# Role Signatures for Access Control in Grid Computing

Jason Crampton and Hoon Wei Lim

Technical Report
RHUL–MA–2007–2
10 May 2007


Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

# Role Signatures for Access Control in Grid Computing

Jason Crampton Hoon Wei Lim

Information Security Group, Royal Holloway, University of London

May 10, 2007

**Abstract**

Implementing access control efficiently and effectively in an open and distributed grid environment 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 pre-defined mappings of principals in one domain to those in the domain containing the resources are needed. A second problem in such environments is that 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 structure of a virtual organization within a grid environment. Our approach makes use of a hierarchical identity-based signature scheme whereby verification keys are defined by generic role identifiers defined within a hierarchical namespace. We show that 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

Grid computing [13, 14] is a form of large-scale highly distributed computing, offering users access to vast repositories of electronic resources. These resources and users, from geographically dispersed organizations, are connected through a uniform grid infrastructure. These organizations are grouped in virtual organizations (VOs), which, in turn, enable collaboration between users and inter-operability of resources of different platforms, typically to solve resource-intensive and scientifically complex problems.

Access control is a central security issue within grid environments. A VO potentially spans multiple administrative domains; hence the VO must establish agreements with the resource providers (or owners) in terms of sharing policies *a priori*. This sharing is, necessarily, highly controlled, with resource providers and users defining clearly and carefully what is shared, who is allowed to share, and the conditions under which sharing occurs. The shared resources may be basic computational resources, such as compute cycles and storage; sophisticated scientific instruments, such as sensors and telescopes; data elements, such as files and information in databases; or services provided by specialized application programs. Note that while all the physical organizations within a VO may, in principle, agree to allow members of the VO (some) access to their resources, they still retain ultimate control over the policies that govern access to their respective resources [14].

In a closed distributed computing environment, there is a centralized repository for principal names, and all authorization services trust that repository to attest to the identity of, or authenticity of keys associated with, those principals. Kerberos is the *de facto* standard for supporting authentication and authorization in such environments.

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 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 pre-supposes that $D_2$ is aware of the roles defined in $D_1$'s security policy.

In grid applications, the most basic means of controlling access to resources is through the use of gripmap files [8, 34], mapping user identities to permissions. This is analogous to the conventional approach of using access control lists. One major limitation of this approach is that it essentially requires each member organization (MO) within a VO to maintain a list of all users who are authorized to access its resources. Clearly, this approach incurs substantial administrative overheads for each MO. This prompted the development of more scalable access control mechanisms, including Community Authorization Service (CAS) [24], Virtual Organization Membership Service (VOMS) [2], and Privilege and Role Management Infrastructure Standards Validation (PERMIS) [9].

The issues of identity-permission mapping and pre-establishment of user accounts can be alleviated by employing a centralized CAS server within a VO. The CAS server maintains a mapping between users in the VO and the permissions for which they are authorized. A user obtains a signed assertion from the CAS server before making an access request, indicating the permissions for which the user is authorized. Each MO maintains a CAS server account and runs an external job request using the CAS server account. In other words, the local CAS server account "impersonates" the remote user, and uses the permissions encoded in the CAS server's assertion for access control. Obviously, it is the responsibility of the CAS server to identify the correct permissions that should be used to run the job. This means that the CAS server has to be aware of all users in the VO and the resources for which they are authorized throughout the VO. Again, this is likely to incur substantial overheads.

VOMS is architecturally similar to CAS in that the VOMS server also issues signed assertions to users on request, who then present these assertions to remote resource providers. The primary difference between CAS and VOMS is that the VOMS server issues assertions about the user's attributes, such as group or role membership. Nevertheless, we still require the existence of mappings between different groups and roles within the VO.

While both CAS and VOMS are examples of a *push* model (in which the user obtains credentials and "pushes" them to the resource provider), PERMIS adopts a *pull* model. It is a role-based access control infrastructure customized for managing role-based access control to resources within grid environments. It makes use of attribute certificates, issued by a trusted Attribute Authority (AA), to bind user identities to VO roles and policies. These attribute certificates are stored in a public directory. Once a user has authenticated himself to a resource provider, the resource provider passes the user's request to a policy engine, which then retrieves attribute certificates associated with the user's identity and makes an authorization decision. PERMIS relies on privilege management infrastructure (PMI), which seems to result in an architecture that is more complicated than either CAS and VOMS.

In short, it seems inevitable that pre-defined mappings will need to be defined between principals in one security domain to 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 user'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 [4], SPKI/SDSI [12], RBTM [20] and Akenti [32], 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 proving compliance of the user with the policy. A considerable amount of research effort has been devoted to *credential chain discovery algorithms* in both SPKI/SDSI [10] and RBTM [21], for example. In essence, existing approaches may require management, particularly verification, of a large number of credentials.

In this paper, then, we consider the problems of *interdomain principal mapping* and *authentication of user credentials* that make authorization so difficult in open distributed environments. We believe the nature of grid computing and VOs offers some opportunities to reduce the impact of the difficulties posed by principal mapping, credential verification and credential chain discovery. Typically, a VO will have a hierarchical structure, enabling MOs and principals within those organizations to be identified uniquely within a hierarchical namespace. 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 also make the common assumptions that the VO is a principal who is trusted to enrol new MOs into the VO, and that MOs are trusted to assign their own users to generic roles.

We then employ a hierarchical identity-based signature scheme (HIBS), using the identifiers in the hierarchical namespace defined by the VO to define signing-verifying key pairs for generic roles. Access requests are signed using a role's signing key, hence the moniker *role signatures*, and can be verified by any authorization service using the verification key, which is defined by the role identifier. If the identifier is correctly formed and the signature on the request can be verified, then the user is known to be authorized for that role in his home organization. The verifier then uses its local policy to map the generic role to local roles and thus evaluate the request.

The main contributions of our approach are:

- 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 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 describe what policies need to be defined by member organizations. In Section 3, we describe the Gentry–Silverberg HIBS scheme and use this to construct role signatures for generic roles. In Section 4, we describe our security architecture, and how users are authenticated, keys distributed and access requests evaluated. We then discuss related work in Section 5. We conclude with some ideas for future work.

# 2 Access control policies

Most current computational grids are used by scientists and researchers within academic communities. Within Europe, for example, we have the UK National e-Science community.[1] A central, national-scale certification authority (CA) issues public-key certificates to VOs, which in turn issue certificates to MOs in their respective VOs. A user $u$ in member organization $Org_1$ might be authenticated by $Org_1$ and issued with a short-term certificate signed using $Org_1$'s signing key; $u$ can present this certificate to another MO $Org_2$ to prove that he is an authenticated user within the VO. These kinds of interactions between users, their home MO and the resource provider MO are central to existing approaches to authorization in grid computing.

[1] See the UK National e-Science Centre `http://www.nesc.ac.uk`

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. Each MO defines and maintains an access control policy (ACP). As usual, a policy decision point (PDP) for a resource controlled by a particular organization uses the ACP to decide requests for access to that resource from users authenticated directly by that organization.

We also assume that the VO defines a finite set of generic role identifiers $vr_1, \ldots, vr_m$. 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. We now examine the options for policy specification. We write $VR$ for the set of generic roles identified within a VO. Given a set of local role identifiers $R$, we write $R^*$ for $R \cup VR$.

Our access control policies make use of concepts from RBAC96 [28], RBTM [20] and OASIS [35]. Each MO $Org_i$ defines a "local" 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 $PA_i \subseteq P_k \times R_i^*$, 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. Moreover, a user $v \in U_j$ assigned to generic role $vr$ is mapped directly onto $vr$ in $Org_i$ (and any other junior roles implied by the $RH_i$ relation).

This is equivalent to the following $RT$ rules [20].

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

which can be reduced to the rule

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

This means that any member of generic role $vr$ defined by any MO is 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 $Org_i$. In Section 3 will show that these credentials can be encoded in a single role signature.

Some organizations may not want such a tightly coupled interaction between local roles and generic roles. In this case, no authorizations are assigned directly to generic roles. Instead, mappings between generic roles and local roles are defined using something analogous to $RT$-style rules or OASIS-style role triggers. Hence an organization may choose to omit generic roles from either the role hierarchy or the user-role assignment relation.

We extend the basic ACP described above to include rules that map 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 $r \leftarrow vr_1 \cap \cdots \cap vr_m$, where $r \in R_i$ and $m \geqslant 1$. This is shorthand for the following $RT$-style rules:

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

  which can be reduced to the rule

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

  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 $Org_i$. It can be seen that this requires checking $m+1$ credentials. In Section 3, we show how key aggregation can be used to construct a single role signature, whose verification proves that all $m+1$ credentials are valid.

- For each generic role $vr$ that $Org_i$ chooses to assign to its local users, $Org_i$ defines one or more rules of the form $vr \leftarrow r$. This is shorthand for $Org_i.vr \leftarrow Org_i.r$, and means that any user in $Org_i$ that is a member of role $r$ is also a member of generic role $vr$.

  In conventional RBTM, if user $u \in U_i$ is indeed a member of role $r \in R_i$, then $Org_i$ may issue a signed credential asserting $Org_i.vr \leftarrow u$. Let us suppose that organization $Org_j$ defines a rule of the form $r' \leftarrow vr$. Then organization $Org_j$ can take the statement $Org_i.vr \leftarrow u$ and a statement of the form $VO.memberOrg \leftarrow Org_i$, and deduce that $u$ is a member of the role $r'$.

As we have seen in the examples, in conventional RBTM (and other trust management frameworks) it is common for the authorization service to verify the authenticity of a number of different credentials in order to evaluate an access request. In Section 3, we demonstrate how the structure of many virtual organizations and hierarchical identity-based signature schemes can be exploited to simplify the credential 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.

# 3 Role signatures

## 3.1 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 [29]. 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. Hence, the job of the trusted third party (the PKG) is to ensure the correct binding of private keys to identities.

The main motivation for this approach is to eliminate the need for certificates and the problems associated with them [18]. Since a user's public key is based on some publicly available information that uniquely represents the user, an identity-based cryptosystem eliminates public key directory maintenance and certificate management. At the time, however, Shamir was only able to develop an identity-based signature scheme, based on the RSA primitive.

Only in the early 2000s did the emergence of cryptographic schemes based on pairings on elliptic curves result in the construction of a feasible and secure identity-based encryption (IBE) scheme. This area began with the novel work of Sakai *et al.* [27] on pairing-based key agreement protocols and signature schemes, and subsequent work on the three-party key agreement protocol by Joux [19]. Boneh and Franklin [6] then presented the first practical and secure IBE scheme based on the Weil pairings. These three key contributions have stimulated the development of a wide range of pairing-based cryptographic schemes and protocols. The basic concept of pairings is described in Appendix A; a more detailed survey on cryptography from pairings can be found in [23].

### 3.2 The Gentry–Silverberg hierarchical identity-based signature scheme

Shortly after Boneh and Franklin proposed their identity-based encryption scheme [6], Gentry and Silverberg developed a hierarchical identity-based encryption (HIBE) scheme and a hierarchical identity-based signature (HIBS) scheme [16]. These schemes are provably secure, practical, and fully scalable, with total collusion resistance, regardless of the number of levels in the hierarchy.

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 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 $id_1.\cdots.id_j$.

We now informally describe the Gentry–Silverberg HIBS scheme. A more formal description of the scheme is given in Appendix B.

ROOT SETUP**:** This algorithm is run by the root PKG. It takes as input a security parameter and outputs a master secret $s_0$ and a set of system parameters.

LOWER-LEVEL SETUP**:** A lower-level entity (lower-level PKG or user) at level $t$ picks a random secret $s_t$, which will be used for extracting private keys or signing messages.

EXTRACT**:** This algorithm is used by a PKG at level $t-1$ to generate a private key $S_t$ for an entity at level $t$. The algorithm also generates a set of "Q-values", $Q_1, \ldots, Q_{t-1}$.[2] The algorithm takes the entity's identifier $id_1.id_2.\cdots.id_t$, the parent's private key (and other secret information), and the system parameters as inputs.

SIGN**:** This algorithm is used by an entity at level $t$ for signing and takes a private key $S_t$, a message $M$ and the system parameters as inputs. The output signature is of the form $\langle \sigma, Q_1, \ldots, Q_t \rangle$.

VERIFY**:** This algorithm is used to verify the validity of the signature. It takes a signature $\langle \sigma, Q_1, \ldots, Q_t \rangle$, a message $M$, the identifier $id_1.id_2.\cdots.id_t$ of the signer and the system parameters as inputs.

It is worth noting that there are other HIBS schemes in the literature, for example [5], that may be used for role signatures. Our proposal is based on Gentry–Silverberg's scheme because it can be extended to support aggregation of signing keys in a natural way.

[2] Q-values are needed for the soundness and correctness of the Gentry–Silverberg HIBS scheme.

### 3.3 Key aggregation

As explained in Section 2, MOs may specify rules of the form $r \leftarrow vr_1 \cap \cdots \cap vr_m$. In order to handle rules of this type, we observe that the Gentry–Silverberg HIBS scheme can be straightforwardly extended to include the idea of *aggregate keys*. In other words, a user may sign a message to prove possession of two or more signing keys, provided those keys are associated with entities having the same parent in the hierarchy.

We assume that there exist a set of signing keys $S_{t,1}, \ldots, S_{t,m}$, where $S_{t,i}$ is the signing key for an entity with identifier $id_1.\cdots.id_{t-1}.id_{t,i}$, $i = 1, \ldots, m$. We introduce two new algorithms.

AggSign**:** This algorithm takes as input a set of signing keys $S_{t,1}, \ldots, S_{t,m}$, a message $M$ and the system parameters, and outputs a signature of the form $\langle \sigma, Q_1, \ldots, Q_{t-1}, Q_{agg} \rangle$.

AggVerify**:** Given a signature $\langle \sigma, Q_1, \ldots, Q_{t-1}, Q_{agg} \rangle$, a message $M$ and identifiers associated with the signing keys, this algorithm verifies whether or not the signature is valid.

Further details of these algorithms can be found in Appendix C. A formal security analysis of the Gentry–Silverberg HIBS scheme which supports aggregation of signing keys is beyond the scope of this paper.

### 3.4 HIBS, VOs and roles

The existing grid infrastructure in the UK has a root CA, which is the root of trust, and is responsible for attesting to the authenticity of public keys associated with VOs working on grid projects. Below the root CA we have the VOs who are responsible for issuing public keys to organizations such as universities and research laboratories that wish to participate in a particular VO. We treat the root CA 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. It is 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 then apply the Gentry–Silverberg HIBS scheme to this hierarchical structure, replacing the CA with a root PKG or trusted authority (TA). 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 Why is it more useful .

A user may use this key 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 aggregate signing keys enables MOs to articulate policy rules of the form $r \leftarrow vr_1 \cap \cdots \cap vr_m$. In this case, the user should possess a set of signing keys associated with role identifiers $VO_1.Org_1.vr_1, \ldots, VO_1.Org_1.vr_m$.

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.

#### 3.4.1 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 minimise the risk of exposing long-term keys. This is analogous to the relatively short lifetimes given to Kerberos tickets. Job-signing keys within a grid environment are not usually valid for longer than 12 hours.

Therefore, we envisage that role identifiers will include a lifetime $L$. A typical identifier would have the form $VO_1.Org_1.vr\|L_1$. That means that the corresponding signing key would only be valid for time $L_1$ after its issuance and key revocation is not a major concern. Note that $L_1$ can also be set to the validity period of the RBAC session[3] associated with role $vr$.

### 3.4.2 User identifiers

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 $Uid$ in a role identifier, $VO_1.Org_1.vr\|Uid_1\|L_1$, 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 inclusion of user identifiers within role identifiers obviates potential issues caused by key sharing. It is worth noting that although user identifiers are used in role identifiers, principal mappings are still based on roles only.

### 3.4.3 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 $VO_1.Org_1.(vr_1, \ldots, vr_m)\|Uid_1\|L_1$. 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.

We have described how additional fields, such as lifetime and user identifier, can be optionally appended to role identifiers to construct more meaningful signing keys with more specific and fine-grained attributes. In the next section we show how the concept of role signatures can be easily integrated into an existing security architecture for grid computing.

# 4 Security architecture

Our access control framework is built on a grid authentication framework recently proposed by Crampton *et al.* [11]. The password-enabled and certificate-free grid security infrastructure (PECF-GSI) is based on hierarchical identity-based cryptography, 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.

Figure 1 illustrates a fragment of the hierarchical namespace. We assume that every member organization in the VO is aware of the generic role identifiers $vr_1$ and $vr_2$ defined by the VO.

[3] 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.

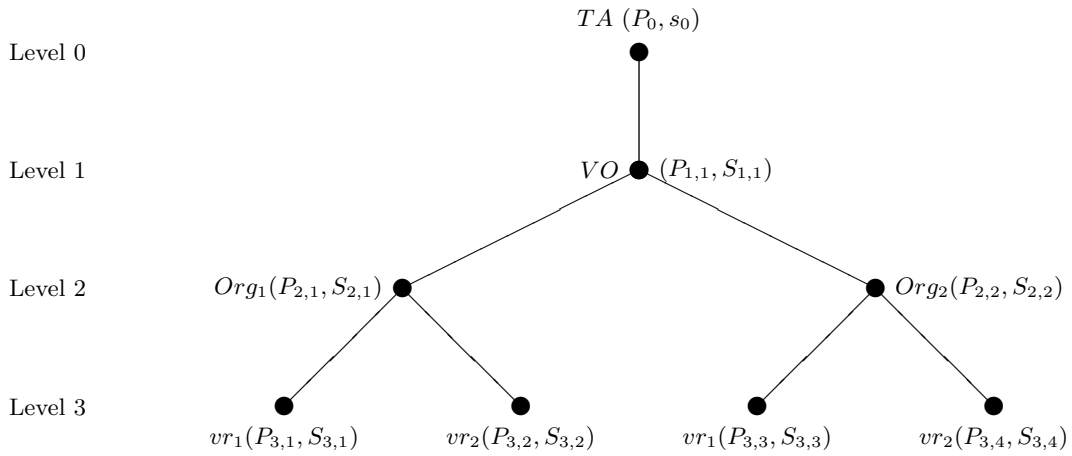


Figure 1: Hierarchy of principals and roles in the VO

## 4.1 System initialization

The TA runs the Root Setup of the Gentry–Silverberg HIBS schemes to produce a master secret $s_0$ and the system parameters.

An authentic set of the TA parameters must be made available to authentication and hosting servers within the grid. One way to achieve this is by bootstrapping these parameters into the grid system. Alternatively, distribution of the parameters is also possible through the use of a certificate obtained from a conventional CA that certifies the parameters.

## 4.2 Key issuance

Once the system parameters have been established, the TA (at level 0) can issue private keys to VOs, using its master secret $s_0$ and the Gentry–Silverberg Extract algorithm. For example, the long-term private key of the VO (with identifier $VO$) is $S_{1,1} = s_0 P_{1,1}$, where $P_{1,1} = H_1(VO)$ is the matching public key. Note that $s_0 P_{1,1}$ refers to a point multiplication between the master secret $s_0$ and the public key $P_{1,1}$; while $H_1$, a component of the system parameters, denotes a mapping of a string onto a group element (see Appendices A and B for further details). The associated Q-value, $Q_{1,1}$ is $s_{1,1} P_0$, where $s_{1,1}$ is a secret value randomly chosen by VO using the Lower-level Setup algorithm, and $P_0$ is available from the system parameters. Note that we use $i$ and $j$ as subscripts in $P_{i,j}$, $S_{i,j}$ and $Q_{i,j}$ to represent hierarchy depth index and row index (within the same level of hierarchy), respectively.

Subsequently, the VO at level 1 issues private keys to principals at level 2, e.g. $Org_i$, which in turn issuing private keys corresponding to some generic roles, e.g. $vr_i$, to lower level principals by performing the same algorithms. The key sets that each principal derives are summarized in Table 1.

## 4.3 User authentication and single sign-on

In 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.

Table 1: Key sets of various principals in the VO hierarchy

| Identifier | Public Key | Private Key | Q-value |
|---|---|---|---|
| $VO$ | $P_{1,1} = H_1(VO)$ | $S_{1,1} = s_0 P_{1,1}$ | $Q_{1,1} = s_{1,1} P_0$ |
| $Org_i$ | $P_{2,i} = H_1(VO.Org_i)$ | $S_{2,i} = S_{1,1} + s_1 P_{2,i}$ | $Q_{2,i} = s_{2,i} P_0$ |
| $vr_i$ | $P_{3,i} = H_1(VO.Org_i.vr_i)$ | $S_{3,i} = S_{2,i} + s_2 P_{3,i}$ | $Q_{3,i} = s_{3,i} P_0$ |

When performing single sign-on, the user establishes a secure TLS channel with the authentication server based on a shared password. The authentication server then creates a proxy (short-lived) credential and transmits it to the user. An authenticated copy of the TA system parameters are sent to the user, enabling her to execute the SIGN algorithm. 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.

Each user in an MO is authorized for a number of roles in the local RBAC policy maintained by the MO. This may include a number of generic roles defined by the VO, either directly using the *UA* relation, or indirectly using RT-style rules.

Suppose that user $A$ is authorized for two generic roles, $vr_1$ and $vr_2$, by $Org_1$'s ACP. Once $A$ and $Org_1$ have successfully mutually authenticated, $Org_1$ issues proxy credentials for roles $vr_1$ and $vr_2$ (that is the key pairs $(P_{3,1}, S_{3,1})$ and $(P_{3,2}, S_{3,2})$) to $A$ through a secure TLS channel. As explained in Section 3.4, there are several ways in which $A$'s role identifiers can be specified. In our following example scenario, we assume that user $A$ is given two signing keys associated with identifiers $VO.Org_1.vr_1 \| A \| L_A$ and $VO.Org_1.vr_2 \| A \| L_A$, respectively. Here, $L_A$ denotes the lifetime of the key issued to $A$. We assume there is a common format for key lifetimes within the VO and system clocks are loosely synchronized.

## 4.4 Job submission

Our job submission model is based on the Globus Toolkit's Grid Resource Allocation and Management (GRAM) [34]. GRAM is designed to provide a single common protocol and API for requesting and using remote system resources, by providing a uniform and flexible interface to local job scheduling systems. To invoke a job using GRAM, a user creates a job request, describes the job to be run, and sends it to a GRAM adapter running on a resource. The GRAM adapter maps the request onto an appropriate request to a local scheduler. The local scheduler has an interface called Managed Job Factory (MJF) which then creates one or more managed job instances to execute the submitted job.

### 4.4.1 Access request

Before submitting a job to the computational grid within the VO, $A$ first selects the appropriate credential or signing key, depending on the job and the types of resources that $A$ needs to access.

Let's assume that $A$ submits a job using the key $S_{3,1}$ corresponding to the identifier $VO.Org_1.vr_1 \| A \| L_A$, which requires access to some resource hosted by $Org_2$. She can then use the credential to produce a role signature of the form $\langle \sigma, Q_{1,1}, Q_{2,1}, Q_{3,1} \rangle$ over the job request, *Req*, using the Gentry–Silverberg SIGN algorithm. $A$ submits *Req*, $VO.Org_1.vr_1 \| A \| L_A$ and $\langle \sigma, Q_{1,1}, Q_{2,1}, Q_{3,1} \rangle$ via GRAM to $Org_2$. Naturally, $A$ would use the AGGSIGN algorithm if she wished to sign a request indicating authorization for both $vr_1$ and $vr_2$.

### 4.4.2 Access decision

Upon receiving the signed job request and the relevant information from $A$, (the PDP for) $Org_2$ performs the following steps:

1. verify the role signature (using VERIFY or AGGVERIFY as appropriate);
2. if the role signature can be successfully verified, extract the appropriate generic role(s) from the identifier;
3. map the generic role(s) to local roles using the ACP defined by $Org_2$ for those generic role(s);
4. access is granted if the local role is authorized for *Req*.

Since $Org_2$ trusts $Org_1$ in issuing $S_{3,1}$ to the correct principal, $Org_2$ can be convinced of $A$ being an authorized user with the appropriate role, if the signed request is valid.

# 5 Related work

## 5.1 Credential verification in RBTM

Assuming that a PDP trusts a TA to only issue keys to valid VOs, any VO that the TA trusts to only issue keys to valid MOs, and any MO that a trusted VO trusts to only issue role keys to valid users, the verification of a role signature is equivalent to verifying a form of linked containment rule in the *RT* language. Specifically, consider the following rules:

$$
\begin{aligned}
&Org_2.r \leftarrow Org_2.MO.vr\\
&Org_2.MO \leftarrow Org_2.VO.MO\\
&Org_2.VO \leftarrow TA.VO\\
&TA.VO \leftarrow VO_1\\
&VO_1.MO \leftarrow Org_1\\
&Org_1.vr \leftarrow u
\end{aligned}
$$

These $RT_0$ rules would enable $Org_2$ to conclude that $u$ was authorized to be a member of role $Org_2.r$, *provided* the authenticity of the credentials $TA.VO \leftarrow VO_1$, $VO_1.MO \leftarrow Org_1$ and $Org_1.vr \leftarrow u$ could be verified. We claim that role signatures makes policy specification easier (compare the rules that $Org_2$ needs to define above, with those required in Section 2) and credential verification easier. Of course, the comparison is slightly unfair, because RBTM is for arbitrary open distributed environments, whereas the grid environments we are discussing have a natural hierarchical disposition.

## 5.2 Policy-based and attribute-based cryptography

A number of authors have considered the idea of *policy-based cryptography* [3, 30] 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 [3] recently introduced a policy-based signature scheme, derived from an identity-based ring signature scheme of [36], 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 on *attribute-based systems* [17, 25], which are based on Sahai and Waters's attribute-based encryption (ABE) scheme [26]. ABE is closely related to the work of Bagga and Molva [3] and of Smart [30]. In ABE, the recipient's identifier comprises a set of attributes $\Psi$. A policy enforcer (sender) can specify another set of attributes $\Psi'$, such that the recipient can only decrypt the ciphertext if his identifier $\Psi$ has at least $k$ attributes in common with the set $\Psi'$. Here $k$ is a parameter set by the system.

As with [3, 30], the proposals of [17, 25] 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 thresholding 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.

### 5.3 Role-based cascaded delegation

Perhaps the work that is most similar in spirit to ours, is that of Tamassia *et al.* on *role-based cascaded delegation* (RBCD) [31]. RBCD combines the advantages of RBTM with those of cascaded delegation [22]. Their proposal uses a hierarchical certificate-based encryption scheme [15], itself based on an aggregate signature scheme [7], to simplify credential accumulation and verification. The basic idea is to encode the chain of credentials into a single signed delegation credential.

RBCD is 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 used by Tamassia *et al.* has the form

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

The scenario is 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 Future work

## 6.1 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 may have *ethz.edu* and *ethz.cern.e-science*, for example. Informally, the first identifier means that Swiss Federal Institute of Technology, Zurich is an accredited higher education institution, while the second means that Swiss Federal Institute of Technology is an accredited member of the virtual organization working on data generated by the large hadron collider at CERN. It will be a challenging problem to devise a role signature scheme that enables a user to sign a request using keys from multiple namespaces.

## 6.2 More meaningful identifiers

Currently, we use an identifier that is analogous to a fully qualified domain name. However, consider a scenario in which there may be more than one type of MO in a grid: we might have industrial and academic partners, for example; academic partners might be treated as members of an $AcMO$ role and industrial partners as members of an $IndMO$ role. In this scenario, the fully qualified domain names $VO_1.ibm.vr_1$ and $VO_1.mit.vr_1$ fails to distinguish between the roles played by a particular entity. At the level of generic roles, it might be useful to explicitly identify the user associated with a generic role, particularly for auditing and accountability.

To address these issues, we think it would be interesting to investigate the use of X.500-style distinguished names. Specifically, an entity identifier is based on the fully qualified domain name, but it is expanded to a sequence of role-identifier pairs. Hence, for example, we might have an identifiers of the form $vr_1 = Alice, AcMO = mit, VO = VO_1$ and $vr_1 = Bob, IndMO = ibm, VO = VO_1$. This type of approach would also be useful when considering the generation of proxy credentials, something widely used in grid computing.

## 6.3 Delegation and proxy credentials

Delegation of credentials in the context of grid computing means that a user issues a proxy credential to an entity or a proxy acting on the user's behalf [33]. Using the approach to identifiers suggested in the previous section, each user has their own namespace and can issue new keys based on a role key. In the example above, given an identifier for $Alice$, $vr_1 = Alice, MO = rhul, VO = VO_1$, we can construct a proxy identifier of the form $vr_2 = AliceProxy, vr_1 = Alice, MO = rhul, VO = VO_1$. The assignment of $vr_2 \neq vr_1$ to the proxy could be used by $Alice$ to limit the privileges of her proxy.

# 7 Conclusions

We have proposed the use of role signatures for access control in grid environments. Our work build 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. It seems reasonable to assume that this approach will be applicable to any form of dynamic coalition with some hierarchical structure, not just virtual organizations in grid computing.

Our approach provides greater flexibility than conventional grid access control mechanisms, in the sense that generic roles defined by a member organization can be mapped to local roles of another member organization without relying on additional dedicated authorization servers. Key management in our proposal is simple as role signatures can be used to both authenticate users and make access control decisions. Moreover, our approach also allows aggregation of signing keys, so that the keys can be associated with multiple roles.

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.

# 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] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell' Agnello, Á. Frohner, K. Lőrentey, and F. Spataro. From gridmap-file to VOMS: Managing authorization in a Grid environment. *Future Generation Computer Systems*, 21(4):549–558, April 2005.

[3] W. Bagga and R. Molva. Policy-based cryptography and applications. In A.S. Patrick and M. Yung, editors, *Proceedings of the 9th International Conference on Financial Cryptography and Data Security (FC 2005)*, pages 72–87. Springer-Verlag LNCS 3570, February 2005.

[4] 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.

[5] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2005*, pages 440–456. Springer-Verlag LNCS 3494, 2005.

[6] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology - Proceedings of CRYPTO 2001*, pages 213–229. Springer-Verlag LNCS 2139, August 2001.

[7] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2003*, pages 416–432. Springer-Verlag LNCS 2656, May 2003.

[8] R. Butler, V. Welch, D. Engert, I. Foster, S. Tuecke, J.Volmer, and C. Kesselman. A national-scale authentication infrastructure. *IEEE Computer*, 33(12):60–66, 2000.

[9] D.W. Chadwick and A. Otenko. The PERMIS X.509 role based privilege management infrastructure. *Future Generation Computer Systems*, 19(2):277–289, February 2003.

[10] 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.

[11] 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*, to appear.

[12] 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.

[13] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier, San Francisco, 2004.

[14] 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.

[15] C. Gentry. Certificate-based encryption and the certificate revocation problem. In E. Biham, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2003*, pages 272–293. Springer-Verlag LNCS 2656, May 2003.

[16] C. Gentry and A. Silverberg. Hierarchical ID-Based cryptography. In Y. Zheng, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2002*, pages 548–566. Springer-Verlag LNCS 2501, December 2002.

[17] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In R.N. Wright, S.D.C. di Vimercati, and V. Shmatikov, editors, *Proceedings of the 13th ACM Computer and Communications Security Conference (CCS 2006)*, pages 89–98. ACM Press, October 2006.

[18] P. Gutmann. PKI: It's not dead, just resting. *IEEE Computer*, 35(8):41–49, August 2002.

[19] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of 4th Algorithmic Number Theory Symposium (ANTS-IV)*, pages 385–394. Springer-Verlag LNCS 1838, 2000.

[20] 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.

[21] 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.

[22] 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.

[23] K.G. Paterson. Cryptography from pairings. In I.F. Blake, G. Seroussi, and N.P. Smart, editors, *Chapter 10 of Advances in Elliptic Curve Cryptography*, pages 215–251, Cambridge, 2005. Cambridge University Press, LMS 317.

[24] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Proceedings of the 3rd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, pages 50–59. IEEE Computer Society Press, June 2002.

[25] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In R.N. Wright, S.D.C. di Vimercati, and V. Shmatikov, editors, *Proceedings of the 13th ACM Computer and Communications Security Conference (CCS 2006)*, pages 99–112. ACM Press, October 2006.

[26] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2005*, pages 457–473. Springer-Verlag LNCS 3494, May 2005.

[27] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of the 2000 Symposium on Cryptography and Information Security (SCIS 2000)*, January 2000.

[28] 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.

[29] A. Shamir. Identity-based cryptosystems and signature schemes. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology - Proceedings of CRYPTO '84*, pages 47–53. Springer-Verlag LNCS 196, August 1985.

[30] N.P. Smart. Access control using pairing based cryptography. In M. Joye, editor, *Proceedings of the RSA Conference: Topics in Cryptology - the Cryptographers' Track (CT-RSA 2003)*, pages 111–121. Springer-Verlag LNCS 2612, April 2003.

[31] R. Tamassia, D. Yao, and W.H. Winsborough. Role-based cascaded delegation. In T. Jaeger and E. Ferrari, editors, *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*, pages 146–155. ACM Press, June 2004.

[32] M.R. Thompson, A. Essiari, and S. Mudumbai. Certificate-based authorization policy in a PKI environment. *ACM Transactions on Information and System Security*, 6(4):566–588, November 2003.

[33] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 proxy certificates for dynamic delegation. In *Proceedings of the 3rd Annual PKI R&D Workshop*, pages 42–58, April 2004.

[34] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for Grid services. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12 2003)*, pages 48–61. IEEE Computer Society Press, June 2003.

[35] W. Yao, K. Moody, and J. Bacon. A model of OASIS role-based access control and its support for active security. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, pages 171–181. ACM Press, May 2001.

[36] F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In Y. Zheng, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2002*, pages 533–547. Springer-Verlag LNCS 2501, December 2002.

# A Pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order $q$ for some large prime $q$, where $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ denotes a related multiplicative group. Typically, $\mathbb{G}_1$ is a subgroup of the group of points on a suitable elliptic curve over a finite field, $\mathbb{G}_2$ is obtained from a related finite field, and $\hat{e}$ is obtained from the Weil or Tate pairing on the curve.

An *admissible pairing* in the context of identity-based cryptography is a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

**Bilinear:** Given $P, Q, R \in \mathbb{G}_1$, we have

$$\hat{e}(P, Q + R) = \hat{e}(P, Q) \cdot \hat{e}(P, R) \text{ and}$$
$$\hat{e}(P + Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R).$$

Hence, for any $a, b \in \mathbb{Z}_q^*$, we have

$$\begin{aligned}\hat{e}(aP, bQ) &= \hat{e}(abP, Q) = \hat{e}(P, abQ) \\ &= \hat{e}(aP, Q)^b = \hat{e}(P, Q)^{ab}.\end{aligned}$$

**Non-degenerate:** There exists $P \in \mathbb{G}_1$ such that $\hat{e}(P, P) \neq 1$.

**Computable:** Given $P, Q \in \mathbb{G}_1$, $\hat{e}(P, Q)$ can be efficiently computed.

# B The Gentry–Silverberg HIBS scheme

ROOT SETUP: The root PKG chooses a generator $P_0 \in \mathbb{G}_1$, picks a random secret $s_0 \in \mathbb{Z}_q^*$, and sets $Q_0 = s_0 P_0$. It also selects cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \{0,1\}^* \to \mathbb{G}_1$. The root PKG's master secret is $s_0$ and the system parameters are $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2 \rangle$.

LOWER-LEVEL SETUP: A lower-level entity (lower-level PKG or user) at level $t$ picks a random secret $s_t \in \mathbb{Z}_q^*$.

EXTRACT: For entity $e$ (with identifier $id_1.id_2.\cdots.id_t$), the entity's parent:

- computes $P_t = H_1(id_1.\cdots.id_t) \in \mathbb{G}_1$,
- sets the secret point
$$S_t = \sum_{i=1}^{t} s_{i-1} P_i = S_{t-1} + s_{t-1} P_t,$$ [4]
- defines Q-values by setting $Q_i = s_i P_0$ for $1 \leq i \leq t-1$.

Entity $e$ is given both $S_t$ and the Q-values by its parent.

SIGN: Given a private key $S_t$ and a message $M \in \{0,1\}^*$, $e$ computes

$$h = H_2(id_1.\cdots.id_t, M) \in \mathbb{G}_1 \quad \text{and} \quad \sigma = S_t + s_t h.$$

The algorithm outputs $\langle \sigma, Q_1, \ldots, Q_t \rangle$ as the signature.

VERIFY: Given a signature $\langle \sigma, Q_1, \ldots, Q_t \rangle$ of a message $M$ signed by $e$, the verifier checks if:

$$\hat{e}(P_0, \sigma) = \hat{e}(Q_0, P_1)\hat{e}(Q_t, h) \prod_{i=2}^{t} \hat{e}(Q_{i-1}, P_i),$$

where $h = H_2(id_1.\cdots.id_t, M)$.

To reduce the time taken to perform signature verification, which is particularly useful in situations where many signatures from the same signer need to be verified, the values $\hat{e}(Q_0, P_1)$ and $\prod_{i=2}^{t} \hat{e}(Q_{i-1}, P_i)$ can be pre-computed. This means that only two pairing computations are required when verification is performed.

# C Aggregation of signing keys

AGGSIGN: Given signing keys $S_{t,1}, \ldots, S_{t,m}$ and a message $M$, the signer: chooses a secret value $s_{agg} \in \mathbb{Z}_q^*$; computes

$$h = H_2(id_1.\cdots.id_{t-1}.(id_{t,1}||\ldots||id_{t,m}), M);$$

computes

$$\sigma = s_{agg} h + \sum_{i=1}^{m} S_{t,i}$$

and $Q_{agg} = s_{agg} P_0$. The algorithm outputs the signature $\langle \sigma, Q_1, \ldots, Q_{t-1}, Q_{agg} \rangle$.

[4] Note that $S_{t-1}$ is the parent's secret point and $s_{t-1}$ is a secret value known only to the parent.

AggVerify**:** Given a signature $\langle \sigma, Q_1, \ldots, Q_{t-1}, Q_{agg} \rangle$ of a message $M$ signed by the keys $S_{t,1}, \ldots, S_{t,m}$, the verifier checks if $\hat{e}(P_0, \sigma)$ is equal to

$$\hat{e}\left(Q_{t-1}, \sum_{i=1}^{m} P_{t,i}\right) \cdot \hat{e}(Q_{agg}, h) \cdot \left(\prod_{i=1}^{t-1} \hat{e}(Q_{i-1}, P_i)\right)^m$$

In order to justify the correctness of the AggVerify algorithm, consider

$$\begin{aligned}
& \hat{e}\left(Q_{t-1}, \sum_{i=1}^{m} P_{t,i}\right) \cdot \hat{e}(Q_{agg}, h) \cdot \left(\prod_{i=1}^{t-1} \hat{e}(Q_{i-1}, P_i)\right)^m \\
= \; & \hat{e}\left(s_{t-1}P_0, \sum_{i=1}^{m} P_{t,i}\right) \cdot \hat{e}(s_{agg}P_0, h) \cdot \left(\prod_{i=1}^{t-1} \hat{e}(s_{i-1}P_0, P_i)\right)^m \\
= \; & \hat{e}\left(s_{t-1}P_0, \sum_{i=1}^{m} P_{t,i}\right) \cdot \hat{e}(s_{agg}P_0, h) \cdot \left(\prod_{i=1}^{t-1} \hat{e}(P_0, s_{i-1}P_i)\right)^m \\
= \; & \left(\prod_{i=1}^{m} \hat{e}(s_{t-1}P_0, P_{t,i})\right) \cdot \hat{e}(s_{agg}P_0, h) \cdot \left(\hat{e}\left(P_0, \sum_{i=1}^{t-1} s_{i-1}P_i\right)\right)^m \\
= \; & \left(\prod_{i=1}^{m} \hat{e}(P_0, s_{t-1}P_{t,i})\right) \cdot \hat{e}(P_0, s_{agg}h) \cdot \left(\hat{e}\left(P_0, S_{t-1}\right)\right)^m \\
= \; & \left(\prod_{i=1}^{m} \hat{e}(P_0, s_{t-1}P_{t,i})\right) \cdot \hat{e}(P_0, s_{agg}h) \cdot \left(\prod_{i=1}^{m} \hat{e}\left(P_0, S_{t-1}\right)\right) \\
= \; & \left(\prod_{i=1}^{m} \hat{e}(P_0, s_{t-1}P_{t,i}) \cdot \hat{e}(P_0, S_{t-1})\right) \cdot \hat{e}(P_0, s_{agg}h) \\
= \; & \left(\prod_{i=1}^{m} \hat{e}(P_0, S_{t-1} + s_{t-1}P_{t,i})\right) \cdot \hat{e}(P_0, s_{agg}h) \\
= \; & \left(\prod_{i=1}^{m} \hat{e}(P_0, S_{t,i})\right) \cdot \hat{e}(P_0, s_{agg}h) \\
= \; & \hat{e}\left(P_0, s_{agg}h + \sum_{i=1}^{m} S_{t,i}\right) \\
= \; & \hat{e}(P_0, \sigma)
\end{aligned}$$