# Resource Management with X.509 Inter-domain Authorization Certificates (InterAC)

Vishwas Patil[1], Paolo Gasti[2], Luigi Mancini[3], and Giovanni Chiola[2]

[1] Cryptography and Security Department
Institute for Infocomm Research, Singapore
vtpatil@i2r.a-star.edu.sg
[2] Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova, Italy
{gasti,chiola}@disi.unige.it
[3] Dipartimento di Informatica
Università di Roma – La Sapienza, Italy
mancini@di.uniroma1.it

**Abstract.** Collaboration among independent administrative domains would require: i) confidentiality, integrity, non-repudiation of communication between the domains; ii) minimum and reversible modifications to the intra-domain pre-collaboration setup; iii) maintain functional autonomy while collaborating; and, iv) ability to quickly transform from post-collaboration to pre-collaboration stage. In this paper, we put forward our mechanism that satisfies above requirements while staying within industry standards so that the mechanism becomes practical and deployable. Our approach is based on X.509 certificate extension. We have designed a non-critical extension capturing users' rights in such a unique way that the need for collaboration or the post-collaboration stage does not require update of the certificate. Thus, greatly reducing the revocation costs and size of CRLs. Furthermore, rights amplification and degradation of users from collaborating domains into host domain can be easily performed. Thus, providing functional autonomy to collaborators. Initiation of collaboration among two domains require issuance of one certificate from each domain and revocation of these certificates ends the collaboration – ease of manageability.

**Keywords:** inter-domain authorization, collaboration, access control, PKI, manageability.

## 1 Introduction

In the age of globalization, organizations have to collaborate to stay competitive so that they can concentrate on their core competencies. A collaboration happens in several forms like; outsourcing, workflow integration. Collaboration is an agreement between two or more organizations to achieve a common goal. It can be short-term or long-term. To initiate a collaboration, the organizations share their users and resources. An organization allowing users from collaborator's domain to perform actions on its resources is called host domain. And, a domain is an independent administrative domain

when the state of its users, resources, and their relations, is readily available within the domain. Change in state of an administrative domain happens when the need for change in access control arises. The internal change in state, at times, may be needed to communicate across collaborating domains. A mechanism that facilitates collaboration should ensure that the internal state changes of a domain should not always necessitate the change to be communicated to peer collaborating domains. That is, the mechanism should allow internal state changes in a host domain while keeping the cost of inter-domain communication for such state changes to a minimum. Let us list out the other requirements from a collaboration mechanism and the rationale behind our mechanism.

As collaborators open up their resources for users from collaborating domains, off-line authentication of the users, and confidentiality, integrity, non-repudiation of communication between the domains becomes important. These properties can be easily achieved through a public-key infrastructure (PKI) like X.509; which is a widely deployed ITU (International Telecommunication Union) standard across organizations. Building a collaboration facilitating mechanism around X.509 keeps the mechanism practical and widely acceptable. Several other collaboration facilitating mechanisms exist that rely on X.509 for authentication and communication security but perform actual authorization decisions through other means. We are interested in finding a solution within X.509 specifications because this is the bare minimum common thing two diverse organizations would have.

The collaboration facilitating mechanism should also keep the modifications needed in intra-domain setup to a minimum in order to quickly gear up for collaboration. Also, such modifications should be reversible so that in case of unsuccessful collaboration, due to unforeseen reasons, the domain quickly reverts back to its pre-collaboration status. This property is very important for successful but ephemeral collaborations.

It is paramount to maintain the functional autonomy of collaborating domains during the collaboration period. That is, the fact of being in collaboration state should not hinder a collaborating domain from performing a task that could be performed in its pre-collaboration state. Depending upon the type of collaboration facilitating mechanism, there is a cost associated with functional autonomy of a domain. The cost can be quantized in terms of the number of revocation of assertions (therefore, size of CRL and associated overheads) performed and communicated across domains. The functional autonomy should also allow the domain, from which users are accessing resources of collaborating domain, to degrade or amplify rights over collaborator's resources apart from resource owner doing the same.

Post collaboration, it is equally important to see how quickly a domain can fall-back to its pre-collaboration state. If the modifications to the pre-collaboration setup are kept to a minimum and non-intrusive, it is evident that post collaboration a domain can quickly fall back to its pre-collaboration state.

Having listed the expectations from a collaboration facilitating mechanism we should also note the fact about digital certificate around which we are building our mechanism. Digital certificates are static, off-line verifiable, cryptographic data structures. The static nature of certificates limits the later rights (authorizations/permissions) amplification or reduction and collaborators sharing resources may not always know the complete authorization requirements *a priori*. Off-line verifiability of certificate does guarantee

the freshness of assertions made via that certificate. Despite these facts, digital certificates provide tangible assertions which can be relied upon with varying degree of trust and context under which they are used. Revocation or suspension of a single permission/right over a collaborating resource requires appropriate changes in permissions previously conferred on users participating from peer domain. Therefore, we started this work to investigate to find whether it is possible to re-arrange permissible rights on a shared resource so that the number of certificate revocations/issuance are minimized when rights are withdrawn/added. We could address this quandary by introducing two things in our mechanism: segregation of permissions/rights and hierarchy in flow of permission. This our approach brought huge advantage, in terms of number of certificate revocations, autonomy, manageability. These benefits under our approach come with a slight computational cost which is justifiable.

*Organization of the paper:* In the next Section, we take a stock of current relevant works on the lines of cross-domain authorization mechanisms based on digital certificates and policy languages. In Section 3 we give the rationale behind our approach and present our mechanism in Section 4. In Section 5 we show how our mechanism brings functional autonomy and manageability to collaborators while in collaboration. Section 6 gives the algorithm for certificate chain composition and rights computation. In Section 7 we compare our approach with existing certificate based approach in terms of computational cost and functionality. We conclude in Section 8.

## 2 Background and Related Work

Several proposals exist in literature to address collaboration in distributed environment. Most of these proposal are policy based approaches in which certificates are used as assertions and actual authorizations of a user are computed based on policy-based language. In authentication-cum-authorization approach [1] certificates play a role of identity authentication and in policy based approach they play a role of conveying assertions. In a dynamic distributed setup, policy based authorization mechanism provide a better solution over authentication-cum-authorization mechanism. Policy based authorization mechanisms [2–9] overcome the shortcomings like, for example, context-sensitive authorizations, dynamic rights amplification, suspension or degradation of rights. Certificates are prone to revocation in a dynamic setup if one does not carefully choose the "security assertion values" (permissions) to be embedded into the certificates. It is a common practice to insert only the information that is not going to change for a relatively longer time period, and dynamic information is captured and interpreted separately, using a policy language [2–5, 8]. The problem overlooked by existing approaches is to make a systematic distinction between dynamic and static information (permissions), which we feel is almost impossible or cannot be precisely captured *a priori* while issuing the certificates [10]. Through our approach, we put forward a mechanism that shields the authorization certificates from the need of revocation/re-issuance in synchronization with the dynamic state changes in a domain.

X.509 was originally conceived to authenticate the entries in X.500 directory structure. Later on, it was exploited to perform authentication-cum-authorization decisions over resources scattered across independent administrative domains. Efforts to embed

authorizations of a subject into the certificate itself were made through certificate extensions [11]. The obvious challenge in such an approach of embedding is to maintain the certificate's validity due to change in subject's authorization status. This challenge led to the need for separating authentication and authorization of a subject, and attribute certificates [12] were conceived. Attribute certificates provide the foundation upon which the Privilege Management Infrastructure (PMI) can be built. They don't contain any public key, but attributes that may specify group membership, role, security clearance or other authorization information corresponding to the attribute certificate holder. A subject may have multiple attribute certificates associated with each of its PKCs. There is no requirement that the same authority create both the public key certificate and attribute certificate(s) for a user. This also brought along the need for attribute authority (AA, similar to Certificate Authority – CA) and attribute revocation lists (ARLs). PERMIS [13], Akenti [14], Argos [15], Shibboleth [16], CAS/Globus [17], WS-Security [6], SALSA [18], etc., are some of the existing inter-domain authorization mechanisms or frameworks that mainly rely on X.509 type of PKI. There also exist policy based approaches like PolicyMaker/KeyNote [2, 3] that use cryptographic security assertions to derive to an authorization decision. RBAC (Role-Based Access Control [19]) is a *de facto* standard in industry to perform authorizations over an organization's resources by its users. In [20, 1, 21], the authors propose a X.509 based approach to extend the framework of RBAC across domains. There also exist standards like SAML [4], XACML [5], RT/RTML [8, 9] meant for designing interoperation interfaces for organizations that need to collaborate. Specification languages [4, 5] and frameworks [6] have gathered much relevance in work-flow and grid computing fields.

A deep analysis of these practices made us conclude that in most of the existing approaches for collaboration, cryptographic primitives are mainly used to perform authentication. The authorization related attributes are specified in XML-like language with a plausible integration of cryptographic primitives over such attributes to provide authenticity and non-repudiation properties for the credentials flowing across domains. Policy-based approaches may not be able to quickly gear up for collaborations as the participating domains may have different policy languages used in their setups. Domains' transition from post-collaboration to pre-collaboration state may not be smooth and quick. Therefore, it was interesting for us to investigate if we could design a mechanism purely within X.509 framework. We would like to quickly highlight that though policy based inter-domain access control mechanisms (SAML, XACML, RT/RTML, *et.al.*) are more expressive than our approach, it would be unfair to compare them with our mechanism as they fall in different categories. We postpone the comparative analysis to Section 7.1.

## 3 Need for Hierarchy and Segregation of Rights

Before we introduce you to our proposal, we would like to underline the need for hierarchy in authorization flow and segregation of rights. In a collaboration realized solely using digital certificates, a collaborator sharing its resource will issue a certificate, to user from peer domain, containing appropriate permissible rights on the resource. Assume two collaborating domains $D_1$ and $D_2$, where $D_2$ is offering its resource $R_2$ for

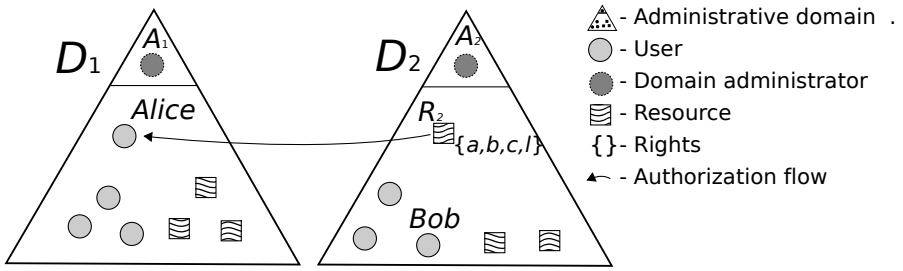


**Fig. 1.** Flow of authorization using direct authorization certificate

collaboration for user *Alice* from peer domain $D_1$. Domain administrator of $D_2$ issues a digital certificate containing permissions $\{a,b,c,d\}$ to *Alice*. When *Alice* needs to access resource $R_2$, she simply makes an access request along with the authorization certificate. The flow of authorization from $R_2$ to *Alice* due to the authorization certificate is shown in Figure 1. Domain $D_2$ starts incurring collaboration cost (affecting functional autonomy) when there is a state change in its domain. For example, $D_2$ needs to withdraw permission *d* over its resource $R_2$. This change requires revocation and re-issuance of certificate to *Alice*. The cost is directly proportional to the number of collaborating users having access to resource $R_2$. Imagine $D_2$ sharing several other of its resources with $D_1$ and most of these resources having some permissions that are frequently enabled/suspended. Introduction of a new permission will either require re-issuance of certificates or issuance of separate certificates containing new permission.

To reduce the collaboration costs to participating domains and to retain their functional autonomy we propose a novel approach in which we segregate the permissions on collaborating resource into static and dynamic sets. Static permission are those permission that are less likely to be withdrawn by resource administrator for a relatively longer period as compared to dynamic permissions that may be suspended (temporarily or permanently) frequently. We also introduce a level of indirection in the flow of authorization flowing from resource to its collaborating users. In next section, we give the details of our approach with the help of a running example.

## 4 *InterAC*: Dynamic Inter-domain Authorization via X.509

In this section, we explain our collaboration facilitating mechanism *InterAC*. *InterAC* is purely within X.509 specifications. Under *InterAC* we have designed a *non-critical* extension to X.509 digital certificate [Appendix A]. Through this extension we allow segregation of permissions on a collaborating resource. *InterAC* uses its type of certificates for the following three purposes:

- for subject binding,
- to define an ACL over resource, and
- as a collaboration agreement.

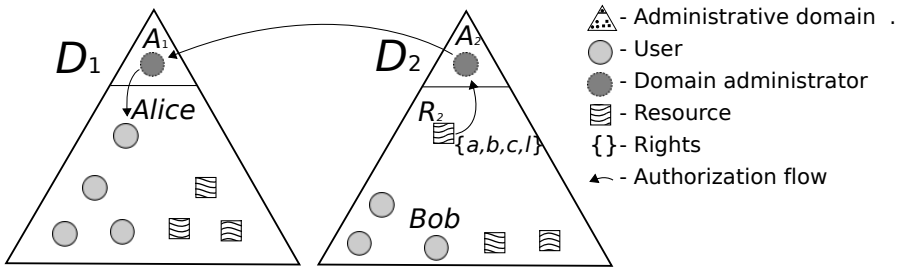


**Fig. 2.** Flow of authorization using indirect authorization certificate

Using such certificates, *InterAC* allows a domain administrator to initiate collaboration and define the flow of authorization over its resource from collaborating users from peer domain. In Fig. 2 one such flow of authorization from resource $R_2$ to user *Alice* is shown. In short, to access a resource from a collaborating domain, a user need to compose a chain of certificates that proves a valid flow of authorization from the resource to the user. The chain composition and evaluation algorithm is explained in Section 6. Let us explain the syntax of *InterAC* certificate and semantics behind it.

Let us denote the *InterAC* digital certificate as CERTIFICATE. Let $D_1$ and $D_2$ be two independent administrative domains willing to collaborate. That is, for example, $D_2$ agreeing to share its resources with the users from domain $D_1$ as shown in Fig. 2. To share resource $R_2$, $A_2$ – domain administrator of $D_2$ – issues a special type of certificate to $R_2$ that $R_2$ will use as an ACL for requests coming from collaborating users. The short hand notation of this CERTIFICATE is: $R_2 \longrightarrow R_2 \boxed{\{a,b,c\},\{l,m,n\}}$. Where, $\{a,b,c\}$ are the set of static permissions and $\{l,m,n\}$ are the set of dynamic permissions. As a next step, $A_2$ takes ownership of this resource: $R_2 \longrightarrow A_2 \boxed{\{a,b,c\},\{l,m,n\}}$.

To initiate a collaboration with domain $D_1$, $A_2$ confers rights on $R_2$ to $A_1$ – the domain administrator for $D_1$. Therefore, $A_2 \longrightarrow A_1 \boxed{\{a,b,c\},\{l\}}$. In turn, $A_1$ issues a CERTIFICATE to *Alice* so that *Alice* contributes to collaboration: $A_1 \longrightarrow Alice \boxed{\{*\},\{*\}}$. The wild character in permission set has a special meaning under *InterAC*. It signifies that the decision to grant permissions to the subject of the certificate has been deferred or, in other words, the subject of the certificate can perform all possible actions provided that the subject comes up with a valid proof (certificate chain showing flow of authorization from resource to the requester). Intuitively, Alice can perform $\{a,b,c,l\}$ rights on resource $R_2$ with the above set of certificates. The computation of effective rights of a requester are done by taking a positional intersection[4] over permissions present in the certificates used for composition of proof. This indirect flow of authorization (hierarchy) from the resource to *Alice* allows each intermediate principal to decide the actual

[4] Though the permissions in static and dynamic sets is treated similarly, i.e., a simple intersection across respective sets in the CERTIFICATE chain, it is important that across all the participating collaborative domains the CERTIFICATES should be issued with a consistent position of permission set – static permission set followed by dynamic permission set.

set of permissions *Alice* will have over resource $R_2$, at any given time. In the following section we shall see how this introduction of hierarchy and segregation of permissions into static and dynamic set contributes to autonomy and manageability of collaboration.

It is interesting to note that as there is no mention of exact rights *Alice* has been given, the same certificate can be used by *Alice* to participate in collaborations with other domains. Since the extension is *non-critical*, the certificate can be used as an identification certificate by *Alice* for purposes other than collaboration. As no rights are specified inside *Alice*'s certificate, privacy violations do not happen when this authorization certificate is used for just authentication purpose.

## 5 Bringing Autonomy and Manageability to Collaborators

Continuing with the setup shown in Fig. 2 we will introduce more scenarios to explain utility of *InterAC* towards autonomy and manageability. Let us begin with an example of effective rights computation for user *Alice* with a sample scenario.

In the following $D_2$'s domain administrator $A_2$ is preparing $R_2$ for collaboration with $\{a,b,c\}$ permissions.

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\},\{\ \}} \tag{1}$$

where, $\boxed{\{a,b,c\},\ \{\ \}}$ is the authorization string in the CERTIFICATE used as an ACL on resource $R_2$. Note that $D_2$ has abstained from conferring permission $c$ over resource $R_2$ to its collaborator. Let the following be the CERTIFICATE denoting the collaboration agreement between domain $D_1$ and $D_2$, where domain $D_2$ is offering its resource to the users from domain $D_1$;

$$A_2 \longrightarrow A_1 \boxed{\{a,b\},\{*\}} \tag{2}$$

In a slight modification to the setup in domain $D_1$, we introduce two roles $G_1$ and $G_2$ and let *Alice* be part of $G_1$, for time being. Therefore;

$$A_1 \longrightarrow G_1 \boxed{\{a\},\ \{*\}} \tag{3}$$

$$A_1 \longrightarrow G_2 \boxed{\{b\},\ \{*\}} \tag{4}$$

$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}} \tag{5}$$

Therefore, *Alice* constructs the following CERTIFICATE chain to access $R_2$

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\},\ \{\ \}}$$

$$A_2 \longrightarrow A_1 \boxed{\{a,b\},\ \{*\}}$$

$$A_1 \longrightarrow G_1 \boxed{\{a\},\ \{*\}}$$

$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}}$$

And the effective permissions at the disposal of user *Alice* are $\{a\}$, upon positional intersection of permissions present in the CERTIFICATE chain;

$$\text{static } \textit{permissions} = \{a,b,c\} \cap \{a,b\} \cap \{a\} \cap \{*\} = \{a\}$$
$$\textit{and}, \text{ dynamic } \textit{permissions} = \{\} \cap \{*\} \cap \{*\} \cap \{*\} = \{\}$$

Should $A_1$ decide *Alice* to avail permission *b*, $G_2$ issues the following to *Alice*

$$G_2 \longrightarrow \textit{Alice} \boxed{\{b\}, \{*\}} \tag{6}$$

Having shown the CERTIFICATE chain construction and evaluation of effective permissions due to the chain, we would like to show you how each principal in the authorization hierarchy can amplify or degrade effective rights of *Alice*.

### 5.1 Rights Amplification

Rights (alternatively referred as permissions or authorizations) can be amplified, that is, extra permissions can be added to the existing permission-set, by either resource controller or collaboration administrators (on either side of the collaboration). The only condition for rights amplification is the entity performing amplification operation itself should have the permission to be amplified. Following are the three instances of rights amplification that amplify the rights of *Alice*.

**By resource controller.** Resource controller is the fundamental authority to decide the actual set of permissions possible over the resource. Let *m* be a new permission that resource controller wants to make available to its collaborators. To do so, the resource controller updates its ACL (CERTIFICATE) with the following.

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\}, \{m\}} \tag{7}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing $R_2$ in domain $D_2$ becomes;

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\}, \{m\}}$$
$$A_2 \longrightarrow A_1 \boxed{\{a,b\}, \{*\}}$$
$$A_1 \longrightarrow G_1 \boxed{\{a\}, \{*\}}$$
$$G_1 \longrightarrow \textit{Alice} \boxed{\{*\}, \{*\}}$$

And the effective permissions at the disposal of *Alice* are $\{a,m\}$, because;

$$\text{static } \textit{permissions} = \{a,b,c\} \cap \{a,b\} \cap \{a\} \cap \{*\} = \{a\}$$
$$\textit{and}, \text{ dynamic } \textit{permissions} = \{m\} \cap \{*\} \cap \{*\} \cap \{*\} = \{m\}$$

**By host domain administrator.** $A_1$ can perform rights amplification for *Alice* by issuing the following CERTIFICATE.

$$A_1 \longrightarrow G_1 \boxed{\{a,b\}, \{*\}} \tag{8}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing $R_2$ becomes;

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\},\ \{\ \}}$$
$$A_2 \longrightarrow A_1 \boxed{\{a,b\},\ \{*\}}$$
$$A_1 \longrightarrow G_1 \boxed{\{a,b\},\ \{*\}}$$
$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}}$$

And the effective permissions at the disposal of *Alice* are $\{a,b\}$, because;

$$\text{static } permissions = \{a,b,c\} \cap \{a,b\} \cap \{a,b\} \cap \{*\} = \{a,b\}$$
$$and, \text{ dynamic } permissions = \{\ \} \cap \{*\} \cap \{*\} \cap \{*\} = \{\ \}$$

**By peer domain administrator.** $A_2$ can perform rights amplification for the users from its collaborating domain by issuing the following CERTIFICATE

$$A_2 \longrightarrow A_1 \boxed{\{a,b,c\},\ \{*\}} \tag{9}$$

Of course, this amplification will not be reflected in domain $D_1$ until $D_1$ does further rights amplification.

### 5.2 Rights Degradation

In this sub-section we show rights degradation, which is similar to rights amplification but the permissions will be removed from the existing set of permissions available to the principal that is performing rights degradation. Before proceeding to the examples of rights degradation let us bring back the CERTIFICATE states to the pre-amplification steps performed in previous sub-section.

**By resource controller.** As mentioned before, resource controller is the fundamental authority to decide the actual set of permissions possible over the resource. Let *a* be the permission that resource controller wants to make unavailable to its collaborators. To do so, the resource controller updates its ACL (CERTIFICATE) with the following.

$$R_2 \longrightarrow A_2 \boxed{\{b,c\},\ \{\ \}} \tag{10}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing the resource $R_2$ becomes;

$$R_2 \longrightarrow A_2 \boxed{\{b,c\},\ \{\ \}}$$
$$A_2 \longrightarrow A_1 \boxed{\{a,b\},\ \{*\}}$$
$$A_1 \longrightarrow G_1 \boxed{\{a\},\ \{*\}}$$
$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}}$$

And the effective permissions at the disposal of *Alice* are $\{\ \}$, because;

$$\text{static } \textit{permissions} = \{b,c\} \cap \{a,b\} \cap \{a\} \cap \{*\} = \{\ \}$$
$$\textit{and}, \text{ dynamic } \textit{permissions} = \{\ \} \cap \{*\} \cap \{*\} \cap \{*\} = \{\ \}$$

**By host domain administrator.** $A_1$ can perform rights degradation for *Alice* by issuing the following CERTIFICATE

$$A_1 \longrightarrow G_1 \boxed{\{\ \}, \{*\}} \tag{11}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing $R_2$ becomes;

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\}, \{\ \}}$$
$$A_2 \longrightarrow A_1 \boxed{\{a,b\}, \{*\}}$$
$$A_1 \longrightarrow G_1 \boxed{\{\ \}, \{*\}}$$
$$G_1 \longrightarrow \textit{Alice} \boxed{\{*\}, \{*\}}$$

And the effective permissions at the disposal of *Alice* are $\{\ \}$, because;

$$\text{static } \textit{permissions} = \{a,b,c\} \cap \{a,b\} \cap \{\ \} \cap \{*\} = \{\ \}$$
$$\textit{and}, \text{ dynamic } \textit{permissions} = \{\ \} \cap \{*\} \cap \{*\} \cap \{*\} = \{\ \}$$

**By peer domain administrator.** $A_2$ can make use of rights degradation facility to achieve an important aspect required in collaboration – temporary suspension of collaboration. To do so, $A_2$ issues the following CERTIFICATE

$$A_2 \longrightarrow A_1 \boxed{\{b\}, \{\ \}} \tag{12}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing $R_2$ is;

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\}, \{*\}}$$
$$A_2 \longrightarrow A_1 \boxed{\{b\}, \{\ \}}$$
$$A_1 \longrightarrow G_1 \boxed{\{a\}, \{*\}}$$
$$G_1 \longrightarrow \textit{Alice} \boxed{\{*\}, \{*\}} \tag{13}$$

And the effective permissions at the disposal of *Alice* are $\{\ \}$, because;

$$\text{static } \textit{permissions} = \{a,b,c\} \cap \{b\} \cap \{a\} \cap \{*\} = \{\ \}$$
$$\textit{and}, \text{ dynamic } \textit{permissions} = \{*\} \cap \{\ \} \cap \{*\} \cap \{*\} = \{\ \}$$

Several combinations of rights amplification and degradation can be engineered by resource controller and corresponding domain administrator, independently or collectively to achieve desired effects in the availability of permissions to the users from collaborating domain.

### 5.3 Rights Suspension

This operation is a special instance of rights degradation. The resource controller can take down the resource temporarily for various reasons by issuing the following CERTIFICATE.

$$R_2 \longrightarrow A_2 \boxed{\{\}, \{\}} \tag{14}$$

Based on the internal dynamics (state changes) of the domain sharing resources, domain administrators can roughly estimate life expectancy (certificate validity period) of CERTIFICATES at different hierarchy levels. We assume that domain administrators issue/revoke *InterAC* certificates to users and resources. The domain administrators are also responsible to initiate the collaboration (by issuing authorization certificate to peer domain administrator). We also assume that the semantics of permissions embedded inside the *InterAC* certificate issued for collaboration initiation is agreed upon. PKI Resource Query Protocol (PRPQ) [22] is a promising utility for seamless, dynamic integration of resources across independent administrative domains.

## 6 Chain Composition and Evaluation

In this section we provide an algorithm to compute a valid CERTIFICATE chain. We assume that the users of a collaborative domain have been made available with the set of CERTIFICATES that affect the permission-set of the user. The onus of authorization proof generation is on the requester of the resource. We continue referring to principals ($R_2$, $A_1$, *Alice*, etc.) from the scenarios presented in previous sections.

**Composition of CERTIFICATE chain:** Authorization proof construction (performed by requester)

- **CERTIFICATE validation** – Discard CERTIFICATES whose validity has expired or stand revoked.
- **Filter CERTIFICATES** – Include CERTIFICATES containing the permission for which request is being made in its authorization string. Discard CERTIFICATES with $\{\}, \{\}$ in its authorization string (i.e., empty static and dynamic permission-sets).
- **Construct directed graph** – For each principal (issuer or subject of a certificate) add a vertex to the graph. For each CERTIFICATE put a directed edge originating in the "issuer" vertex and ending in "subject" vertex.
- **Find path** – Find all possible paths starting in the vertex denoted by the principal "resource controller" (i.e., $R_2$) and terminating in the vertex denoted by the principal "requester" (i.e., *Alice*)
- **Purge paths** – Discard paths in which the positional intersection of the permission under consideration leads to an empty set
  If no paths are left after **Purge paths** step, a valid authorization proof is not available.

**Evaluation of CERTIFICATE chain:** Authorization proof verification (performed by verifier)

**CERTIFICATE validation** – Check CERTIFICATES in authorization proofs for their validity and revocation status.

**Intersection** – Take positional intersection over the authorization strings present in the CERTIFICATES of the authorization proofs.
Access will be granted with effective permissions evaluated upon positional intersection.

## 7 Comparative Analysis

In this section we would like to compare our mechanism with a mechanism that does not treat permissions as we do. The closest contender of such an approach is the X.509 attribute certificate framework defined in [11] that provides the foundation upon which the Privilege Management Infrastructure (PMI) can be built. This framework has been the most commonly used approach to realize inter-domain authorizations.

The PMI approach for inter-domain authorization has following shortcomings: i) size of ARL (attribute revocation list) keeps on growing as the number of collaborating domains of a host domain go on increasing when the state of collaborating domain changes. ii) each collaborating domain of a host domain necessitates issuance of attribute certificates to the users of host domain – collaboration-specific certificates that expire upon completion of collaboration and may be added to ARL. iii) such an approach of embedding exact set of permission-set into user's certificate leaves no scope for later rights amplification or degradation. iv) and, lack of functional autonomy and manageability.

Intuitively, under *InterAC* the number of users in peer domain do not *proportionally* influence the cost of any operation performed towards collaboration. That is, the cost to establish/break a collaboration or to do rights amplification/degradation/suspension is constant.

Table 1 compares our approach with the traditional PMI approach using the example discussed in section 4, as a test-bed; where $n$ is the number of collaborating users and $h$ is the length of CERTIFICATE chain or depth of authorization flow hierarchy. We assume that $A_1$ already issued the appropriate certificates to its users and that there has been no previous interaction between domains $D_1$ and $D_2$. The comparison also assumes that the "push" model is adopted for the PMI [12]. The computational cost introduced by *InterAC* on resource $R_2$ is greater than the computational cost in traditional PMI. This is because, in PMI the authorizations are asserted in one or few attribute certificates, while in our approach the authorizations must be calculated by positional intersection of the authorizations contained in the CERTIFICATE chain. The actual computational overhead is given in Appendix A. We feel the cost overhead is justifiable given the numerous advantages our mechanism brings in for collaboration.

### 7.1 *InterAC* in Perspective of Policy-based Mechanisms

To facilitate collaboration among independent administrative domains, two other distinct research tracks exist: i) policy-based languages (e.g., [3–6]) that allow to capture collaboration requirements, and ii) extensions to RBAC model (e.g., [23, 24, 1, 7–9]).

| | ***InterAC*** | **PMI-based mechanisms** |
|---|---|---|
| Cost of collaboration initiation | $O(1)$<br>Issuance of certificate by a domain administrator to peer domain's administrator. It is assumed that collaboration-independent *InterAC* certificates have been already issued in participating domains. | $O(n)$ [a]<br>Since the authorization certificates are specific to a collaboration, new certificates need to be issued each time a new collaboration is initiated. |
| Cost of incoming authorization request verification | $O(h)$ [b] | $O(1)$ |
| Cost of rights amplification or degradation | $O(1)$ | $O(n)$ |
| Cost of rights suspension | $O(1)$ | $O(n)$ - by revoking all user certs<br>$O(1)$ - by updating the resource ACL, which also disables the access to the resource for users in host domain |
| Cost to revert to pre-collaboration state | $O(1)$ | $O(n)$ |

[a] $n$ – number of participating users from a collaborating domain

[b] $h$ – authorization hierarchy or length of the CERTIFICATE chain

**Table 1.** *InterAC* vs. PMI-based mechanisms *w.r.t.* certificate issuance/verification/revocation cost to a collaborating domain

These approaches have more expressive power as compared to *InterAC*. We say so because *InterAC* does not provide a language to capture context-aware decisions, neither it provides fancy constructs like separation-of-duty as under RBAC family. We refrained from devising an accompanying language in our proposal because all the above mentioned policy/model-based proposals face interoperability issues. We observe that the minimum common that the administrative domains willing to collaborate have is a PKI (digital certificates). *InterAC* provides the basic requirements of collaboration purely through non-intrusive certificate extension. The policy/model-based mechanisms for collaboration use digital certificates as assertions and take access control decisions based on such set of assertions and plausibly other contexts. The *InterAC* certificates can also be used as assertions thus enriching the higher level policy/model-based approaches.

## 8 Conclusion

In this paper, we have argued that it is possible to realize flexible inter-domain authorizations within the X.509 specifications, which is the most widely deployed type of

PKI across the industry. We have shown how our X.509 extension helps collaborators maintain their functional autonomy. The use of $\{*\},\{*\}$ as an authorization string in leaf certificates allow users to participate in any collaboration initiated by its domain, thus reducing the number of certificates a user need to maintain, obviously reducing the size of CRL. The feature of rights amplification and degradation was not possible under any other X.509-compatible approach. The ability to quickly initiate/break collaborations while maintaining domain's functional autonomy is specially very useful for ephemeral collaborations. The performance analysis of our implementation showed that the additional cost introduced by our proposal is usually negligible compared to the benefits *InterAC* offers.

In RBAC framework, a role is a set of permissions. Therefore, the treatment we provide to permissions in our approach can be easily extended to roles when the collaborating domains have RBAC as their underlying access control framework. The static and dynamic permission sets can be further supplemented with an additional set whose members may carry semantics for context-aware, exception-tolerating authorization.

**Acknowledgements:** The authors would like to thank the anonymous reviewers of EuroPKI 2009 workshop and the shepherd Massimiliano Pala for providing useful comments, observations, and suggestions that helped us in improving the paper.

## References

1. Linn, J., Nyström, M.: Attribute certification: an enabling technology for delegation and role-based controls in distributed environments. In: RBAC '99: Proc. of the $4^{th}$ ACM workshop on Role-based access control. (1999) 121–130
2. Blaze, M., Feigenbaum, J., Strauss, M.: Compliance Checking in the PolicyMaker Trust Management System. Financial Cryptography, FC 1998 **LNCS 1465** (1998) 254–274
3. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.: The KeyNote Trust-Management System Version 2. RFC 2704, IETF (1999)
4. Security Assertion Markup Language. OASIS Std., `http://www.oasis-open.org/committees/security` (2005)
5. eXtensible Access Control Markup Language. OASIS Std., `http://www.oasis-open.org/committees/xacml` (2005)
6. Web Services Security v1.1: (OASIS standards) `http://www.oasis-open.org/specs/index.php#wssv1.1`.
7. Joshi, J.B.D., Bhatti, R., Bertino, E., Ghafoor, A.: Access-control language for multidomain environments. IEEE Internet Computing **8**(6) (2004) 40–50
8. Li, N., Mitchell, J.C., Winsborough, W.H.: Design of a role-based trust-management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, IEEE Computer Society Press (2002) 114–130
9. Li, N., Mitchell, J.C., Winsborough, W., Seamons, K., Halcrow, M., Jacobson, J.: RTML: A Role-based Trust-management Markup Language. Technical report, (Purdue University)
10. Patil, V., Shyamasundar, R.K.: Towards a Flexible Access Control Mechanism for E-Transactions. In: EGCDMAS '04: International Workshop on Electronic Government, and Commerce: Design, Modeling, Analysis and Security, INSTICC (2004) 66–81
11. ITU X.509 Recommendations: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks (2005) `http://www.itu.int/rec/T-REC-X.509/en`.

12. Farrell, S., Housley, R.: An Internet Attribute Certificate Profile for Authorization. RFC 3281, IETF (2002)
13. Chadwick, D.W., Otenko, A.: The PERMIS X.509 Role Based Privilege Management Infrastructure. In: SACMAT '02: Proc. of ACM Symp. on Access Control Models & Tech. (2002) 135–140
14. Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., Essiari, A.: Certificate-based Access Control for Widely Distributed Resources. In: $8^{th}$ USENIX Security Symp. (1999) 215–228
15. Jonscher, D., Dittrich, K.R.: Argos – Configurable Access Control System for Interoperable Environments. In: Proc. of the $9^{th}$ annual IFIP TC11 WG11.3 working conf. on Database security IX : status and prospects, Chapman & Hall Ltd. (1996) 43–60
16. Shibboleth, `http://shibboleth.internet2.edu/` (2005)
17. CAS - Community Authorization Service. The Globus Alliance, (`http://www.globus.org/grid_software/security/cas.php`)
18. Kang, M.H., Park, J.S., Froscher, J.N.: Access Control Mechanisms for Inter-organizational Workflow. In: SACMAT '01: Proc. of ACM Symp. on Access Control Models & Tech. (2001) 66–74
19. Ferraiolo, D.F., Sandhu, R.S., Gavrila, S.I., Kuhn, D.R., Chandramouli, R.: Proposed NIST Standard for Role-based Access Control. ACM Trans. on Info. and Sys. Sec. **4**(3) (2001) 224–274
20. Herzberg, A., Mass, Y., Michaeli, J., Ravid, Y., Naor, D.: Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. In: SP '00: Proc. of the IEEE Symp. on Security and Privacy. (2000) 2–14
21. Shands, D., Yee, R., Jacobs, J., Sebes, E.J.: Secure Virtual Enclaves: Supporting Coalition use of Distributed Application Technologies. ACM Trans. Inf. Syst. Secur. **4**(2) (2001) 103–133
22. PRPQ: (OpenCA PKI Project)
23. Cohen, E., Thomas, R.K., Winsborough, W., Shands, D.: Models for coalition-based access control (CBAC). In: SACMAT'02: Proc. of ACM Symp. on Access Control Models & Tech. (2002) 97–106
24. Chadwick, D., Dimitrakos, T., Dam, K.K.V., Randal, D.M., Matthews, B., Otenko, A.: Multi-layer privilege management for dynamic collaborative scientific communities. In: Workshop on Grid Security Practice and Experience, Oxford. (2004) II 7–14
25. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: A Community Authorization Service for Group Collaboration. In: POLICY '02: Proc. of the $3^{rd}$ International Workshop on Policies for Distributed Systems and Networks. (2002) 50–59
26. Housley, R., Polk, T., Ford, W., Solo, D.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, IETF (2002)
27. Clarke, D., Elien, J.E., Ellison, C., Fredette, M., Morcos, A., Rivest., R.: Certificate Chain Discovery in SPKI/SDSI. Journal of Computer Security **9(4)** (2001) 285–322
28. Denker, G., Millen, J., Miyake, Y.: Cross-Domain Access Control via PKI. In: POLICY '02: Proc. of the $3^{rd}$ International Workshop on Policies for Distributed Systems and Networks. (2002) 202–205
29. Fisher, J.L.: Side-Effects of Cross-Certification. In: $4^{th}$ PKI R&D Workshop. (2005) `http://middleware.internet2.edu/pki05/proceedings/fisher-cross_cert.pdf`.
30. Ford, W., Baum, M.S.: Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption, $2^{nd}$ Ed. Prentice Hall (2002)
31. Gasti, P., Patil, V.: Interdomain Access Control (2006) `http://www.disi.unige.it/person/GastiP/publications/interac/`.

32. Harrington, A., Jensen, C.: Cryptographic Access Control in a Distributed File System. In: SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies, ACM Press (2003) 158–165

# Appendix

## A Implementation and Performance Analysis

### A.1 Extension's Structure

The ASN.1 notation for our X.509v3 certificate extension is depicted in Fig. 3. `Static`

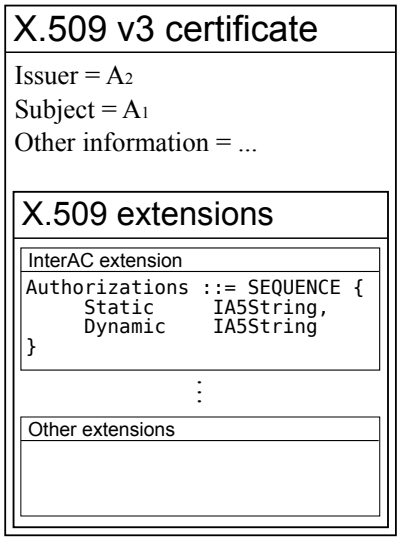


**Fig. 3.** Sample X.509v3 certificate with the *InterAC* extension

and `Dynamic` are two strings which hold the permissions that are less prone (non-volatile) and more prone (volatile) to frequent modifications, respectively. Effective permissions of a certificate's subject are captured in two distinct sets. Depending upon the requirements, authorization delegation authority may include either a "$*$" or a " " (null) or a comma-separated list of permissions in any of the set. For example, `static` = `a,b,c` means that the set of non-volatile permissions for the certificate's subject are $\{a, b, c\}$, where $a, b$ and $c$ represent three different permissions.

### A.2 Performance Analysis

Our prototype implementation [31] is done in C (gcc 4.1.2) on a Pentium III (933MHz processor with 512MB RAM) hardware running GNU/Linux (kernel 2.6). The cryptographic primitives are supported by OpenSSL library (0.9.8e). The certificates used

for measuring performance results are generated with 1024-bit RSA public keys. An inter-domain authorization request consists of a well-formed sequence of digital certificates as a proof of credentials. The resource controller verifies such certificate chains before granting access. The algorithm to perform verification is given in Section 6. The performance results of our approach against PMI-based approach is summarized in the graph shown in Fig. 4.

The graph is plotted for two different authorization proof chains consisting certificates with different extension types: i) typical authorization extension (i.e., without segregated permission-sets), and ii) our extension. The slight increase in the computational cost for our approach is justifiable by the benefits it provides. Taking a closer look at the difference between the two values we observe that it is around 1% on an average. For chains with realistic length (i.e., composed of 15 certificates or less), the actual computational cost overhead is around 0.5*ms* in our operating environment.

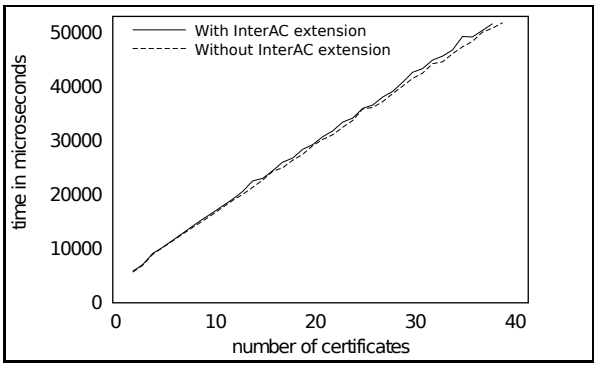


**Fig. 4.** Time to evaluate certificates with different types (*InterAC* and non-*InterAC*) of authorization extensions