J. Lopez, et al., "Integrating PMI Services in CORBA Applications", *Computer Standards & Interfaces*, vol. 25, pp. 391-409, 2003.
NICS Lab. Publications: https://www.nics.uma.es/publications

Integrating PMI services in CORBA Applications

Abstract

Application-level access control is an important requirement in many distributed environments. For instance, in new scenarios such as e-commerce, access to resources by previously unknown users is an essential problem to be solved. The integration of Privilege Management Infrastructure (PMI) services in the access control system represents a scalable way to solve this problem. Within the CORBA standards, the Resource Access Decision (RAD) facility is a mechanism used by security-aware applications to obtain authorization decisions and to manage access decision policies. This paper presents PMI-RAD, an approach to integrate the services of an external PMI into CORBA applications using the RAD facility. In particular, the integration of the external PMI in the access control system is based on the semantic description of the PMI services. Our RAD implementation requests and verifies attribute certificates from the PMI in a transparent way for CORBA objects.

Keywords: CORBA, Distributed systems security, Privilege Management Infrastructures, XML metadata.

1 Introduction

Distributed applications are playing an increasingly important role motivated by the need of scalable and open environments that support the expansion of organizations computing boundaries. As applications move from corporate environments to the Internet, the lack of security and trustworthy mechanisms becomes evident. Security is a critical requirement in scenarios such as electronic commerce and collaborative work, especially if distributed object technology is used as a backbone. Some of the most relevant and interesting domains are middleware, web services and grid computing.

The standardization of services and architectural abstractions provided by *middleware systems* has facilitated the composition of new systems based on the combination of functional components. CORBA is one of the most relevant examples [1]. The fact that middleware systems demand the abstraction[1] of the applications from the underlying mechanisms further confirms the need to revise security services. Access control and use-control are, probably, the most important ones.

New scenarios where distributed objects are emerging share common problems. One of the most relevant general problems is that resources are accessed by previously unknown users; thus, an external authorization mechanism is needed. Other related issues are the high degree of flexibility

[1] It is necessary to realize that achieving a complete abstraction is not possible because some of the security mechanisms are designed to be used by security-aware applications.

required because of the heterogeneous nature of the resources, the ability to change the access control parameters dynamically and transparently, and the establishment of access conditions in an automatic way based on information about the objects.

Paradoxically, authorization in distributed systems often relies on centralized access control management, but centralized control has important disadvantages: (a) the control point represents a weak part against security attacks and fault tolerance, (b) does not facilitate the deployment of owner-retained-control mechanisms, (c) reduces system performance because it introduces a bottleneck for request handling and, (d) usually, enforces homogeneous access control schemes that do not fit naturally in heterogeneous user groups and organizations. Distributed access control is still an open problem because solutions proposed so far do not provide the flexibility and manageability that are required by distributed objects systems. In order to solve this problem we propose a solution that makes use of *attribute certificates*, a standardized type of independent data object that can contain user privileges.

However, the use of attribute certificates and Privilege Management Infrastructures (PMIs) does not solve all the problems. Application-level access control is required in many distributed computing environments because some authorization decisions are based on application domain-specific factors. Fine-grained access control is also a common requirement. Approaches that embed authorization logic into the applications do not promote interoperability, introduce great administrative overhead on the security administrator and increase the chance of errors. Therefore, when considering security, scalability and interoperability requirements simultaneously, the most appropriate solution is to separate the certification of attributes from access control functionalities. In this way, the access control system will need the complement of an external component providing attribute certification functions. The full integration of an external access control facility, the *Resource Access Decision* (RAD) [2], and an infrastructure for the management of privileges, the PMI, represents a big step towards the solution of the general problem.

Other security challenges must be addressed in distributed systems. For instance, malicious components, end-to-end service requirements, evolving security technologies and security policy administration. The first three challenges have received some attention and a variety of solutions. Conversely, the last one has not been sufficiently addressed. In fact, it is difficult to correctly manage security policies in large complex systems with many objects. In order to address this problem, we have made an extensive use of semantic information about resources to be accessed, and have defined an expressive access control language that allows the modularisation, parameterisation and dynamic allocation of policies. Furthermore, a set of tools has been developed to help the administrator in all those tasks related to the creation, management and semantic and contextual validation of policies.

The semantic description of the external privilege infrastructure is an essential technique for its integration in the access control system, and metadata is the instrument used to convey this semantic information. There are ongoing standardization efforts in the field of metadata with the goal of promoting interoperability and efficiency. The Semantic Web Activity builds upon the earlier W3C Metadata Activity, including the definition of RDF [3] aimed to enable an XML based standardized and interoperable specification of semantics for resources. XML-Schema is the de-facto standard for expressing shared vocabularies and defining the structure, content and semantics of XML documents. XML-Schema introduces a powerful type mechanism that allows specifying primitive data types, and also types of structures. Both technologies, RDF and XML-Schema, are used in PMI-RAD for the representation of semantics.

Consequently, the approach presented in this paper, PMI-RAD, integrates the services of an external PMI into CORBA applications using the RAD facility. Our RAD implementation requests

and verifies attribute certificates from the PMI in a transparent way for CORBA objects. Additionally, our solution takes into account the enforceability of many access policies by many applications. An application access policy may control who can invoke the application, extending the object invocation access policy enforced by the *Object Request Broker* (ORB), and taking into account other items such as the value of the parameters or the data being accessed. PMI-RAD also addresses this problem while, at the same time, controlling access to finer-grained functions and data encapsulated within it.

The paper is organized as follows. Section 2 presents the most important previous results related to our research. Section 3 overviews CORBA security, focusing on the CORBA Security Service and the RAD facility. In section 4, attribute certificates and PMIs are studied in detail. Section 5 describes our approach, the PMI-RAD. We explain the system operation and the role of semantic information and discuss the most relevant advantages of the solution. Finally, section 6 summarizes the conclusions and presents ongoing and future work.

2 Related work

Some interesting research has been done in the area of access control services in CORBA. Karjoth has proposed a formal model of authorization, revealing some important deficiencies in the 1995 CORBA security standard [4]. A view-based model is presented in [5] for designing access policies. Additionally, the paper presents an alternative access control model for CORBA systems. Beznosov et al. [6] have addressed the problem of providing fine-grained application-level access control for security aware applications. The work originated the adoption of the RAD specification within the OMG standards. The difficulties of describing an appropriate notion of the security attributes “caller” and “target” in object-oriented middleware systems such as CORBA are discussed in [7]. This paper shows that a security service layer can not fully abstract the underlying security mechanisms without having implications on granularity and semantic mismatches.

Regarding PMIs, several proposals have been introduced for access control to distributed heterogeneous resources from multiple sources. The *Akenti* Project [8] proposes an access control system designed to address issues raised when allowing restricted access to distributed resources controlled by multiple stakeholders. The requirement for the stakeholders to trust the rest of the servers in the network, the assumption of the existence of a supporting identity *Public Key Infrastructure* (PKI) and some security vulnerabilities related to the existence of positive and negative use-conditions are the main drawbacks of Akenti. The *PERMIS* Research Project [9] objective is to set up an integrated infrastructure to solve identification and authorization problems. PERMIS group has examined various policy languages concluding that XML is the most appropriate candidate for a policy specification language. Because the PERMIS system is based on the *Role-Based Access Control* (RBAC) model [10] it shares its limitations. Also the requirement of a supporting PKI is hard to fulfil and it is not necessary in many authorization scenarios.

In the context of policy specification, several languages based on the XML standard have been developed for access control, digital rights management, authentication and authorization [11-15]. Many similarities and interesting features can be found among these languages. Nevertheless, they do not support relevant properties such as policy parameterisation and composition. Moreover, many features provided by those languages are not necessary in the scenarios we are addressing [16]. Two relevant proposals are the *Author-X* system [17] and the *FASTER* project [18], which describe two similar systems for access control to XML documents. Because both systems have been specifically developed for XML documents, they do not fit well in the CORBA environment.

However, they share some features with our solution, PMI-RAD. For instance, FASTER and our *Semantic Policy Language* (SPL) use XML-Schema [19], the W3C successor of *Data Type Definitions* (DTDs) [20]. Additionally, the scheme we have designed and developed uses a second XML metadata technology, RDF-Schema, the W3C standard for metadata interoperability [21]. This enables the dynamic allocation, composition and instantiation of SPL policies, based on metadata about the resources to be accessed.

3 Overview of CORBA Security

Although CORBA significantly simplifies the implementation of complex distributed systems, the support of techniques for reliable, fault-tolerant, and secure software is limited in the state-of-the-art CORBA. The CORBA specifications include several independent security related components. In this section we review these components paying special attention to those ones related to our work.

3.1 CORBA Security Service

The *CORBA Security Services Specification* (CORBASec) specifies a number of security functionality components [22]. It provides only a limited choice of coarse-grained mechanisms to specify access rights for components. The security service controls access to application objects not being aware of security, so security granularity is, in practice, limited to the object level.

The security functionality defined by this specification comprises:

- *Identification* and *authentication* of principals, human users and objects that need to operate under their own rights, to verify they are who they claim to be.
- *Authorization* and infrastructure based *access control* - deciding whether a principal can access an object domain, individual object, or operation on an object, normally using the identity and/or privilege attributes of the principal (such as role, groups, security clearance, etc.).
- *Security auditing* to make users accountable for their security related actions. Auditing mechanisms should be able to identify the user correctly, even after a chain of calls through many objects.
- *Security of communication* between objects, which is often done over insecure lower communication layers. Establishing trust between the client and target may require mutual authentication. It also may require integrity protection and, optionally, confidentiality protection of messages transferred between objects.
- *Non-repudiation* provides irrefutable evidence of actions such as proof of origin of data to the recipient, or proof of receipt of data to the sender to protect against subsequent attempts to falsely deny the reception or sending of data.

Some services are implemented by CORBA on the ORB layer, with the use of so-called interceptors, e.g. access control and audit. However, they rely strongly on the services provided by the underlying security technology, such as authentication and message protection. CORBASec acts to some extent like an API that calls underlying security mechanisms such as Kerberos v5 [23], and SESAME [24], instead of implementing all security functionality itself. Therefore, the functionality offered by CORBASec is always limited by the functionality offered by the underlying security mechanisms. *Secure Sockets Layer* (SSL) [25] is also widely used as a basic security mechanism for CORBA security, but it is not well integrated into the CORBA security architecture. The reason is that SSL works as a secure transport mechanism, that is, it creates a network connection as part of

the security context establishment. SSL has to be integrated as an alternative transport mechanism into the ORB. In this way, the security context is automatically set up when the ORB opens a new network connection.

3.2 Other CORBA Security Specifications

The *Authorization Token Acquisition Service* (ATLAS) [26] is used by a *Client Security Service* (CSS) to acquire authorization tokens to be delivered to a *Target Security Service* (TSS) for security purposes. This specification defines the method by which a client locates an ATLAS, but only to make requests on a specific target. The location of ATLAS is explicitly declared out of the scope of the specification.

The *Security Domain Membership Management* (SDMM) architecture defines interfaces that are used by different players for the interaction with SDMM mechanisms [27]. The architecture has two major objectives: (a) the first is to define standard mechanisms (interfaces) for obtaining information about the membership of a given object in groups, called security policy domains; and (b) the second is to define standard mechanisms for retrieving information about the state of objects.

The *Common Secure Interoperability Specification - version 2* (CSIv2) [28] defines the *Security Attribute Service* (SAS protocol) that enables authentication, delegation, and privilege data interoperability. This is achieved by acting as a higher-level protocol under which secure transports may be unified. The protocol provides client authentication, delegation, and privilege functionality that may be applied to overcome corresponding deficiencies in an underlying transport.

Although the *Naming Service* [29] is not a CORBA security specification we include it in this section because the way to use it is very relevant for the implementation of any security solution. The Naming Service provides the principal mechanism through which most clients of an ORB-based system locate objects that they intend to use. The *Uniform Resource Locator* (URL) scheme is used to represent a CORBA object bound in a *NamingContext*. Basically, there are three different ways to describe a CORBA Object Reference in string form: *IOR* addresses, *corbaloc* addresses, and *corbaname* addresses. For our purposes, the second one is the most convenient. A *corbaloc* address has the format `corbaloc:proto:hostname:port/objectID` and represents a CORBA Object Reference as a URL. The optional *proto* part denotes the transport protocol to be used defaulting to *Internet Inter-ORB Protocol* (IIOP); the *hostname* and *port* parts define the IP host and the TCP port where a suitable IIOP message must be sent in order to contact the CORBA object. The *objectID* part identifies a specific CORBA object.

The *Resource Access Decision* facility is a mechanism used to obtain authorization decisions and to manage access decision policies. It provides an enhanced functionality that is not provided by CORBASec. The practical result of not being able to use an infrastructure security service to express application specific policies is that each business application must contain its own implementation of an access control facility. This introduces severe problems for organizations that need to define access policies that are consistent across applications. Nowadays, these policies are often coded as part of the application, which makes impossible to administer and maintain auditable access policies. The OMG RAD addresses these problems providing a uniform way for application systems to enforce resource-oriented access control policies. The standardization of this service enables organizations to define and manage an *Enterprise Security Policy* for use by all their software components. The RAD service provides several functions, as the ability to use credentials supplied by diverse security mechanisms, and to integrate facilities delivered as a product by security providers. Additionally, it facilitates to consider multiple access control policies and to define how these policies are reconciled to govern access to the same resource.

4 Authorization using Attribute Certificates: Towards PMIs

The ITU-T (International Telecommunications Union) X.509 recommendation [30] standardizes the concept of attribute certificate[2], and defines a framework that provides the basis upon which a PMI can be built. As it is well known, a PKI supports encryption, integrity and non-repudiation services, and essentially focus on the management of *identity certificates* (a.k.a. *public-key certificates*). Precisely, the foundation of the PMI framework is the PKI framework defined by ITU [31]. In fact, attribute certificates have been designed to be used in conjunction with identity certificates, that is, PKI and PMI infrastructures are linked by information contained in the identity and attribute certificates. Although linked, both infrastructures can be autonomous, and managed independently, what provides a good advantage. Creation and maintenance of identities can be separated from PMI, as authorities that issue certificates in each of both infrastructures are not necessarily the same ones. In fact, the entire PKI may be existing and operational prior to the establishment of the PMI.

One of the main advantages of an attribute certificate is that it can be used for various purposes. It may contain group membership, role, clearance, or any other form of authorization. A very essential feature is that the attribute certificate provides the means to transport authorization information in distributed applications. This is especially relevant because through attribute certificates, authorization information becomes "mobile", which is highly convenient for CORBA. The mobility feature of attributes has been used in applications since the publication of the 1997 ITU-T X.509 recommendation. However, it has been used in a very inefficient way. That recommendation introduced an ill-defined concept of attribute certificate. For this reason, most of actual applications do not use specific attribute certificates to carry authorization information. On the contrary, attributes of entities are carried inside identity certificates. The *subjectDirectoryAttributes* extension field is used for this purpose. This field conveys any desired directory attribute values for the subject of the certificate, and is defined as follows:

```
subjectDirectoryAttributes EXTENSION ::= {
     SYNTAX  AttributesSyntax
     IDENTIFIED BY id-ce-subjectDirectoryAttributes }
AttributesSyntax ::= SEQUENCE SIZE (1..MAX) OF Attribute
```

This solution does not make entity attributes independent from identity, which can cause problems. Firstly, this is not convenient in the frequent situations where the authority issuing the identity certificate is not the authority for the assignment of privileges. Secondly, even in the situations where the authority is the same one, we must consider that life of identity certificates is relatively long when compared to the frequency of change of user privileges. Therefore, every time privileges change it would be necessary to revoke the identity certificate, and it is widely known that certificate revocation is a costly process. Moreover, many applications deal with authorization issues like delegation (conveyance of privilege from one entity that holds a privilege to another entity) or substitution (one user is temporarily substituted by another user, and this one holds the privileges of the first one for a certain period of time). Identity certificates do support neither delegation nor substitution.

The ITU-T X.509 recommendation of year 2000 provides the solution to these problems. Attribute certificates are conveniently described, including an extensibility mechanism and a set of specific extensions. A new type of authority for the assignment of privileges is also defined, the *Attribute Authority* (AA), while a special type of Authority, the *Source of Authority* (SOA), is settled as the root of delegation chains. The recommendation defines a framework that provides a foundation upon which a Privilege Management Infrastructure is built to contain a multiplicity of AAs and

[2] Although attribute certificates were introduced by ITU in its previous X.509 recommendation, the concept was originally ill-defined and its use has not been the appropriate one, as it is explained later.

final users. Revocation procedures are also considered by defining the concept of *Attribute Certificate Revocation Lists* (ACRLs), which are handled in the same way as for *Certificate Revocation Lists* (CRLs) published by *Certification Authorities* (CAs). The attribute and identity certificates of one user are bound as shown in Fig. 1. We can see that the field holder in the attribute certificate contains the serial number of the identity certificate.

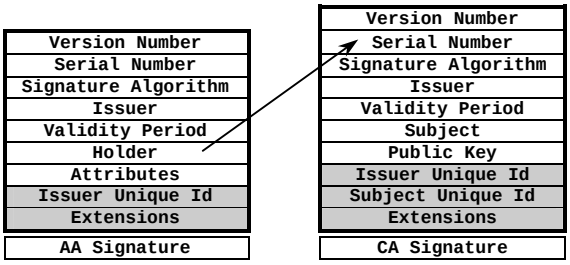

Fig. 1. Relation between attribute and identity certificate

Although linked, both certificates are independently managed. The important meaning of this is that a PKI and PMI are separate infrastructures in the sense that either structure can work on its own, or to be more precise, they can be established and managed independently. Next subsection describes in more detail this possibility.

4.1 Independence of the Infrastructures

The mutual independence of the two infrastructures is also valid when considering other ways to describe the holder of the attribute certificate. In spite of using the serial number of the identity certificate, or even additionally to it, it is possible to bind the attribute certificate to any object by using the hash value of that object. For instance, the hash value of the public key, or the hash value of the identity certificate itself. All possibilities for binding can be concluded from the ASN.1 [32] specification for the field holder shown in figure 2, where other related data structures are also specified. The content of this specification is essential for the solution that we have developed. We can see that one of the possible assignments to `GeneralName` is `uniformResourceIdentifier`, which precisely coincides with the use of *corbaloc* for the description of a CORBA object, as seen in previous section.

The infrastructures are absolutely separated when considering the situation in which some other authentication method different from that one based on identity certificates is used. In these cases, a PKI is not even used. The discussion about the separation of functions between PKIs and PMIs is a very relevant issue. From the point of view of the theory, the separation is possible, as we have argued in previous paragraphs. From the point of view of real application scenarios separation is not only possible but, in our opinion, very convenient. We previously argued that in most cases the authority issuing the identity certificate is not the authority for assigning privileges. That is, the entity having the role of CA is not the same one as that having the role of AA.

The main reason for this argument is that the identities have a global meaning in most certification schemes (although a few schemes do not support this idea, for instance, SPKI [33]). Thus, the CA does not necessarily belong to the organization where the entity belongs. The identity certificate can be issued by a CA managed by a governmental organization, a public organization, or even by a private company specialized in providing services and facilities related to certification of identities.

On the contrary, we believe that attributes have a non-global meaning. Moreover, because of the restricted scope of an attribute certificate, it is convenient that this certificate is issued by an authority that is more or less local to the scope of the user. This argument is even more valid if entity attributes are considered as confidential information. The certificate may contain some sensitive information and then attribute encryption may be needed, as proposed by PKIX [34].

```
Holder  ::= SEQUENCE {
  baseCertificateID [0] IssuerSerial OPTIONAL,    -- issuer and serial number of the identity certificate
  entityName        [1] GeneralNames  OPTIONAL,   -- the name of the entity or role
  objectDigestInfo  [2] ObjectDigestInfo  OPTIONAL -- used to directly authenticate the holder,
}
GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
GeneralName ::= CHOICE {
  otherName                 [0] INSTANCE OF OTHER-NAME,
  rfc822Name                [1] IA5String,
  dNSName                   [2] IA5String,
  x400Address               [3] ORAddress,
  directoryName             [4] Name,
  ediPartyName              [5] EDIPartyName,
  uniformResourceIdentifier [6] IA5String,
  iPAddress                 [7] OCTET STRING,
  registeredID              [8] OBJECT IDENTIFIER
}
ObjectDigestInfo  ::= SEQUENCE {
  digestedObjectType  ENUMERATED {
    publicKey       (0),
    publicKeyCert     (1),
    otherObjectTypes    (2)
  },
  otherObjectTypeID   OBJECT IDENTIFIER  OPTIONAL,
  digestAlgorithm   AlgorithmIdentifier,
  objectDigest      BIT STRING
}
```

Fig. 2. ASN.1 specification of *Holder* and related data structures

5 Description of the PMI-RAD

Our PMI-RAD is aimed to the integration of PMI services for CORBA security-aware applications. Consequently, our solution has been focused on the RAD facility. PMI-RAD is an open solution that adheres to the functionality of the standard specifications. Therefore, the necessary modifications of the original RAD have been carefully designed in order to guarantee transparency and interoperability. Our proposal relies on the semantic description of the PMI, using XML metadata standards for the full integration of this service into the CORBA security components and the interoperation of the PMI with other systems. In fact, we have successfully applied this approach to a Digital Library access control scenario. Therefore, the semantic description represents a valuable tool for the interfacing of different distributed systems. This solution adds enhanced functionalities for security administrators and represents a flexible, scalable and interoperable approach for the integration of two independent distributed systems.

5.1 Overview of the PMI-RAD

5.1.1 Analysis of the RAD specification

The RAD facility reference model defines a framework in which a wide variety of access control policies may be supported [2]. However, not all access control schemes can be represented in the RAD model. Some of its limitations are concerned with the `DecisionCombinator` object, the `SecuredResource` representation, and the policy evaluation scheme.

The `DecisionCombinator` object is used to combine the results produced by a series of `PolicyEvaluator` objects. The inclusion of the `DecisionCombinator` object and the one-to-

one cardinality of the relation between several `PolicyEvaluator` and `Policies` conflicts with the modular policy approach. That is, this design allows the combination of results but not the combination of different policies. On the other hand, both the definition of modular policies and their combination are important features that increase the flexibility and expressiveness of the access control specifications while reducing the complexity of management.

The representation of the list of `SecuredResource` is simple, and represents a negative aspect of the RAD reference model. A `SecuredResource` is represented within the RAD by a `ResourceName`, a structure containing an `AuthorityId` for the namespace and a sequence of name/value pairs. This sequence is intended to provide different ways to identify the resource but not to state properties about it. This fact is especially important because the RAD facility is used by security-aware applications. In these applications, domain-specific factors and particular properties of objects accessed must be considered when taking an access decision. In order to make a final access decision, the model takes into consideration properties (`SecAttributes`) about the client object. In contrast, it is not straightforward to include properties about the target object. Therefore, in the reference model, a different policy must be defined for each target object.

In practice, it is frequent that policies applicable to several objects are instances of a more general policy. For instance, consider this simple policy: "A client with security level *l* can have access to any object with security level equal or lower than *l*". Given two objects *a* and *b* with different security levels l_a and l_b we would need to specify two policies instead of the generic one: (i) P_a for object *a* stating "grant access to clients with security level grater or equal to l_a", and (ii) P_b for object *b* stating "grant access to clients with security level grater or equal to l_b". The reason is that no information is provided about the properties of the target object (`SecuredResource`). Moreover, although dynamic properties are claimed to be supported by the model, they are intended to represent the client, not the `SecuredResource`.

Finally, the policy evaluation scheme is based on a `PolicyEvaluatorLocator` that obtains object references to several `PolicyEvaluators` and the `DecisionCombinator` that are required for an access decision. Again, the `PolicyEvaluatorLocator` object does not consider any dynamic information about the target object. On the other hand, it is supposed to be able to find the set of required `PolicyEvaluator` on the basis of the target object `ResourceName`. As a consequence the location of the policy applicable to a given object is inflexible and static. Additionally, the specification does not address the association of policies to newly created objects.

5.1.2 Proposed approach

Previous analysis reveals some drawbacks of the RAD specification that can be solved by introducing some modifications that are transparent to the rest of the CORBA system. These modifications are based on meta-models representing properties about the group of `SecuredResource`**,** the `AttributeEvaluator` external component (in our approach, the PMI) and the applicability of the policies to those resources. Policy modularisation, parameterisation and composition are also at the core of our proposal. Fig. 3 shows the structure of the PMI-RAD model.

Essentially, this structure follows the RAD specification. In our scheme, the `DynamicAttributeService` is the ***`PMIClient`*** and the external `AttributeEvaluator` is the PMI. Our ***`GlobalPolicyEvaluator`*** component replaces three RAD components: `PolicyEvaluator`, `PolicyEvaluatorLocator` and `DecisionCombinator`. The `SecuredResource` representation is improved with the inclusion of metadata concerning the resource expressed in *Secured Resource Representation* (***`SRR`***). We also add semantic information in *Policy Applicability Specifications (**`PAS`**)*, used to relate policies to resources, and *SOA Descriptions*

(*SOADs*), which convey semantic information about the SOAs of the PMI. An external, but important component is the *PolicyAssistant* management tool. This takes into account information contained in the *SOADs* for the production and semantic validation of policies. We review now each component separately.

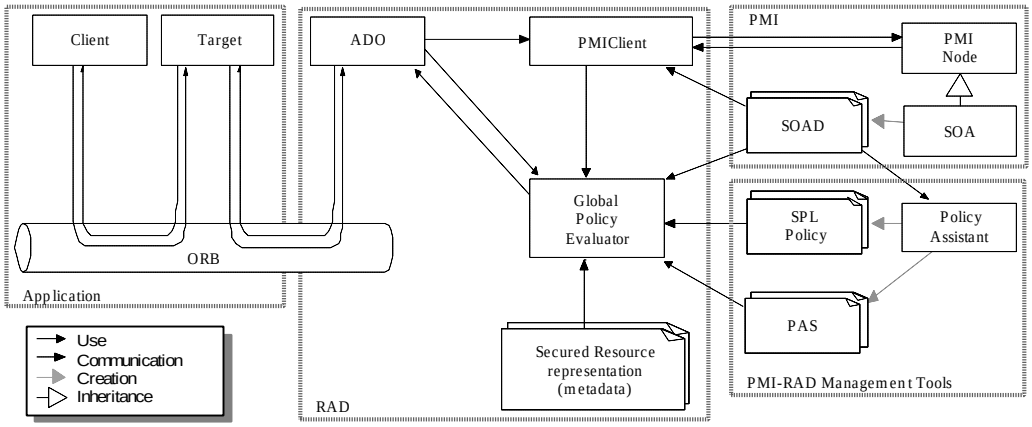

Fig. 3. PMI-RAD access control model.

The first of the components is the PMI. It is an essential component because it enables the separation of attribute certification and access control functionalities, which is widely accepted as the most scalable and flexible access control architecture. In CORBA systems, the integration of the standard external access control facility (the RAD) and a PMI represents a step towards the solution of the interoperability of different object sources with heterogeneous access control systems. It is important to note that the PMI component is intended to be common to other applications and external to this system. Metadata described about the PMI, represented as *SOADs*, is the key to achieve the necessary interoperability. Another component, the *PMIClient*, requests attribute certificates to the PMI and submits them to the *GlobalPolicyEvaluator*, replacing the original `DynamicAttributeService`. The access decision request from the client and the *SOADs* are used by this component to find the PMI entity to be contacted in order to obtain the corresponding attribute certificates.

Usually, each SOA in a PMI issues certificates for a small number of semantically related attributes. In our scheme, another semantic description mechanism, the *SOAD*, establishes trust between the security administrators and the PMI. *SOADs* are RDF statements protected by digital signatures [35]. They convey semantic information about the certificates issued by each SOA to assist the security administrators in the creation and semantic validation of access control policies. They are also used by the *GlobalPolicyEvaluator* for policy evaluation. These descriptions state a series of facts about the environment of the system using metadata about the different attributes that are certified by the SOA, including names, descriptions and relations among attributes. This semantic information allows the detection of possible inconsistencies in our SPL policies.

Although the RAD specification states that the policy administration components are out of its scope, we consider that the relevance of the *PolicyAssistant* component fully justifies its inclusion in our proposal. The main reason is that solutions that do not include automatic management tools are merely useful for scenarios and systems containing only a few objects, which is clearly not the goal of PMI-RAD. The *PolicyAssistant* uses the *SOADs* for the specification of policies and *PAS*. Additionally, the *PolicyAssistant* includes components for the automated

validation of policies at different levels, syntactically and semantically. Therefore, this component integrates all the tools to facilitate the administration of the access control system.

PAS are used to relate policies to resources. Conceptually, the *PAS* could be interpreted as part of the PolicyEvaluatorLocator. *PAS* are XML-Schema based, providing an expressive way to relate policies to resources by including conditions about the semantics of the target resources.

Policies are expressed using SPL, an XML-Schema based policy language designed to specify policies in a simple way, achieving a high level of expressiveness and an efficient evaluation. In order to facilitate the definition and management of policies, we have taken an approach based on the modular definition of policies that can be combined without ambiguity. Tools provided to support policy specification, composition and validation also serve this objective.

The basic description about SecuredResources is not adequate to be used in the process of dynamic allocation of policies to resources. Our *SRR* is designed specifically for this purpose. Dynamic allocation is a very flexible and useful mechanism that solves the problem of associating policies to newly created objects. The use of dynamic policy allocation needs a rich set of metadata about the resources. This semantic meta-model is used to locate the right policy for each resource, based on its relevant properties.

Finally, the *GlobalPolicyEvaluator* component incorporates the policy location and evaluation functions. This approach allows us to achieve more expressive ways of computing the access decision. Opposed to the simple combination of results, which is usually limited to either all (logical *and*) or any (logical *or*) operators, our approach enables the combination of policies. Some decisions cannot be expressed by a simple combination of results. The combination of policies represents a more general approach that helps reducing the complexity of administration while enabling more expressive and flexible policies to be considered in the access decision.

5.2 System operation

In PMI-RAD, an access decision is requested by a client by invoking the access_allowed() method of the AccessDecision object (ADO) passing a ResourceName, Operation, and a list of SecAttribute. In the original RAD specification every SecAttribute represents an attribute of the principal, where identity is considered as an attribute.

The SecAttribute definition is biased to the use of credentials. As a consequence, the use of attribute certificates within this structure is not straightforward. The main problem is that the structure is passed as a parameter of the access_allowed() method. The inclusion of the attribute certificates in the request would introduce efficiency and flexibility problems. An additional important issue is that attribute certificates are not only expected to be short-lived but to be revoked more frequently than identity certificates. For this reason, approaches that allow users to distribute their own certificates entail the use of ACRLs. Thus, the certificate validation procedure becomes more complicated and the overall system performance is reduced.

Although this invocation scheme is not optimal, we have left it unmodified in order to be interoperable and transparent to existing RAD clients. In PMI-RAD the SecAttribute is used to convey the certification information but not the certificate. The identity is included as a *corbaloc* address. Our PMI follows the *Cert'eM* approach to avoid the need for ACRLs [36]. The ADO consults the PMI through a *PMIClient* to obtain an updated list of SecAttributes including dynamic attributes. The ADO also consults the *GlobalPolicyEvaluator*. This component uses the *PAS* to obtain the applicable Policies. Moreover, it instantiates and combines all the

applicable policies using several metadata sources. Finally, it produces a final access decision that is returned to the client. The activity diagram of the PMI-RAD is depicted in Fig. 4.

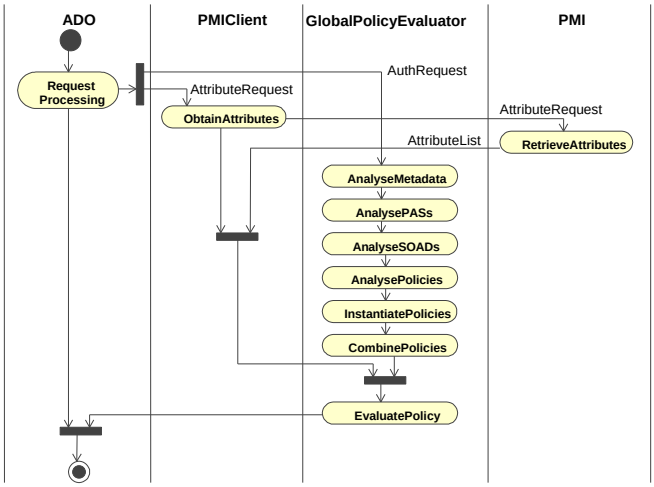

Fig. 4. Activity diagram of the PMI-RAD operation

After receiving an access request the `AccessDecisionObject` requests the attribute certificates from the *`PMIClient`* and forwards the authorization request to the *`GlobalPolicyEvaluator`*. Using the *`SOADs`* to determine which node of the PMI must be contacted, the *`PMIClient`* forwards the attribute certificates request to the appropriate node.

The *`GlobalPolicyEvaluator`* analyses the semantic metadata available for the target resource contained in the enhanced *`SRR`* along with other contextual metadata, finds the appropriate *`PAS`* and retrieves the necessary *`SOADs`*. Using this information, the *`GlobalPolicyEvaluator`* is able to find the applicable policies. All applicable policies are then analysed and instantiated using the metadata about the resource (*`SecuredResourceRepresentation`*) and the environment (*`SOAD`*). Finally, all policies are combined and evaluated and the access decision is returned to the ADO.

5.3 The role of the semantic information in PMI-RAD

The semantic description of the external PMI is an essential tool for its integration in the RAD component. The semantic information is described in our proposal through XML metadata technologies such as XML-Schema, RDF and RDF-Schema. In our proposal, metadata are applied at different levels. On one hand, access control policies benefit from metadata about the PMI for its creation and semantic and contextual validation. On the other hand, resources have associated metadata that are used for the dynamic policy allocation and parameter instantiation.

5.3.1 PMI-RAD Languages

Semantic Policy Language, SPL. PMI-RAD policies are expressed using SPL, an XML-Schema based policy definition language designed to specify policies in a simple way, enabling a high level of expressiveness and an efficient evaluation. SPL policies are modular and can be composed without ambiguity, which facilitates their definition and management. Tools provided to support the policy specification, composition and validation also serve this objective. The schema for SPL

policies, included in Appendix A, facilitates their creation, allowing automatic syntactic validation. SPL policies consist of a set of `access_Rule` elements. Each `access_Rule` defines a particular combination of attribute certificates required to gain access, associated with an optional set of `actions` (to be performed before access is granted). Examples of these actions are `Notify_To`, `Payment` and `Online_Permission`. A policy includes zero or more `parameter` declaration elements. References to a defined parameter can appear anywhere in the policy. The target object metadata and the `PAS` determine the instantiation of parameter references. `Import` elements can substitute some of the previous contents of the policy. `Import` elements allow the modular composition of policies based on the X-Path standard [37]. Section 5.4 includes an example of the high expressiveness of the SPL language. The XML descriptive tags make SPL policies understandable to administrators and XML-aware applications.

Policy Applicability Specification, PAS. The `PAS` specification provides an expressive way to relate policies to resources. The XML-Schema for `PAS` documents is included in Appendix A. `PAS` documents have a root `PAS` element that includes zero or more `parameter` declaration elements. References to a defined parameter can appear anywhere in the `PAS`. Parameter references will be instantiated based on the target object metadata. A set of one or more `policy` elements declare the policy or policies to be applied to objects declared in the `object` element. More than one `object` element can appear in the same `PAS`. Optionally, `operation` elements can be used to define which operations of the target object are controlled by the declared `policy`. In case no `operation` element is included, the `policy` is applicable to all of the object operations. The applicability is expressed as `conditions` to be fulfilled by these objects. The `instantiation` element describes the mechanism to instantiate parameters in the policies. An `instantiation` element must exist for every parameter defined in the policies. Fig. 5 shows an example of applicability rules for SPL policies to objects indicating that the `Right_Policy.xml` is applicable to all objects of type 'Register' in the 'Admin' folder.

Secured Resource Representation, SRR. The `SRR` is an enhanced version of the basic `SecuredResources` description of the original RAD. The `SRR` is a simple and powerful mechanism to describe properties about resources. Properties described in `SRRs` are used for the instantiation of policies and `PAS`, and to locate the applicable policies. The `SRR` schema is included in Appendix A. An example of an `SRR` is also included in Fig. 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<spl:PAS xmlns:spl="http://www.lcc.uma.es/CORBA"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lcc.uma.es/CORBA pasPMI_RAD.xsd">
  <spl:policy>file:///c:/CORBA//Right_Policy.xml</spl:policy>
    <spl:object> http://www.uma.es/Admin/ </spl:object>
    <spl:operations>
      <spl:operation>update</spl:operation>
    </spl:operations>
    <spl:conditions>
      <spl:condition predicate="equals">
        <spl:property_Name>object_Type</spl:property_Name>
        <spl:property_Value>Register</spl:property_Value>
      </spl:condition>
    </spl:conditions>
    <spl:instantation>
      <spl:formal_Parameter>Target</spl:formal_Parameter>
      <spl:actual_Parameter>subject_Code</spl:actual_Parameter>
    </spl:instantation>
</spl:PAS>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<spl:SRR xmlns:spl="http://www.lcc.uma.es/CORBA"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lcc.uma.es/CORBA
                    SRR_PMI_RAD.xsd"
resource="http://www.uma.es/Admin/Register_DB201_0207.obj">
  <spl:property predicate="equals">
    <spl:property_Name>object_Type</spl:property_Name>
    <spl:property_Value>Register</spl:property_Value>
  </spl:property>
  <spl:property>
    <spl:property_Name>subject_Code</spl:property_Name>
    <spl:property_Value>DB201</spl:property_Value>
  </spl:property>
  <spl:property>
    <spl:property_Name>examination_Session</spl:property_Name>
    <spl:property_Value>200207</spl:property_Value>
  </spl:property>
</spl:SRR>
```

Fig. 5. PAS for Right_Policy.xml (left) and SRR for Register_DB201_0207.obj (right)

Source Of Authorization Description, SOAD. These RDF documents express the different attributes certified by each SOA, including their names, descriptions and relations, enabling the detection of semantically

incomplete or incorrect policies. SOADs are digitally signed. The set of SOADs represents the semantic description of the PMI. Full integration of the PMI can be achieved transparently for the rest of the system thanks to this description.

5.3.2 Creation and Semantic and Contextual Validation of SPL Policies

The creation and maintenance of access control policies is a difficult and error prone activity. The *PolicyAssistant* component is designed to help administrators to specify those policies and validate them to find syntactic and semantic errors. For this purpose, the *PolicyAssistant* provides the administrator with the information about the attributes that can be included in the policies, their sources and relation. The automated validation of policies is performed at different levels. SPL policies are validated syntactically using XML-Schema. Semantic validation is made possible by the use of our *SemanticPolicyValidator*, included in the *PolicyAssistant* component. This Java component has been developed using the DOM API [38]. It parses SPL policies validating them with respect to the semantic information available through the different metadata defined. An interesting feature of the *PolicyAssistant* is that, based on the SOADs, it allows policies to be validated in the context where they will be applied.

5.4 An Example

To illustrate how our approach can achieve a level of expressiveness not offered by the actual RAD specification, we are going to use the context of an e-Learning scenario. In this example we will focus on the access to the Update operation of the official Registers of the different courses. In order to clarify our example we have included a partial class diagram of the e-Learning environment in Fig. 6. Clients (teachers and students) access CourseServers to perform various tasks. Clients are authenticated by their personal smart card.

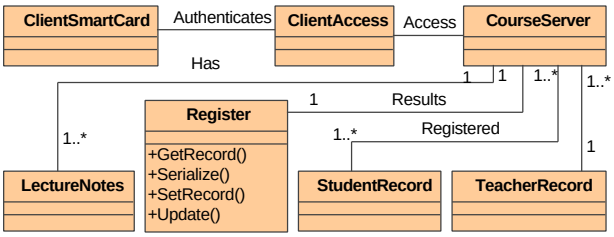

Fig. 6. Partial Class Diagram of an e-Learning system

In the PMI-RAD the SRR of the target resource is used to instantiate the PAS and the SPL policies. Additionally, during the validation of policies SOADs are used to determine the possible conditions under which access is granted. Lets suppose that the administrator wants to grant authorization to update the official course registers to teachers and deny it to students. Without the semantic information provided by the SOADs, the administrator might state the policy shown in Fig. 7. The error is motivated by the assumption that teachers and students are disjoint sets. But, in some institutions, it is possible for teachers to be registered as students. In such case, the previous policy would grant access to their own marks to students who are also teachers. An RDF statement declaring that ‘teachers may also be registered as students, except in the subjects they teach’ must be part of the corresponding SOAD. Additionally, the SOAD should state that ‘there is only one teacher for each course’ (see Fig. 6). This semantic information allows the context validation component to automatically detect possible inconsistencies in SPL policies.

In this example, the policy should have been stated as Fig. 7 shows. Additionally, this policy includes a validity interval, after which it has to be revised. Also, the `Public` attribute has been set to `false` in order to avoid that the client receives information about the terms of the policy during its evaluation. The corresponding `PAS`, shown in Fig. 5, states that the `subject_Code` value in the target object `SRR` must be used to instantiate the `Target` parameter of the policy. In this case, the instantiated policy only grants access to the teacher responsible for the `DB201` subject. `SOADs` are essential to check the soundness of policies. Additional attributes might be needed to make the access decision in case several teachers could be responsible for a subject.

```
<?xml version="1.0" encoding="UTF-8"?>
<spl:policy xmlns:spl="http://www.lcc.uma.es/CORBA"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lcc.uma.es/CORBA
PolicyTemplatePMI_RAD.xsd"
policy_Description="GRANT access IF Position='Professor' ">
  <spl:access_Rules>
    <spl:access_Rule>
      <spl:attribute_Set>
        <spl:attribute>
          <spl:attribute_Name>Position</spl:attribute_Name>
          <spl:attribute_Value>Professor</spl:attribute_Value>
          <spl:SOA_ID>LCC_ADM</spl:SOA_ID>
        </spl:attribute>
      </spl:attribute_Set>
    </spl:access_Rule>
  </spl:access_Rules>
</spl:policy>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<spl:policy xmlns:spl="http://www.lcc.uma.es/CORBA"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lcc.uma.es/CORBA
PolicyTemplatePMI_RAD.xsd"
policy_Description="GRANT access IF Position='Professor' ">
  <spl:access_Rules>
    <spl:access_Rule valid_From="2002-06-15T15:00:00"
     valid_Until="2002-09-30T24:00:00" Public="false">
      <spl:attribute_Set>
        <spl:attribute predicate="equals">
          <spl:attribute_Name>Position</spl:attribute_Name>
          <spl:attribute_Value>Professor</spl:attribute_Value>
          <spl:SOA_ID>LCC_ADM</spl:SOA_ID>
        </spl:attribute>
        <spl:attribute>
          <spl:attribute_Name>Teaches</spl:attribute_Name>
          <spl:attribute_Value>*Target</spl:attribute_Value>
          <spl:SOA_ID>LCC_ADM</spl:SOA_ID>
        </spl:attribute>
      </spl:attribute_Set>
    </spl:access_Rule>
  </spl:access_Rules>
</spl:policy>
```

Fig. 7. An incorrect policy (left) and the corresponding revised policy (right).

6 Discussion and Conclusions

We have concluded that an appropriate authorization approach that considers security, scalability and interoperability requirements simultaneously must separate the certification of attributes from access control functionalities. In this way, the access control system needs the complement of an external component providing attribute certification functions. The last ITU-T X.509 recommendation standardizes the concept of attribute certificate, and defines a framework that provides the basis upon which an infrastructure for privilege management, the PMI, can be built. Our work is based on the original idea of providing PMI services for CORBA applications. However, in the initial phase of the project we have realized that the use of attribute certificates and PMIs still left some problems opened. Hence application-level access control is required in many CORBA scenarios because some authorization decisions are based on application domain-specific factors. Also, fine-grained access control is a common requirement.

One of the most important contributions of this work is the full integration of the services of an external PMI in CORBA systems using the RAD facility. Our RAD implementation requests and verifies attribute certificates from the PMI in a transparent way for CORBA objects. This way of using the PMI makes the global system more scalable. Additionally, our solution takes into account the enforceability of many access policies by different applications. While addressing this problem, PMI-RAD controls access to finer-grained functions and data encapsulated within it. The higher expressiveness of SPL policies along with the additional semantic information allows us to enhance the functionality of the original RAD component. Our approach for the full integration of the PMI and the RAD facility is based on the semantic description of the PMI services. This approach can

easily be applied to other middleware systems. The core ideas are also valuable in other distributed systems such as Digital Libraries, Web Services or Grid Computing.

New scenarios of CORBA applications need external authorization mechanisms because resources can be accessed by previously unknown users. Authorization in distributed systems often relies on centralized access control management, what introduces important disadvantages in terms of efficiency, manageability and security. Although distributed authorization is positioned as a better alternative, solutions proposed so far do not provide the flexibility and manageability required by distributed objects systems. Scalability is also a crucial requirement. PMI-RAD is prepared to operate in scenarios where the number of users, resources, attributes and policies are very large and where access by previously unregistered users must be supported.

We have also addressed the problem of the administration of security policies, which is a difficult task in every large complex system with many objects. The extensive use of semantic information about resources to be accessed, and the definition of an expressive access control language are the basis of our solution. Furthermore, a set of tools has been developed to help the administrator in all those tasks related to the creation, management and semantic and contextual validation of policies. There is no doubt that the combination of a highly expressive language and the use of attribute certificates for access control enables the modular definition and dynamic allocation of policies to resources. On the other hand, the flexibility of modular, combinable and parameterised policies facilitates access control management. Within our approach administrators are able to manage the resources they control regardless of the resource location.

In policy specification languages, the existence of positive and negative authorizations is usually justified because they facilitate the definition of simple policies and reduce the number of policies applicable to each object. On the other hand, they introduce ambiguities and require the definition of conflict resolution rules. Furthermore, the administration of the system becomes complex and difficult to understand, increasing the chance of producing incorrect policies. To facilitate the definition and management of policies, we have taken a different approach based on the modular definition of policies that can be composed without ambiguity. Additionally, the semantic integration of a PMI permits us to consider access control policies that are not expressible with traditional access control schemes.

Regarding the ongoing and future work, after analysing the results of the metadata model that we have developed we have started to use the model for other parts of the system. At the same time, we are integrating other services into the PMI. That is, in this work we have basically used the PMI for access control purposes in the authorization scope. However, authorization is a broader concept that involves, for instance, delegation and substitution services. It is our intention to use our system in other scenarios, like those ones related to web services and smart card-based electronic commerce.

References

1. Object Management Group, *The Common Object Request Broker: Architecture and Specification*. 2002.
2. Object Management Group, *Resource Access Decision Facility Specification – Version 1.0*. 2001.
3. World Wide Web Consortium, *Resource Description Framework (RDF)*. 1999. http://www.w3.org/RDF/
4. Karjoth, G., *Authorization in CORBA Security*, European Symposium on Research in Computer Security – ESORICS'98, LNCS 1485, Springer. 1998.
5. Brose,G., *A View-Based Access Control Model for CORBA*, Secure Internet Programming: Security Issues for Mobile and Distributed Objects, LNCS 1603, Springer. 1999.
6. Beznosov, K., Deng, Y., Blakley, B., Burt, C., Barkley, J., *Resource Access Decision Service for CORBA-Based Distributed Systems*, 15th Annual Computer Security Applications Conference. 1999.

7. Lang, U., Gollmann, D. Schreine, R., *Verifiable Identifiers in Middleware Security*, 17th Annual Computer Security Applications Conference. 2001.
8. Thompson, M., et al., *Certificate-based Access Control for Widely Distributed Resources*, Eighth USENIX Security Symposium. 1999.
9. Chadwick, D. W., *An X.509 Role-based Privilege Management Infrastructure*, Global Infosecurity. 2002.
10. Sandhu, R.S., E.J. Coyne, H.L. Feinstein and Youman, C.E., *Role-Based Access Control Models*, IEEE Computer. 29(2): pp. 38-47. 1996.
11. ContentGuard, Inc., *eXtensible Rights Markup Language, XrML 2.0*. 2001. http://www.xrml.org
12. Open Digital Rights Language Initiative, *Open Digital Rights Language*. 2001. http://odrl.net/
13. Organization for the Advancement of Structured Information Standards, *Security Assertion Markup Language*. 2002. http://www.oasis-open.org/committees/security/
14. Kudo, M., Hada, S., *XML Document Security Based on Provisional Authorisation*, 7th ACM Conference on Computer and Communications Security. 2000.
15. Organization for the Advancement of Structured Information Standards, *eXtensible Access Control Markup Language*. 2002. http://www.oasis-open.org/committees/xacml/
16. Yagüe, M.I., Troya, J.M., *On the Suitability of Existing Access Control and DRM Languages for Mobile Policies*. University of Málaga. Department of Computer Science Tech. Rep. nb. LCC-ITI-2002-7. 2002.
17. Bertino, B., Castano, S., Ferrari, E., *Securing XML Documents with Author-X*, IEEE Internet Computing, 5 (3): 21-1. 2001.
18. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P., *A Fine-Grained Access Control System for XML Documents*, ACM Transactions on Information and Systems Security, vol. 5, n. 2, pp. 169-202. 2002.
19. World Wide Web Consortium, *XML-Schema*. 2001. http://www.w3.org/XML/Schema
20. World Wide Web Consortium, *Guide to the W3C XML Specification ("XMLspec") DTD, Version 2.1*. 1998. http://www.w3.org/XML/1998/06/xmlspec-report.htm
21. World Wide Web Consortium, *RDF Vocabulary Description Language 1.0: RDF Schema*. 2002. http://www.w3.org/TR/rdf-schema/
22. Object Management Group, *Security Service Specification - Version 1.8*. 2002.
23. Kohl, J., Neuman, B., *The Kerberos Network Authentication Service (V5)*, Internet RFC 1510. 1993.
24. McMahon, P., *Sesame V2 Public Key and Authorization Extensions to Kerberos*, Symposium on Network and Distributed Systems Security. 1995.
25. Netscape Communications, *SSL 3.0 Specification*. 1997.
26. Object Management Group, *The Authorization Token Layer Acquisition Service Specification*. 2001.
27. Object Management Group, *Security Domain Membership Management Service*. 2000.
28. Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Chapter 26: Secure Interoperability. 2002.
29. Object Management Group, *Naming Service Specification*. 2001.
30. ITU-T Recommendation X.509, *Information Technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks*. 2000.
31. ITU-T Recommendation X.509, *Information Technology – Open systems interconnection – The Directory: Authentication Framework*. 1997.
32. Kaliski, B., *A Layman's Guide to a Subset of ASN.1, BER, and DER*. 1993.
33. Ellison, C. et al., *SPKI Certificate Theory*, RFC 2693, IETF SPKI Working Group. 1999.
34. Farrell, S., Housley, R., *An Internet Attribute Certificate Profile for Authorization*, RFC 3281, IETF PKIX Working Group. 2002.
35. World Wide Web Consortium, *XML-Signature Syntax and Processing*. 2002. http://www.w3.org/TR/xmldsig-core/
36. López, J., Maña, A., Ortega, J., Troya, J. M., *Cert'eM: Certification System Based on Electronic Mail Service Structure*, Secure Networking – CQRE'99, LNCS 1740, Springer. 1999.
37. World Wide Web Consortium, *XML Path Language (XPath)*. 1999. http://www.w3.org/TR/xpath/
38. World Wide Web Consortium, *Document Object Model (DOM) Level 1 Specification*. 1998. http://www.w3.org/TR/REC-DOM-Level-1/

Appendix A. Schemas.

PolicyTemplate.xsd

```
<?xml version="1.0"?>
<xsd:schema   targetNamespace="http://www.lcc.uma.es/CORBA"   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:spl="http://www.lcc.uma.es/CORBA" elementFormDefault="qualified">
  <xsd:simpleType name="policy_ID_type">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[A-Z]{3}-[0-9]{3}"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="Actions_Type">
    <xsd:all>
      <xsd:element name="Notify_To" type="xsd:anyURI" minOccurs="0"/>
      <xsd:element name="Online_Permission_From" type="xsd:anyURI" minOccurs="0"/>
      <xsd:element name="Payment_Value" type="xsd:float" minOccurs="0"/>
      <xsd:element ref="spl:import" minOccurs="0"/>
    </xsd:all>
  </xsd:complexType>
  <xsd:simpleType name="predicate_Type">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="equals"/>
      <xsd:enumeration value="greaterOrEqual"/>
      <xsd:enumeration value="lessOrEqual"/>
      <xsd:enumeration value="greather"/>
      <xsd:enumeration value="less"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:element name="policy">
    <xsd:complexType>
      <xsd:choice>
        <xsd:sequence>
          <xsd:element ref="spl:import" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element name="parameter" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element ref="spl:access_Rules"/>
        </xsd:sequence>
        <xsd:element ref="spl:import" maxOccurs="unbounded"/>
      </xsd:choice>
      <xsd:attribute name="policy_Description" type="xsd:string" use="optional"/>
      <xsd:attribute name="policy_ID" type="spl:policy_ID_type" use="optional"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="access_Rules">
    <xsd:complexType>
      <xsd:choice>
        <xsd:sequence>
          <xsd:element ref="spl:import" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element ref="spl:access_Rule" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:element ref="spl:import" maxOccurs="unbounded"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="access_Rule">
    <xsd:complexType>
      <xsd:choice maxOccurs="unbounded">
        <xsd:sequence>
          <xsd:element ref="spl:import" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element ref="spl:attribute_Set" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:element ref="spl:import" maxOccurs="unbounded"/>
      </xsd:choice>
      <xsd:attribute name="Name" type="xsd:ID" use="optional"/>
      <xsd:attribute name="Public" type="xsd:boolean" use="optional" default="true"/>
      <xsd:attribute name="valid_From" type="xsd:dateTime" use="optional"/>
      <xsd:attribute name="valid_Until" type="xsd:dateTime" use="optional"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="attribute_Set">
    <xsd:complexType>
      <xsd:choice maxOccurs="unbounded">
        <xsd:sequence>
          <xsd:element ref="spl:import" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element ref="spl:attribute" maxOccurs="unbounded"/>
          <xsd:element ref="spl:import" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element name="action" type="spl:Actions_Type" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:element ref="spl:import" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="attribute">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:choice>
          <xsd:element name="attribute_Name"/>
          <xsd:element ref="spl:import"/>
```

```
        </xsd:choice>
        <xsd:choice>
          <xsd:element name="attribute_Value"/>
          <xsd:element ref="spl:import"/>
        </xsd:choice>
        <xsd:choice>
          <xsd:element name="SOA_ID" type="xsd:string" nillable="false"/>
          <xsd:element ref="spl:import"/>
        </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="attributeID" type="xsd:ID" use="optional"/>
      <xsd:attribute name="attributeDescription" type="xsd:string" use="optional"/>
      <xsd:attribute name="predicate" type="spl:predicate_Type" use="optional" default="equals"/>
      <!-- Each (attribute_Name, attribute_Value) is certified by a signer - SOA_ID - -->
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="attribute_Name">
    <xsd:complexType>
      <xsd:choice>
        <xsd:sequence/>
        <xsd:sequence/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="import">
    <xsd:complexType>
      <xsd:attribute name="Url" type="xsd:string" use="required"/>
      <xsd:attribute name="XPath" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

SRRTemplate.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.lcc.uma.es/CORBA"
 xmlns:spl="http://www.lcc.uma.es/CORBA"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:simpleType name="predicate_Type">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="equals"/>
      <xsd:enumeration value="greaterOrEqual"/>
      <xsd:enumeration value="lessOrEqual"/>
      <xsd:enumeration value="greather"/>
      <xsd:enumeration value="less"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="SRR">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="spl:property" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="resource" type="xsd:anyURI" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="property">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="property_Name" type="xsd:string"/>
        <xsd:element name="property_Value" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="predicate" type="spl:predicate_Type" use="optional" default="equals"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

PASTemplate.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- A PAS states that 'policy' is applicable to the 'operations' of 'object' fullfiling 'conditions' -->
<!-- 'instantiation' defines the substitutions to be applied to the parameters appearing in 'policy' -->
<xsd:schema      targetNamespace="http://www.lcc.uma.es/CORBA"      xmlns:spl="http://www.lcc.uma.es/CORBA"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:simpleType name="predicate_Type">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="equals"/>
      <xsd:enumeration value="greaterOrEqual"/>
      <xsd:enumeration value="lessOrEqual"/>
      <xsd:enumeration value="greather"/>
      <xsd:enumeration value="less"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="parameter" type="xsd:string"/>
  <xsd:element name="policy" type="xsd:anyURI"/>
  <xsd:element name="object" type="xsd:anyURI"/>
  <xsd:element name="predicate" type="spl:predicate_Type"/>
  <xsd:element name="property_Name" type="xsd:string"/>
  <xsd:element name="property_Value" type="xsd:string"/>
  <xsd:element name="PAS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="spl:parameter" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="spl:policy"/>
        <xsd:element ref="spl:object"/>
        <xsd:element ref="spl:operations" minOccurs="0"/>
        <xsd:element ref="spl:conditions" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="spl:instantation" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="actual_Parameter" type="xsd:string"/>
  <xsd:element name="operations">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="operation" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="condition">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="spl:property_Name"/>
        <xsd:element ref="spl:property_Value"/>
      </xsd:sequence>
      <xsd:attribute name="predicate" type="spl:predicate_Type" use="optional" default="equals"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="conditions">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="spl:condition"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="formal_Parameter" type="xsd:string"/>
  <xsd:element name="instantation">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="spl:formal_Parameter"/>
        <xsd:element ref="spl:actual_Parameter"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```