations to be identified uniquely within a hierarchical namespace. Access requests are signed using a hierarchical identity-based signature scheme (HIBS), in which signing keys correspond to role identifiers, hence the terminology *role signatures*. These identifiers are based on the hierarchical namespace in the VO and associated with some generic roles defined by the VO. If the identifiers are correctly formed and the associated signature on the request can be verified, then the user is known to be authorized for those roles in his home organization.

[1] Hereafter, a VO, a term commonly used in large-scale distributed computing systems [8], is assumed to be a collection of geographically dispersed organizations with heterogeneous systems, each of which has individual organizational autonomy.

We now summarize the main contributions of this paper.

- There does not need to be agreement between individual member organizations about how to map principal identifiers. This means that the composition of the VO can be dynamic without compromising the effectiveness of the authorization mechanisms in member organizations. New member organizations can join the VO and need only define some additional rules mapping their local roles to the VO roles.
- User authentication and credential verification is unified and credential verification is rendered trivial. The authorization service is required to verify a single role signature to both confirm that the user is an authenticated member of some other member organization and occupies a particular generic role within that organization.

In the next section, we provide a brief overview of identity-based cryptography, the Gentry-Silverberg HIBS scheme, and a recent extension to this scheme. In Section 3, we present the concept of role signatures and describe the use of a HIBS scheme for constructing and verifying such signatures. We also describe what policies need to be defined by member organizations. In Section 4, we describe a security architecture in which the concept of role signatures can be deployed. We discuss related work in Section 5.

2 Hierarchical Identity-Based Cryptography

The idea of generating public keys based on user names, or some other publicly available information that could uniquely identify a user (such as an email address), was conceived by Shamir more than two decades ago [20]. The corresponding private keys are computed and distributed by a trusted Private Key Generator (PKG). The usual role of a trusted third party (the CA) in a PKI is to attest to the authenticity of public keys. In identity-based cryptography, public keys are derived from public information and their authenticity can be assumed, obviating the requirement for certificates. Hence, the job of the trusted third party (the PKG) is to ensure the correct binding of private keys to identities.

Hierarchical identity-based signatures (HIBS) schemes were developed to reduce the burden of (private) key generation on the PKG. In such schemes, it is assumed that entities can be arranged in a rooted tree and that entities at one level are trusted to issue private keys to entities immediately below them in the tree. More specifically, the root PKG, located at level 0, produces private keys for entities at level 1, who in turn act as PKGs for entities in their respective domains at level 2, *etc*. In the context of this paper, the root PKG is the trusted authority (TA), who issues keys to VOs, who in turn issue keys to MOs, who in turn create role signing keys.

Each node in the tree has an identifer. The identifier of an entity is the concatenation of the node identifiers in the path from the root to the node associated with the entity. Hence, the string $id_1.id_2.\cdots.id_t$ represents an entity at level t whose ancestor

at level 1 has identifier id_1 and whose ancestor at level j has identifier $id_1.\cdots.id_j$. In other words, the tree defines a hierarchical namespace.

We base our work around the Gentry-Silverberg HIBS scheme [10], which works in the following way. The root PKG computes a master secret s_0 and a set of system parameters, and every other entity chooses a secret value. Each non-leaf entity is a PKG and is responsible for computing private keys for each of its children using each child's identifier, the entity's secret information and the system parameters. Each entity may sign messages using the private key generated by its parent. Any other entity may verify the validity of a signature using the signed message, the signer's identifier and the system parameters as inputs.

The purpose of a role signature is to prove membership of a role. As we will see in Section 3.1.2, there may be situations in which it is useful to prove membership of multiple roles with a single signature.

There are other HIBS schemes in the literature, for example [4], that may be used for role signatures. Our proposal is based on Gentry-Silverberg's scheme because it can be extended to support "multi-key signatures" [14], in which several keys are used to generate a single signature, thereby enabling a user to sign a message to prove possession of two or more signing keys.

3 Role Signatures

We assume that there exists a hierarchical structure within an open distributed system, where a trusted authority (TA) is at the top of the hierarchy. Below the TA, we have the VOs who are formed by MOs. We assume that the VO specifies a small number of generic roles that can be used as principals in the authorization policy of each MO. We would argue that this is a much weaker assumption than assuming the existence of mappings between the principals referenced in each of the MOs' access control policies. It is also important to stress at this point that we are using *identifiers* (for VOs, MOs and generic roles), rather than (user) identities in our framework.

We treat the TA as a level 0 entity in a tree, the VOs as level 1 entities, the MOs as level 2 entities, and generic roles as level 3 entities. We then apply the Gentry-Silverberg HIBS scheme to this hierarchical structure. The TA is responsible for issuing signing keys to VOs. We view the issuance of a signing key as analogous to assigning a role to a principal. Hence, if the TA issues a signing key to principal VO_1, this means that VO_1 is a legitimate VO principal (recognized by the TA). This signing key will be derived from the identifier VO_1. Similarly, if the principal VO_1 issues a signing key to Org_1, this means that Org_1 is a legitimate MO principal in VO_1. This signing key will be derived from the identifier $VO_1.Org_1$. Finally, Org_1 may issue a signing key to user u, based on the generic role identifier $VO_1.Org_1.vr$. This is the simplest form of generic role identifier: additional information can be encoded in the identifier to specify the user to which the role is assigned or the lifetime of a key. We discuss these issues in more detail in Section 3.3.

3.1 RBAC Policies for Open Distributed Systems

In our proposal, we assume that a VO comprises a countable set of MOs, $Org_1, Org_2 \ldots$, and that membership of this set may change over time. We also assume that the VO defines a finite set of generic role identifiers $vr_1, \ldots, vr_m$.

Each MO Org_i defines and maintains role-based access control (RBAC) policies. As usual, a policy decision point (PDP_i) for a resource controlled by Org_i uses an access control policy (ACP_i) to decide requests for access to that resource from users authenticated directly by that organization. This is the *internal* ACP.

Each MO extends its ACP so that users in that MO are assigned to zero or more of the generic roles. These role identifiers will be used to map users in one MO to roles in another MO. In addition, the ACP must be extended to specify how members of generic roles in other MOs are mapped to local roles. This is the *external* policy.

3.1.1 Internal ACPs

We use RBAC96 syntax [19] for internal ACPs. We write VR for the set of generic roles identified within a VO. Given a set of internal role identifiers R, we write R^* for $R \cup VR$. Each MO Org_i defines an internal set of roles R_i and defines

- a user-role assignment relation $UA_i \subseteq U_i \times R_i^*$, where U_i is the set of authorized users in Org_i;
- a permission-role assignment relation $PA_i \subseteq P_i \times R_i^*$, where P_i is the set of permissions for resources maintained and protected by Org_i;
- a role hierarchy relation $RH_i \subseteq R_i^* \times R_i^*$, where the graph (R_i^*, RH_i) is directed and acyclic.

We write $(R_i^*, \leqslant)$ for the reflexive transitive closure of RH_i. Henceforth, we drop the subscript i whenever no ambiguity can arise.

Hence, a user $u \in U_i$ may be assigned directly to a generic role vr via the UA_i relation, or assigned implicitly via inheritance in the RH_i relation. This assignment may enable u to access resources in another organization Org_j, depending on the external policy defined by Org_j.

3.1.2 External ACPs

Informally, each member organization needs to decide which other member organizations it trusts, and which generic roles defined by those organizations can be mapped to internal roles. The RT family of languages [12] provides a natural way of stating these external policies. The RT_0 language defines four different types of rules:

- $A.r \leftarrow A'$ is an assertion made by principal A that principal A' is assigned to role r. Such an assertion is equivalent to saying that $(A, r) \in UA_A$, where UA_A

denotes the user-role assignment relation defined by A. In a distributed setting, the assertion may be presented by A' to another principal as a credential signed by A.

- $A.r \leftarrow A'.r'$ is a policy statement made by principal A that any member of role $A'.r'$ is also a member of the role r. In general, this assertion has no direct equivalent in RBAC96, which is concerned with RBAC policies in closed environments. (If, however, $A = A'$, then the statement is analogous to defining a child-parent relationship between r and r' in RBAC96.)
- $A.r \leftarrow A.r'.r''$ is a policy statement made by principal A that says any member of a role $B.r''$, where B is itself a member of role $A.r'$, is a member of role r. Statements of this form allow A to delegate responsibility (to members of $A.r'$) for assigning principals to role r''.
- $A.r \leftarrow A_1.r_1 \cap A_2.r_2$ is a statement made by principal A that says that any member of roles $A_1.r_1$ and $A_2.r_2$ is also a member of role r.

We now show how rules of this form can be used to encode our external policies. The RT rules

$$Org_i.memberOrg \leftarrow VO.memberOrg \tag{1}$$

$$Org_i.vr \leftarrow Org_i.memberOrg.vr \tag{2}$$

assume that (principal) VO defines a role called *memberOrg* and state that

- Org_i defines a role called *memberOrg* and any member of $VO.memberOrg$ is also a member of the local *memberOrg* role;
- any member of a generic role *vr* defined by a member of role *memberOrg* is also a member of the generic role defined by Org_i.

In other words, these rules state that if the VO says that Org_j is a member organization and Org_j says that u is a member of generic role *vr*, then Org_i is prepared to accept u as a member of *vr* as defined and used in ACP_i.

This means that any member of generic role *vr* defined by any MO is also a member of generic role *vr* in Org_i. In particular, in order to be assured that a user is authorized for generic role *vr*, Org_i needs to confirm that there exists a credential from the VO asserting that the MO is a legitimate member of the VO and a credential from the MO asserting that the user is a legitimate member of the role *vr*. In other words, if Org_j signs a credential of the form $Org_j.vr \leftarrow u$ (meaning u is a member of role *vr* defined by Org_j), then Org_i may deduce that u is a member of $Org_j.vr$, *provided* that Org_i can be convinced that Org_j is a genuine MO. The latter check requires the existence of a credential of the form $VO.memberOrg \leftarrow Org_j$ signed by the VO principal. In principle, then, the authenticity of two different credentials needs to be established by Org_i. Moreover, these credentials are issued by different entities. In Section 3.2 will show that these credentials can be encoded in a single role signature.

In fact, Org_i could map generic roles directly to local role r_i using the rules

$$Org_i.memberOrg \leftarrow VO.memberOrg, \tag{3}$$

$$Org_i.r_i \leftarrow Org_i.memberOrg.vr. \tag{4}$$

In general, the external ACP includes rules that map multiple generic roles to local roles and vice versa. For each generic role vr that Org_i chooses to recognize, Org_i defines one or more rules of the form

$$Org_i.r \leftarrow \bigcap_{j=1}^{m} Org_i.memberOrg.vr_j, \tag{5}$$

$$Org_i.memberOrg \leftarrow VO.memberOrg. \tag{6}$$

That is, any user who is a member of each of the generic roles $vr_1, \ldots, vr_m$ defined by any MO (that is recognized by the VO) is a member of role $r \in R_i^*$ in Org_i. It can be seen that this requires checking $m+1$ credentials. In Section 3.2, we show how key aggregation can be used to construct a single role signature, whose verification proves that all $m+1$ credentials are valid.

3.2 Access Request Signing And Verification

As we noted in the preceding section, in conventional RBTM (and other trust management frameworks) it may be necessary for the authorization service to obtain and verify the authenticity of a number of different credentials in order to evaluate an access request. We now demonstrate how hierarchical identity-based signature schemes can be exploited to simplify credential discovery and verification. Essentially, we associate each generic role with a unique identifer within the VO namespace and use this to generate a private key that is used to sign access requests — role signatures. Signature verification is performed using a key that can be derived from the identifier by any principal, thereby enabling that principal (or the PDP acting for that principal) to verify that the user is indeed a member of a particular generic role.

Note first that rules (3) and (4) can be reduced to the rule

$$Org_i.r_i \leftarrow VO.memberOrg.vr.$$

In other words, if a user can provide a credential proving that she is a member of a generic role vr in a member organization of the virtual organization, then she can be mapped directly to role vr in Org_i.

We adopt a push model in which the user supplies authorization credentials as part of an access request. In particular, a user u uses a signing key, such as the one associated with role identifier $VO_1.Org_1.vr$, to sign an access request. If the PDP in Org_2 can verify the signature on the request using the verification key associated with $VO_1.Org_1.vr$, then the PDP in Org_2 can be convinced that VO_1 is a legitimate VO (as far as the TA is concerned), Org_1 is a legitimate MO (as far as the VO is concerned), and u is a legitimate user assigned to role vr (as far as the MO is concerned). The PDP in Org_2 may then use its policy to map the generic role to local

roles, and hence evaluate the access request. Note the definition of a comparatively small number of generic roles and a single signature verification are sufficient to both solve the principal mapping problem and eliminate credential chain discovery.

Moreover, the use of multi-key signatures enables a user to prove authorization for multiple roles in a single signature. Hence external policy rules (5) and (6), which can be reduced to the rule

$$Org_i.r \leftarrow \bigcap_{j=1}^{m} VO.memberOrg.vr_j,$$

can be matched using a single (multi-key) signature. In this case, the user should possess a set of signing keys associated with role identifiers $VO_1.Org_1.vr_1, \ldots, VO_1.Org_1.vr_m$.

3.3 Fine-Grained Identifiers

So far we have looked at how basic role-only identifiers are used to construct the associated signing keys. We now discuss more fine-grained ways of specifying identifiers.

Key Lifetimes It is well known that effective revocation of public-private key pairs is rather difficult to achieve. Within our framework, this is related to user-role revocation. Many practical applications prefer, instead, to use ephemeral keys that have a limited time period for which they are valid. In a grid environment, for example, short-lived keys are used for secure job submissions, to minimize the risk of exposing long-term keys. This is analogous to the relatively short lifetimes given to Kerberos tickets.

Therefore, we envisage that role identifiers will include a lifetime L. A typical identifier would have the form $VO_1.Org_1.vr\|L_1$, the interpretation being that the corresponding signing key would only be valid for time L_1 after its issuance. Note that L_1 can also be set to the validity period of the RBAC session[2] associated with role vr.

User-Role Bindings We remark that the use of signing keys based on role-only identifiers provides user privacy and pseudo-anonymity. However, in some applications, it may be desirable for a resource provider to keep track of the identities of users who accessed its resources for auditing and accountability purposes. This can be achieved by including a local user identifier u in a role identifier, $VO_1.Org_1.vr\|u\|L_1$, for example. The use of user identifiers and lifetimes may be essential for commercial grid applications when billing comes into play.

A role is likely to be shared by more than one user, and hence a signing key, which is based on a role-only identifier, may well be shared by a group of users. The

[2] In an RBAC session, a user activates a number of the roles to which he is assigned, thereby gaining the privileges associated with those roles for that interaction with the system.

inclusion of user identifiers within role identifiers obviates potential issues caused by key sharing. It is worth noting that although user identifiers may be used in role identifiers, principal mappings are still based on roles only.

Generic Role Sets There may also be situations where it is more appropriate for a user presenting all roles to which she is entitled within a single identifier. The user can obtain, from her organization, a signing key associated with all her roles $vr_1, \ldots, vr_m$. Her role identifier now becomes $VO_1.Org_1.(vr_1, \ldots, vr_m) \| u \| L_1$. One advantage of this approach is that the user is relieved of her responsibility in selecting the appropriate signing keys for a particular session. Clearly, on the other hand, the limitation of this method is that it would undermine the principle of least privilege, which may be desirable in some system environments.

3.4 Supporting Multiple Namespaces

It may well be useful to have a number of distinct hierarchical namespaces having different root TAs, with principals having distinct identities in different namespaces. We observe that Lim and Paterson's multi-key signature scheme [14] can be extended naturally to support multiple distinct hierarchical namespaces. The only requirement is that the root TAs of these distinct hierarchies must use the same group generator when computing their respective system parameters. Furthermore, the scheme can take as input private keys which correspond to entities at different levels in a hierarchy.

Consider, for example, an (imaginary) academic institution, the Missouri Institute of Science and Technology (MIST). We may have role identifier $VO_1.Org_1.vr_1$ in a 3-level hierarchical namespace rooted at TA_1 and role identifier $Uni_2.vr_2$ in a 2-level namespace rooted at TA_2, where $Org_1 = Uni_2 = mist$. Informally, the first identifier may be interpreted as: the Missouri Institute of Science and Technology is an accredited member of the virtual organization VO_1 working on data generated by the large hadron collider at CERN, where TA_1 is the EU Grid TA. On the other hand, the second identifier means: the Missouri Institute of Science and Technology is a higher education institution accredited by TA_2, the Accrediting Board for Universities, for example.

Supporting distinct hierarchical namespaces in role signatures is a very desirable feature in the sense that role signatures can now be used to articulate policy rules of the form $Org_i.r \leftarrow VO.memberOrg.vr_j \cap memberUni.vr_k$, where member organizations and universities belong to different hierarchical namespaces.

3.5 Supporting More Complex Namespaces

In the interests of simple exposition, we have assumed so far that the hierarchical namespaces are rather simple, being based on a model in which the level 1 entities are virtual organizations, level 2 entities are member organizations, and level 3 entities are generic roles. This type of structure is characteristic of certain large-scale distributed systems, for example computational grids, but not of open distributed systems in general.

We now discuss how we can build more complex namespaces. The basic ideas are to include the notion of a domain as a generic role and to generalize the binding of users to generic roles in identifiers.

More specifically, identifiers are formed from the concatenation of one or more identifier-role pairs. Hence, the root TA can create level 1 domains. Each level 1 domain is provided with a signing key and material with which to generate signing keys for generic roles, including level 2 domains. In this way, arbitrarily deep hierarchies can be constructed. Identifiers have the form $domain\|D_1\|domain\|D_2\|\ldots\|domain\|D_n\|vr\|u$, where vr is a generic role.

Then MIST might act as a level 0 TA and consider faculties to be level 1 entities. Each faculty is associated with a domain and a signing key. Each department within a faculty is treated as a level 2 domain. Each department is autonomous, in that each has a separate access control policy and is able to define child domains if desired. MIST identifies additional generic roles such as registered student and faculty.

Then a student *alice*, belonging to the computer science (CS) department, within the mathematical sciences (MS) faculty would have an identifier

$$domain\|MS\|domain\|CS\|student\|alice$$

and a signing key corresponding to this identifier. *alice* may send a signed request to the physics department and, for example, be assigned the guest role as a result of the department's external ACP, thereby enabling her to run a computer program using certain data collected and stored by the physics department.

There are other possibilities too. We could for example introduce different types of generic roles for level 1 entities. A computational grid, for example, might include partners from academia, industry and government agencies. In such a situation, it might be appropriate to define generic roles AMO, IMO and GMO, representing academic, industrial and governmental member organizations, respectively. We might then have identifiers $AMO\|MIST\|\ldots$, $IMO\|IBM\|\ldots$ and $GMO\|FBI\|\ldots$.

4 Security Architecture

The concept of role signatures can be easily integrated into a security architecture which makes use of hierarchical identity-based cryptography, for example a password-enabled and certificate-free grid security infrastructure (PECF-GSI) pro-

posed by Crampton *et al.* [6]. PECF-GSI allows users to perform single sign-on based only on passwords and does not require a PKI. Nevertheless, it supports essential grid security services, such as mutual authentication and delegation, using public key cryptographic techniques. The fact that users are authenticated using only passwords significantly increases the user-friendliness of the infrastructure and allows users to join or leave a VO in a flexible way. This is mainly because users do not have to go through the hassle of obtaining a public key certificate when joining a VO. Moreover, this approach alleviates the typical private key distribution issue found in standard identity-based cryptosystems.[3]

We note that although identity-based techniques are certificate-free, an authentic set of the TA system parameters (or all sets of parameters for multiple hierarchies) must be made available to system users. One way to achieve this is by bootstrapping these parameters into the system, as with bootstrapping root CA certificates in existing certificate-based approaches. Alternatively, distribution of the parameters is also possible through the use of a certificate obtained from a conventional CA that certifies the parameters.

Using PECF-GSI, a user authenticates to a domain authentication server through a password-based TLS protocol [1]; hence authentication between the user and the server can take place without relying on a PKI. When performing single sign-on, the user establishes a secure TLS channel with the authentication server based on a shared password. The authentication server, which essentially acts as a MO (within a VO), then creates a proxy (short-lived) credential, comprising a role identifier and its corresponding signing key, and transmits it to the user. As explained in Section 3.3, there are several ways in which a role identifier can be specified. An authenticated copy of the TA system parameters are sent to the user, enabling her to execute the relevant cryptographic algorithms. In addition, the user is sent an up-to-date Identity Revocation List (IRL) so that she can be sure that a resource provider to which she submits her job request is still legitimate, respectively. The user is only required to sign-on once and use the fresh proxy credential generated by the authentication server until the credential expires.

Since our approach is applicable to the multiple hierarchical setting, we envisage that no centralized root TA is required in our architecture. Hence our approach is scalable in the sense that each VO or MO can be associated with a "decentralized" TA that it is willing to trust. Moreover, the multiple TAs setting seems to reflect well trust relationships and management in real world systems.

The use of role signatures seems to suit a decentralized access control model, provided that there exist a hierarchical structure which relates principals involved in access control, in such a way that higher-level authorities can delegate access control decisions to lower-level authorities/principals. This is often the case in many real world systems. For example in the finance sector, the head office of each global financial company can act as the root TA issuing credentials to regional main offices, which in turn, issue credentials to local branch offices. Each customer then is allowed multiple credentials, corresponding to different banks.

[3] Typically, a user of an identity-based cryptosystem is required to obtain her private keys from a TA through an independent secure channel or any out-of-bound mechanisms.

5 Related Work

The idea of generic roles is not entirely new. Li *et al.*, in describing role-based trust management [12], said:

> When an entity A defines $A.R$ to contain $B.R_1$, it needs to understand what B means by the role name R_1. This is the problem of establishing a common vocabulary.

Their solution to the problem was to introduce the concept of *application domain specification documents* (ADSDs), which serve to establish a common vocabulary. In particular, they can be used to define roles that are common to a number of different organizations. In a sense, role signatures provide a way of implementing ADSDs and role-based trust management in which credential verification is performed in a lightweight fashion.

A number of authors have considered the idea of *policy-based cryptography* [2, 21] in recent years. This can be used to implement access control by encrypting resources. A user is only able to read a resource if she has the appropriate encryption key. This approach is rather limited in the type of interactions that can be controlled between the user and the resource.

Bagga and Molva [2] recently introduced a policy-based signature scheme, derived from an identity-based ring signature scheme of [23], which provides the inspiration for our work. However, the policies are expressed as monotonic logical expressions involving complex conjunctions and disjunctions of conditions. Bagga and Molva cite a motivating example in which Bob has an ACP such that Alice is authorized to access some sensitive resource if she is an IEEE member *and* she is an employee of either university X *or* university Y.

The policy is expressed as $\langle$IEEE, Alice:member$\rangle \wedge [\langle X$, Alice:employee$\rangle \vee \langle Y$, Alice:employee$\rangle]$. This way of expressing policies does not seem to be practical, since Bob has to specify each policy for each requester who wants to access the resources. Moreover, it assumes that Bob knows something about every user that will make an access request. In short, while the cryptographic techniques they use to enforce such policies are interesting, it seems unlikely that such policies will be useful in practice.

We note in passing that (presumably the intent of) Bob's ACP could be expressed in the following way:

$$Bob.r \leftarrow IEEE.member \cap Bob.uni.employee$$

where r is a role name mapped to some appropriate permissions. This style of ACP is far more appropriate in an open distributed environment. In this paper, we have shown how role signatures can be used to demonstrate that a user is authorized for a particular generic role within a single contiguous namespace. More importantly, our work examines the fundamental principal mapping problem which underlies the use of policy-based cryptography, rather than designing new cryptographic schemes that support access control and policy enforcement.

Apart from policy-based cryptography, there are also proposals for *attribute-based systems*, for example [11, 17], which are based on Sahai and Waters's attribute-based encryption (ABE) scheme [18]. ABE is closely related to the work of Bagga and Molva [2] and of Smart [21]. In ABE, the recipient's identifier comprises a set of attributes Ψ. A policy enforcer (sender) can specify another set of attributes Ψ', such that the recipient can only decrypt the ciphertext if his identifier Ψ has at least k attributes in common with the set Ψ'. Here k is a parameter set by the system.

As with [2, 21], the proposals of [11, 17] attempt to present constructions of more expressive cryptographic schemes in terms of policy specification and enforcement, without dealing with the underlying principal mapping issue. The central idea of their work is about using a threshold primitive to control access to some data (through encryption), whereby only users who fulfill k-of-n attributes can access the data (through decryption). On the other hand, we study how a hierarchical identity-based signature scheme can be used to provide role signatures that potentially greatly simplify inter-domain principal mappings and credential verification.

Perhaps the work that is most similar in spirit to ours, is that of Tamassia *et al.* on *role-based cascaded delegation* (RBCD) [22]. RBCD combines the advantages of RBTM with those of cascaded delegation [15]. Their proposal uses a hierarchical certificate-based encryption scheme [9] to simplify credential accumulation and verification. The basic idea is to encode the chain of credentials into a single signed delegation credential.

RBCD is only described using an extended example, making it difficult to analyze the approach formally. Each component of a delegation credential has the form (iss, r, p), where *iss* is the issuer of the credential, p is the subject of the credential who is authorized for role r. The delegation credential in the example has the form

$$(H, H.guest, M.professor)$$
$$(M, M.professor, Bob)$$
$$(Bob, H.guest, L.assistant)$$
$$(L, L.assistant, Alice)$$

meaning that

- hospital H says that any member of the professor role at the medical school M is also a member of the role $H.guest$;
- M says that *Bob* is a member of the professor role;
- Bob says that any member of the lab assistant role at lab L is a member of role $H.guest$;
- L says that *Alice* is a member of the lab assistant role.

It is suggested by the authors that this implies that H, on receipt of this delegation credential from *Alice*, can verify that she is indeed a member of the $H.guest$ role.

However, H needs to know about the professor role at M, and M is required to know that the professor role is important to H. RBCD also assumes that credentials of the form $(Bob, H.guest, L.assistant)$ are regarded as trustworthy by the hospital.

It also assumes that *Bob* is aware that he can issue credentials of this form, and knows to include the $(M, M.professor, Bob)$ credential in the delegation credential. In short, the problem of principal mapping is not addressed by RBCD.

6 Conclusions

We have proposed the use of role signatures for access control in open distributed systems. Our work is built on three assumptions:

- it is reasonable to define a comparatively small number of generic roles that will be recognized throughout a virtual organization;
- the structure of a virtual organization defines a hierarchical namespace;
- members of the virtual organization are trusted to assign their respective users to generic roles.

We have shown how an hierarchical identity-based signature scheme can be adapted to provide role signatures, where the corresponding verification keys are associated with generic roles.

Key management in our proposal is simple as role signatures can be used to both authenticate users and make access control decisions. Hence, we avoid the use of complex credential or certificate chain discovery mechanisms. Moreover, our approach allows signing with multiple keys. These keys, which are associated with multiple roles, can correspond to nodes at arbitrary positions within the same hierarchy or multiple hierarchies.

To conclude, our work provides nice balance between expressiveness of policy and ease of credential verification as compared to existing role-based access control and trust management frameworks.

Acknowledgements The second author was at Royal Holloway, University of London when he performed this work. He was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through Grant EP/D051878/1.

We would like to thank the anonymous referees for their helpful remarks, which have led to substantial improvements in the presentation and exposition of our work.

References

1. M. Abdalla, E. Bresson, O. Chevassut, B. Möller, and D. Pointcheval. Provably secure password-based authentication in TLS. In *Proceedings of the 1st ACM Symposium on Information, Computer and Communications Security (ASIACCS 2006)*, pages 35–45. ACM Press, March 2006.
2. W. Bagga and R. Molva. Policy-based cryptography and applications. In *Proceedings of the 9th International Conference on Financial Cryptography and Data Security (FC 2005)*, pages 72–87. Springer-Verlag LNCS 3570, February 2005.
3. M. Blaze, J. Feigenbaum, J. Ioannidis, and A.D. Keromytis. The KeyNote trust-management system version 2. *The Internet Engineering Task Force (IETF)*, RFC 2704, September 1999.

4. D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology – Proceedings of EUROCRYPT 2005*, pages 440–456. Springer-Verlag LNCS 3494, May 2005.
5. D. Clarke, J. Elien, C. Ellison, M. Fredette, A. Morcos, and R. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, January 2001.
6. J. Crampton, H.W. Lim, K.G. Paterson, and G. Price. A certificate-free grid security infrastructure supporting password-based user authentication. In *Proceedings of the 6th Annual PKI R&D Workshop 2007*. NIST Interagency Report 7427, September 2007.
7. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. *The Internet Engineering Task Force (IETF)*, RFC 2693, September 1999.
8. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.
9. C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Advances in Cryptology – Proceedings of EUROCRYPT 2003*, pages 272–293. Springer-Verlag LNCS 2656, May 2003.
10. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology – Proceedings of ASIACRYPT 2002*, pages 548–566. Springer-Verlag LNCS 2501, December 2002.
11. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Computer and Communications Security Conference (CCS 2006)*, pages 89–98. ACM Press, October 2006.
12. N. Li, J.C. Mitchell, and W.H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, May 2002.
13. 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.
14. H.W. Lim and K.G. Paterson. Multi-key hierarchical identity-based signatures. In *Proceedings of the 11th IMA International Conference on Cryptography and Coding (IMA 2007)*, pages 384–402. Springer-Verlag LNCS 4887, December 2007.
15. N. Nagaratnam and D. Lea. Secure delegation for distributed object environments. In *Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems*, pages 101–116, April 1998.
16. K.G. Paterson. Cryptography from pairings. In I.F. Blake, G. Seroussi, and N.P. Smart, editors, *Advances in Elliptic Curve Cryptography*, pages 215–251, Cambridge, 2005. Cambridge University Press, LMS 317.
17. M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *Proceedings of the 13th ACM Computer and Communications Security Conference (CCS 2006)*, pages 99–112. ACM Press, October 2006.
18. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology – Proceedings of EUROCRYPT 2005*, pages 457–473. Springer-Verlag LNCS 3494, May 2005.
19. R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.
20. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – Proceedings of CRYPTO '84*, pages 47–53. Springer-Verlag LNCS 196, August 1985.
21. N.P. Smart. Access control using pairing based cryptography. In *Proceedings of the RSA Conference: Topics in Cryptology – the Cryptographers' Track (CT-RSA 2003)*, pages 111–121. Springer-Verlag LNCS 2612, April 2003.
22. R. Tamassia, D. Yao, and W.H. Winsborough. Role-based cascaded delegation. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*, pages 146–155. ACM Press, June 2004.
23. F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In *Advances in Cryptology – Proceedings of ASIACRYPT 2002*, pages 533–547. Springer-Verlag LNCS 2501, December 2002.