A Secure Solution for Commercial Digital Libraries*

Mariemma I. Yagüe, Antonio Maña, Javier López, Ernesto Pimentel, José M. Troya

Computer Science Department
University of Málaga. Spain.
{yague, amg, jlm, ernesto, troya}@lcc.uma.es

Abstract. Distributed systems usually contain objects with heterogeneous security requirements that pose important challenges on the underlying security mechanisms and especially in access control systems. Access control in distributed systems often relies on centralized security administration. Existing solutions for distributed access control do not provide the flexibility and manageability required. This paper presents the XML-based Secure Content Distribution (XSCD) infrastructure, which is based on the production of protected software objects that convey contents (software or data) and can be distributed without further security measures because they embed the access control enforcement mechanism. It also provides means for integrating Privilege Management Infrastructures (PMIs). Semantic information is used in the dynamic instantiation and semantic validation of policies. XSCD is scalable, facilitates the administration of the access control system, guarantees the secure distribution of the contents, enables semantic integration and interoperability of heterogeneous sources, provides persistent protection and allows actions (such as payment) to be bound to the access to objects.

Keywords: Distributed systems security, secure content distribution, XML metadata, Privilege Management Infrastructure.

* Work partially supported by Spanish Ministry of Science and Technology. Project TIC2002-04500-C02-02

1 Introduction

Digital Libraries (DLs) integrate a variety of information technologies that provide opportunities to assemble, organize and access large volumes of information from multiple repositories. Because of the high quality and value of the contents, it is becoming frequent that access to the DL assets (not only the contents but also services such as searching and abstracting) is subject to the payment of a fee. DLs usually contain objects with heterogeneous security requirements. Securing the access to DLs poses important challenges because of the specific characteristics found in DL environments. Some of these characteristics are the following:

(i) usually, libraries are offered to previously unknown users;

(ii) payment or other activities (e.g. the execution of copyright agreements) must be bound to the access to the objects;

(iii) the originator or owner of the object must retain control over it regardless of the storage location and even after it is accessed by users (this is known as persistent protection);

(iv) a high degree of flexibility is required because of the heterogeneous nature of the objects;

(v) the ability to change the access control parameters dynamically and transparently is also essential in most DLs; and finally

(vi) due to the large amount of objects, it is important to determine access conditions automatically, based on information about objects.

Although those problems for controlling access in distributed systems can be seen as general, it is clear that new solutions are required to address the needs of some of the new distributed applications, as it is the case of DLs, but also of web services or grid computing. Among the problems found on existing access control systems, we underline the following ones:

A. Security administration is very complex and error prone. A flexible and powerful policy language that incorporates features to manage the complexity inherent to certain environments, like DLs, represents a step towards the solution of this problem. Automated management tools also serve this objective.

B. The explicit static allocation of policies to objects is not adequate in highly dynamic environments with heterogeneous contents, where new resources are often incorporated to the system and security requirements change frequently. In order to solve this problem, dynamic

allocation of policies to resources and the definition of an access policy language designed to ease the management of the system must be considered.

C. The access control criteria are usually defined either explicitly or on the basis of the structure of the contents. These approaches present severe drawbacks, as we will show in this paper.

D. In new environments we deal with a large number of (possibly anonymous) users. Existing schemes, based on user identity, need to collect some profile information in advance. Therefore, a registration phase is used to collect information about the user and issue the corresponding local credentials each time a new DL is visited. The semantic integration of an external *Privilege Management Infrastructure* (PMI) represents a step towards the solution of the interoperability of different DLs with heterogeneous access control systems.

E. The access policy depends on the administrator of the server where the object is stored. It would be desirable that originators of the contents are able to define the applicable access policy to apply in a dynamic and transparent way, regardless of the object storage location.

F. Finally, no secure content distribution mechanisms are used.

This paper presents XSCD-DL, *XML-based Secure Content Distribution for Digital Libraries*, a particular application of the XSCD infrastructure (López *et al*., 2002b). XSCD represents a flexible solution for different distributed scenarios and any kind of content. It provides persistent protection and can be applied regardless of the attribute certification scheme. Besides, it offers distributed access control management and enforcement mechanisms and allows dynamic modification of policies transparently and efficiently. XSCD-DL, in addition to XSCD, provides secure content distribution and payment in commercial DLs. To achieve the aforementioned goals we combine an external authorization infrastructure, a modular language, called *Semantic Policy Language* (SPL), to specify how access to the contents is controlled and a software protection mechanism (*SmartProt*). The extensive use of semantic information makes possible the integration of such a complex set of subsystems into XSCD-DL.

The paper is organized as follows. Section 2 summarizes some related work. Section 3 describes the fundamentals of access control and the SPL language. Section 4 describes the content protection and payment mechanisms. Section 5 shows the global structure of XSCD-DL. Section 6 illustrates the advantages of SPL via an example. Finally, section 7 summarizes the conclusions and presents ongoing and future work.

2 Related work

Different projects, such as the *ADEPT Digital Library* (Janée and Frew, 2002) or the *Alexandria Digital Library* (Coleman, 2002), have focused on handling various structural and semantic issues, while providing users with a coherent view of a massive amount of information. The use of metadata is essential in these systems, but the application to the security issues is not considered. The *Stanford Digital Library Project* (Baldonado *et al*., 1997) covers most of the different issues involved in this field. One important outcome is the *FIRM architecture* (Ketchpel *et al*., 1996) that proposes the separation of objects that implement control from objects that are controlled, enhancing the flexibility of the system.

Regarding access control, several proposals have been introduced for distributed heterogeneous resources from multiple sources (Thompson *et al*., 2003; see also Chadwick, 2002). Unfortunately, these proposals do not address the specific problems of distributed access control in DLs. Traditional access control schemes such as *Mandatory Access Control* (MAC), *Discretionary Access Control* (DAC) or even *Role Based Access Control* (RBAC) are not appropriate for complex distributed systems such as DLs. It has been shown that an approach based on attribute certificates represents a more general solution that fits more naturally in these scenarios (López *et al*., 2002a). In fact, MAC, DAC and RBAC schemes can be specified using the attribute-based approach.

Because of the specific requirements imposed to the access control systems of DLs, the most widespread architecture is that of a federated set of sources, each one with a centralized access control enforcement point. This architecture has important drawbacks such as the reduced system performance produced because the centralized access control enforcement point becomes a bottleneck for request handling. Other drawbacks are: (a) the control point represents a weak spot for security attacks and fault tolerance; (b) it does not facilitate the deployment of persistent protection mechanisms; and, (c) it usually enforces homogeneous access control schemes that do not fit naturally in heterogeneous user groups and organizations.

On the other hand, distributed access control solutions proposed so far do not provide the flexibility and manageability required. An interesting approach based on the concept of mobile policies (Fayad and Jajodia, 2001) has been proposed to solve some of the limitations of RBAC schemes (McCollum *et al*., 1990). This is a limited improvement because of the requirement to execute the access control policies in trusted computers (object servers in this case). Furthermore, when access to an object is granted, this object has to be sent to the client computer where control over it is lost. Finally, because object and policy are compiled in a package, any single change in the policy that controls an object requires that the object-policy package is recompiled and distributed to all trusted servers.

Several XML based languages have been developed for access control, digital rights management, authentication and authorization. These languages do not support powerful features such as policy

modularisation, parameterisation and composition. Furthermore, some of their features are not necessary in DLs (Yagüe, 2002). Two relevant proposals are the *Author-X* system (Bertino *et al*., 2001) and the *FASTER* project (Damiani *et al*., 2001; see also Damiani *et al*., 2002), which propose two similar systems for access control to XML documents. Both systems define hierarchic access control schemes based on the structure of the document. The FASTER system does not support any content protection mechanism. FASTER access control is based on user groups and physical locations following the well-known technique of defining a subject hierarchy. In scenarios such as digital libraries, this approach is not adequate because a fixed hierarchy can not represent the security requirements of all the different contents, users and access criteria. On the other hand, content protection in Author-X is founded on the concept of (passive) secure container, which introduces disadvantages from the point of view of security and access control system management. Author-X is based on credentials that are issued by the access control administrator. Therefore, in practice, each credential will be useful only for a single source, limiting interoperability. A direct consequence of this approach is that users must subscribe to sources before they can access their contents.

3 Elaborating on Access Control

3.1 General Access Control Issues

Most of current access control schemes base their authorization approaches on locally-issued credentials that are based on user identities. This type of credentials presents many drawbacks. Among them we highlight: (a) they are not interoperable; (b) the same credentials are issued many times for each user, what introduces management and inconsistency problems; (c) credentials are issued by the DL administrator, however, in most cases, the administrator does not have enough information or resources to establish trustworthy credentials; and (d) they are dependent on user identity. However, in practice, it is frequent that the identity of the user is not relevant for the access decision. Sometimes it is even desirable that the identity is not considered or revealed. Furthermore, in systems based on identity, the lack of a global authentication infrastructure (a global *Public Key Infrastructure*, PKI) forces the use of local authentication schemes. In these cases, subscription is required and users have to authenticate themselves to every accessed source. To solve the aforementioned problems, single-sign-on mechanisms are becoming popular. These mechanisms are based on federation of sources that represent a limited improvement because credentials remain local (not to a site, but to a set of them). Moreover, all federated sources must agree on a homogeneous access control scheme.

On the other hand, digital certificates can securely convey authorizations or credentials. Attribute certificates bind attributes to keys providing means for the deployment of scalable access control systems in the scenarios that we have depicted. These authorizations are interoperable and represent a general and trustworthy solution that can be shared by different systems. Taking into account security, scalability and interoperability, the separation of the certification of attributes and access control management responsibilities is widely accepted as a scalable and flexible solution. In this case, the access control system

needs to be complemented by an external component: the PMI (ITU, 2000). The main entities of a PMI, known as *Source of Authorizations* (SOAs), issue attribute certificates. Usually, each SOA certifies a small number of semantically related attributes. This scheme scales well in the number of users and also in the number of different factors (attributes) used by the access control system. The flexibility of this model is such that it can easily simulate other models such as MAC, DAC or RBAC. Furthermore it can represent complex access conditions that are very difficult or impossible to express in other models.

With this approach, each access control system selects the SOAs to trust and which combination of attributes to use. Because they are separate systems, a mechanism to establish the trust between the access control and the PMI is required. The use of metadata to describe the PMI through *Source Of Authorization Description* (SOAD) documents is the key to achieve the necessary interoperability. SOADs are XML documents protected by digital signatures that express the semantics of the different attributes certified by each SOA. These descriptions state a series of facts about the environment of the access control system using metadata to represent the semantics of the different attributes that are certified by the SOA, including names, descriptions and relations among attributes. The semantic information about the certificates issued by each SOA is also used to assist the security administrators in the creation of access control policies. Additionally this semantic information allows the detection of possible inconsistencies in our SPL policies, during the semantic validation process.

When discussing how to establish the access conditions applicable to a particular resource, two main approaches must be considered: (i) conditions are established on the basis of the location of the resources or, (ii) conditions are based on the properties of the resources. The fact is that conditions and restrictions of access depend naturally on the semantic properties of the target resource that are neglected in structure-based approaches. Therefore, an approach based on semantic descriptions of the contents is much more flexible and natural. Moreover, it is easy to incorporate structure-based requirements in the semantic model. Additionally, the structure is much more volatile than the semantics. The incompatibility between the structure required for the application domain and the ones that match the security requirements confirms that structure-based approaches are not able to represent these situations in a natural way.

Another drawback of structure-based approaches is that the number of policies becomes very large. In fact, these approaches usually imply the definition of several policies for each resource. Positive and negative authorizations are used in these cases to facilitate the definition of simple policies and to reduce the number of policies. The price to pay is the introduction of ambiguities, which in turn requires the definition of conflict resolution rules. Consequently, the administration of the system becomes complex and difficult to understand increasing the chance of producing incorrect policies.

The semantic-based and modular approach adopted in XSCD-DL facilitates the definition and management of policies avoiding the use of positive and negative authorizations. Tools provided to support the policy specification, composition and validation also serve this objective.

3.2 A Modular Language for Secure and Flexible Administration

XML is widely considered the best candidate for the definition of a policy specification language. Existing XML-based languages for access control, authorization and digital rights management are not based on a modular approach and do not provide some important features such as policy composition and parameterisation. These features play an essential role in the flexibility of management of the access control system (Yagüe, 2002).

The definition of access control policies is a complex and error prone activity that presents many similarities with computer programming. Therefore, we have included in the design of our language some of the mechanisms used to reduce the complexity in programming languages such as modularity, parameterisation and abstraction. In order to provide the simplicity and flexibility required in complex systems such as digital libraries, our solution is based on the modular definition of policies. Modularity in our solution implies: (a) the separation of specification in three parts; that is, access control criteria, allocation of policies to resources and semantic information (properties about resources and context); (b) the abstraction of access control components; (c) the ability to reuse these access control components; and (d) the reduction of the complexity of management due to previous properties. Moreover, the use of semantic information about the context allows the administrator to include contextual considerations in a transparent manner, also helping the semantic validation task.

Usual components of access policies include the target resource, the conditions under which access is granted/denied and, sometimes, access restrictions. Opposed to other languages, SPL policy specifications do not include references to the target object. Instead, a separate specification called *Policy Applicability Specification* (PAS) is used to relate policies to objects dynamically when a request is received. Both SPL policies and PAS use semantic information about resources included in *Secured Resource Representation* (SRRs) and other contextual information documents. SPL policies and PAS can be parameterised allowing the definition of flexible and general policies and reducing the number of different policies to manage. Parameters, which can refer to complex XML elements, are instantiated dynamically from semantic and contextual information. Finally, policies can be composed importing components from other policies without ambiguity. This compositional approach allows us to define the meaning of the elements of the policies, providing a mechanism to achieve abstraction, which also helps in reducing the complexity of management. Tools to graphically manage the relations among policies and with other components are also essential for a simple and flexible management.

The schema for SPL specifications is represented as a set of XML-Schema templates that facilitate the creation of these specifications, allowing their automatic syntactic validation. Figure 1 shows the structure of the SPL language.

SPL policies can include locally defined components as well as imported elements. The ability to import elements enables the modular composition of policies based on the XPath standard. An SPL Policy is

composed of a set of *access_Rule* elements, each one defining a particular combination of attribute certificates required to gain access, associated with an optional set of actions (such as *Notify_To*, *Payment* and *Online_Permission*) to be performed before access is granted. In this way provisional authorization (Kudo and Hada, 2000) is enabled in SPL.

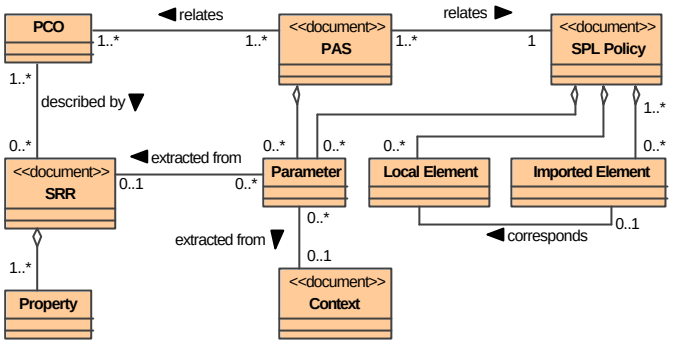


Figure 1. Conceptual model of the SPL Language

The *Policy Applicability Specification* provides an expressive way to relate policies to resources, either explicitly or based on the metadata about the objects (e.g. type of content, owner, price, etc.). PAS documents include three main elements: policy, objects and instantiation. The *policy* element indicates which policy is applicable to the specified objects. Objects are defined by their location as well as by a set of conditions to be fulfilled by the semantics of these objects (SRRs). Optionally, *operation* elements can be used to define which operations of the target object are controlled by the declared policy, allowing a finer grained access control. In case no operation element is included, the policy is applicable to all of the object operations. The *instantiation* element describes the mechanism to instantiate parameters in the policies. Figure 6 shows an example of applicability rules for SPL policies to objects.

The *Secured Resource Representation* 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. An example of an SRR is also included in figure 7. The SRR is designed specifically for the process of dynamic allocation of policies to resources. 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 metamodel is used to locate the right policy for each resource, based on its relevant properties.

It has been mentioned that the creation and maintenance of access control policies is a difficult and error prone activity. The Policy Assistant component is designed to help administrators to specify those policies and validate them to find syntactic and semantic errors. It includes components for the automated validation of policies at different levels. SPL policies are validated syntactically using XML Schema. The *Semantic Policy Validator*, included in the Policy Assistant component, is responsible for the validation of SPL policies with respect to the semantic information available through the metadata defined. Furthermore, it allows policies to be validated taking into account the context where they will be applied. This feature is

supported by SOADs and *Context* metadata. The ability to perform a semantic validation of access control policies is an essential design goal of this access control system. Both, the SPL language and the semantic description of the certificates issued by each SOA (conveyed by SOAD documents) are designed to serve this objective. The semantic validation ensures that the policies written by the administrator produce the desired effects. The Policy Assistant can perform three types of semantic validation:

- *Test Case Validation*: Given a request to access a resource and a set of attribute certificates, we want to know which additional attribute certificates are needed to obtain access to that resource. Most often, this feature will be used to check that a set of attribute certificates is incompatible with the access criteria for that resource. For example, the administrator of our university wants to be sure that it is not possible for a student to access a given resource. For this validation, the Policy Assistant must generate all the attribute certificate sets that are not excluded by the input set of attribute certificates and check those sets against all the possible combinations of attribute certificates that grant access to the resource.
- *Access Validation*: Given a request to access a resource we want to know every set of certificates that gives access to this resource. For this validation, the Policy Assistant must generate the policy for this resource and all the attribute certificate sets from which the attribute certificates required by the policy can be derived.
- *Full Validation*: The goal is to check which resources can be accessed given a set of attribute certificates. Therefore, we generate the policy for each resource and then generate all attribute certificates that can be derived from the input set of attribute certificates. Finally, we inform of every resource that can be accessed using the input attribute certificate set.

4 Elaborating on Content Protection and Payment

Two important issues arise when considering content protection in digital libraries: the secure content distribution mechanism itself and the persistent protection issue. The first one must ensure that contents are protected so that only the intended recipients can access them. In the case of digital libraries it also entails other requirements such as the need to bind the execution of digital rights agreements, payment or other actions to the access to the contents. This is known as provisional authorization or *provision-based access control* (PBAC) (Kudo and Hada, 2000). The second one deals with enabling owners of the contents to retain control over them even when contents are stored in external untrusted servers.

Our solution to the previous problems is based on the use of secure active containers. A *secure active container* (López *et al*., 2002b) is a piece of protected mobile software that conveys the contents and forces the user to fulfil the applicable policy before access is granted. By "protected software" we mean that it is neither possible to discover nor to alter the function that the software performs and it is also impossible to impersonate the software. In our scheme, this is achieved using a variant of the *SmartProt* system (Maña and Pimentel, 2001). Our secure active containers are implemented as Java applets that we call *Protected*

Content Objects (PCOs). They include the contents to be accessed (which are encrypted), the access control enforcement mechanism, and a cryptographic link to the *Mobile Policy* (MP) required to gain access to the contents. We extend the concept of mobile policy described in (Fayad and Jajodia, 2001) by allowing their execution in untrusted systems. Moreover, in our solution policies are bound to the object but not integrated with. This modification makes possible that policies are dynamically changed in a transparent manner. The definition of the MP structure allows a high degree of flexibility.

Figure 2 shows the execution of the PCO. When the client requests some data object from a DL server, it receives the PCO containing it. Before the PCO can execute the protected sections of its code it has to retrieve the corresponding MP sending a request containing the certificate of the public key of the client smart card. In case the server from where the PCO was retrieved is the originator of the PCO, it produces the MP for that PCO. Otherwise the server just forwards this request to the PCO originator. When the MP is received by the client smart card, it is decrypted, verified and stored inside the card until it expires or the user explicitly decides to extract it. Once the MP is correctly installed in the card, the protected sections of the PCO can be executed, which requires the cooperation of the card containing the MP. The protected sections of the software do not reside in the cards. Instead, during the execution of the PCO, these sections are transmitted dynamically as necessary to the card, where they are decrypted using the installed MP and executed. When finished, the card may send back some results. Some other partial results will be kept in the card in order to obtain a better protection against function analysis and other attacks.

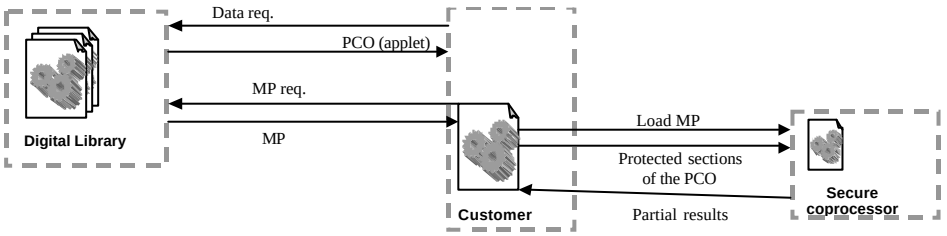


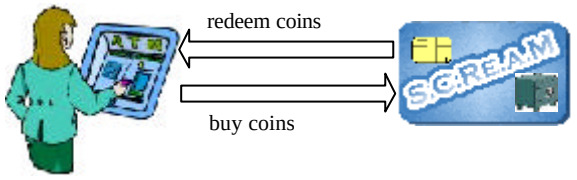
Figure 2. Scenario of execution of the PCO

The main problem when trying to use a payment scheme in information commerce is that payment must be an indivisible part of the transaction. This results in the impossibility to use existing e-purse designs. The integration of the payment mechanism within the process of access to the information is an essential requirement in this case. Therefore we have developed a payment scheme specifically for the XSCD-DL system. The basic idea is to use the capability of controlling the software running at the user side to guarantee the fairness of the process. The basic buying scenario is depicted in figure 3:

Customers buy card money, for instance, in ATM machines from their banks or by Internet. This money is received as a single “unused” coin worth for the amount requested. Unused coins are encrypted for a specific smart card. Therefore, they can not be used by any other card. To make a purchase one of the unused coins in the card is split producing three new coins. The first one is a “paid” coin prepared for the merchant (worth the amount of the purchase). The second coin is a new “unused” coin (worth the rest of the value of the original coin). The third one is a copy of the first one that is marked as “ready” and kept inside

the customer card. The first coin is sent to the merchant in the information request. Once the merchant receives the "paid" coin it produces a MP for that smart card and sends it to the customer. When the customer receives and executes the PCO containing the requested information, the coin state is changed from "ready" to "spent". In case the user finally decides not to access the information of the PCO the coin remains in the "ready" state which allows the user to cancel the payment to the merchant.

"Paid" coins are sent by merchants to the bank as payment orders. Each "paid" coin has a validity interval. Upon reception of the "paid" coin the bank will transfer its value from the customer account to the merchant account. In case the bank also receives the matching "ready" coin the transfer is cancelled. If the customer has not received the license or decides not to execute the PCO, the "ready" coin can be used to cancel the payment to the merchant.



CoinID	Secret	Value	Status	To/From	Date
#1	******	4.25	Spent	WebMerchant1.com	01/01/01
#3	******	14.11	Spent	WebMerchant2.com	02/02/01
#5	******	6.2	Spent	WebMerchant3.com	07/10/01
#12	******	1.02	Ready	WebMerchant4.com	10/11/01
#15	******	51.25	Paid	User89	02/01/01
#0	******	474.42	Unused	Bank1	-
#12	******	57.00	Unused	Bank2	

Figure 3. Coin buying scenario and structure of XSCD-DL cards e-purse

This payment mechanism guarantees that the merchant receives the payment if the user executes the PCO. It also guarantees that the customer can refuse to pay in case the information has not been accessed. Additionally, it provides proofs for the customer in case the information contained by the PCO does not suit the request. Redemption always takes place before the user of the card (in this case the user may be a merchant) buys new money. Before users can buy new money, their "spent" coins and their received "paid" coins are sent to the bank. Optionally, users can also send the "ready" coins that they do not wish to use.

5 System Architecture

XSCD-DL provides different tools to control the entire life cycle of the digital contents: protection, management, distribution and commercialisation. When a new piece of content is incorporated to the DL it is encapsulated into a PCO that protects the contents and enforces the execution of the corresponding access parameters (access control criteria and optionally, associated actions such as payment, notification of access, etc). Both, access control criteria and actions are specified using the *Semantic Policy Language* (SPL) and supported by automated tools. It must be highlighted that it is possible to dynamically change this specification in a transparent way. This is possible because PCOs do not include the access parameters. Instead, a *mobile policy* (MP) containing these parameters is dynamically generated when requested by the client. PCOs can be distributed and copied freely thus enabling superdistribution.

A general overview of the main components of the system and their relation is depicted in figure 4. The first component is the *SmartProt* protection system. This component partitions the software into functions that are executed by two collaborating processors. One of those processors is a trusted computing device

that enforces the correct execution of the functions and avoids that these functions are identified or reverse engineered. We are currently using Java smart cards as secure coprocessors although other alternatives are possible.

SmartProt transforms unprotected content objects in the originator DL server into PCOs as described in section 4. The PCO generation process is independent of the customer card and will be performed just once for each piece of content. One important constraint to the free distribution of protected contents in our system is that originators of those contents must be able to dynamically change the applicable access control policy regardless of the storage location of the PCO. In order to fulfil this requirement, the access control policy and the PCO must be separated. In this way, the MP is retrieved from the originator DL during the execution of the PCO. Requesting the MP at access time from the originator slightly reduces the performance of the system but, in return, it allows a high degree of flexibility and gives the originator more control over the application of the policies. To improve the efficiency and flexibility we have included validity constraints in MPs that can be used to control the need for an online access to the originator server. Consequently, originators can define certain validity constraints for each MP (based on number of accesses, time, etc. depending on the specific smart card features). Hence, MPs can be cached by clients and used directly while they are still valid. The generation of MPs is a fast process while the generation of PCOs is slower. Furthermore, PCOs are much more stable than policies. Finally, opposed to PCOs, each MP is specific for a smart card. PCOs can be managed individually because each one has its own key. This is not possible in other software protection proposals where all applications are protected using the same key.

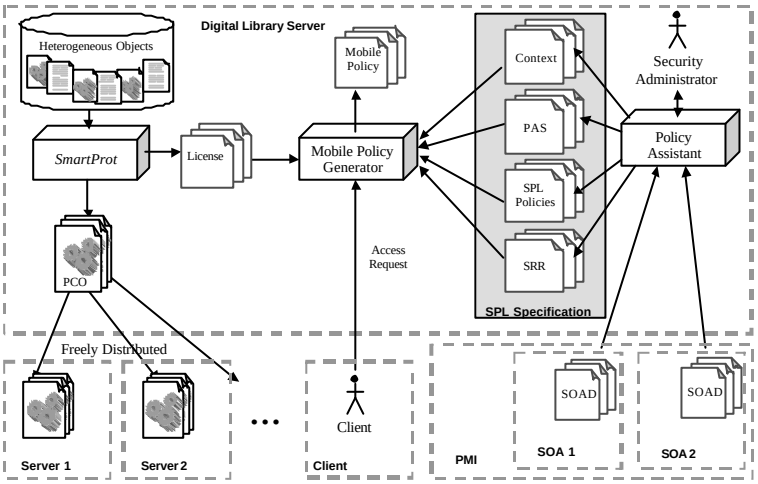


Figure 4. XSCD-DL Infrastructure for Digital Libraries

The second component, *Policy Assistant*, is designed to help security administrators in the specification, management and validation of access control policies. This component is an essential element in the integration of the external authorization infrastructure. It uses the SOADs as a basis for the specification of SPL policies and PAS, being also responsible for the automated validation of policies at different levels. Therefore, the Policy Assistant integrates all the tools to facilitate the administration of the access control system.

The third component, *Mobile Policy Generator*, attends requests from end users producing MPs dynamically. To determine the set of applicable policies for a given PCO, the generator uses different sources of metadata. After receiving a request the Mobile Policy Generator analyses the semantic metadata available for the target PCO, which is contained in SRRs, finds the appropriate PAS and retrieves the necessary SOADs. Using this information, the Mobile Policy Generator is able to find the applicable SPL policies. These policies are then analysed and instantiated using the metadata about the resource (SRRs) and the context. Finally, these policies are combined and sent to the client to be executed. The combination of policies helps reducing the complexity of administration while enabling more expressive and flexible policies to be considered in the access decision.

6 A Digital Library Example

To illustrate the context and semantic validation of policies, lets consider the case of the digital library of a scientific organization that we will call *Association for Computer Societies* (ACS). The ACS has several *Special Interest Groups* (SIGs). Members of the ACS can also join SIGs, although this is not compulsory. ACS publishes journals and newsletters either directly or through SIGs. Newsletters can be accessed by all ACS members and by external (non-members) subscribers. Journals can be accessed by subscribed users (It is not relevant whether they are members of the ACS or not). In case the journal is published by a SIG, all members of this SIG can also access it. There is a special type of subscription called *Portal* that grants access to all publications in the Digital Library.

Figure 5 shows the ACS role hierarchy using a traditional RBAC scheme. This RBAC model requires the definition of:

- A role for each journal (j_1, …, j_n) representing users directly subscribed to this specific journal.
- A role for each newsletter (n_1, …, n_n) representing users directly subscribed to this specific newsletter.
- A role for each SIG (s_1, …, s_n) representing its members.
- A "Portal" role (p), representing members who are subscribed to the Portal.
- An "ACS" role (a), representing all members of the ACS.
- A database relating users with the possible roles they can play.

The arrows in figure 5 show, for instance, that members of the SIG s_1 can play the j_2 and j_3 roles, therefore inheriting all permissions of these roles. The complexity of this model grows exponentially with the number of relations between the different roles. It is also clear that, for this model to work, all user-role and role-role assignments are controlled by a central entity. When several entities have the authority to define these relationships (consider the case of a SIG administrator assigning membership to this SIG), the complexity of management is not acceptable.

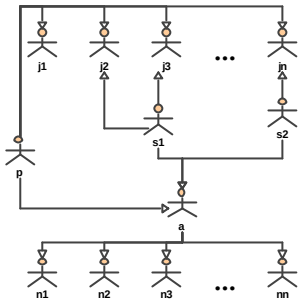


Figure 5. Role hierarchy for the ACS Digital Library

On the other side, considering the *attribute* as the foundation of the access control scheme results in a more flexible approach. It also suits better the requirements of very dynamic environments. An additional advantage is that this scheme enables the distributed issuance and management of authorizations when implemented as attribute certificates. The approach adopted in XSCD-DL is illustrated in the figures of section 6. They show how the use of semantic information, the modularisation, parameterisation and dynamic instantiation of policies results in simple and flexible policies reducing the complexity of management of the system. The specifications also demonstrate how the system can manage dynamic changes in a transparent way. No modifications are necessary in the policies when requirements or properties of resources change. When attributes are considered as the foundation instead of roles, their relations become much more simple and natural as it is shown in Figure 8.

```
<?xml version="1.0" encoding="UTF-8"?>
<PAS xmlns="http://www.lcc.uma.es/OIR"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lcc.uma.es/OIR pas.xsd">
  <Policy>Journal.xml</Policy>
  <Object>
    <ObjectLocation>http://www.acs.org/</ObjectLocation>
    <Conditions>
      <Condition>
        <PropertyName>PublicationType</PropertyName>
        <PropertyValue>Journal</PropertyValue>
      </Condition>
    </Conditions>
  </Object>
</PAS>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="http://www.lcc.uma.es/OIR"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lcc.uma.es/OIR Policy.xsd">
  <Parameter>PublicationName</Parameter>
  <Parameter>PublicationSOA</Parameter>
<!-- instantiated from the SRR for the journal –>
  <AccessRules>
    <AccessRule>
      <AttributeSet>
        <Attribute>
          <AttributeName>Subscription</AttributeName>
          <AttributeValue>*PublicationName</AttributeValue>
          <SOA_ID>*PublicationSOA</SOA_ID>
        </Attribute>
      </AttributeSet>
    </AccessRule>
  </AccessRules>
</Policy>
```

Fig. 6. (a) PAS for ACS journals and (b) its corresponding policy.

Figure 6 shows the policy Journal.xml, which presents two different parameters, the name of the publication and the identifier of the SOA responsible for the issuance of the attribute certificates that certify subscription to this publication. These parameters will be instantiated from the information in SRRs. An

example of SRR showing the semantic properties about the TOSEC journal is shown in figure 7(a). When accessing the TOSEC journal contents, this SRR metadata will be used in the instantiation of the policy. Figure 7 (b) shows the resulting policy after dynamic instantiation.

```
<?xml version="1.0" encoding="UTF-8"?>
<SRR xmlns="http://www.lcc.uma.es/OIR"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lcc.uma.es/OIR SRR.xsd"
Resource="http://www.acs.org/Journals/TOSEC.xml">
  <Property>
    <PropertyName>PublicationlName</PropertyName>
    <PropertyValue>TOSEC</PropertyValue>
  </Property>
  <Property>
    <PropertyName>PublicationSOA</PropertyName>
    <PropertyValue>SIGSEC</PropertyValue>
  </Property>
  <Property>
    <PropertyName>PublicationType</PropertyName>
    <PropertyValue>Journal</PropertyValue>
  </Property>
</SRR>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="http://www.lcc.uma.es/OIR"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lcc.uma.es/OIR Policy.xsd">
  <AccessRules>
    <AccessRule>
      <AttributeSet>
        <Attribute>
          <AttributeName>Subscription</AttributeName>
          <AttributeValue>TOSEC </AttributeValue>
          <SOA_ID>SIGSEC</SOA_ID>
        </Attribute>
      </AttributeSet>
    </AccessRule>
  </AccessRules>
</Policy>
```

Fig. 7. (a). SRR for the TOSEC journal and (b) the resulting policy after dynamic instantiation.

In those access control schemes based on attribute certificates, the semantics of the policies depend heavily on semantics of the attribute certificates. In the SPL access control model, the semantics of the attribute certificates are stated in SOADs (Source of Authorization Description documents). Figure 8 states the descriptions of the source of authorization that certifies the membership to the ACS organization and the subscription to the portal. Relations among attributes certified by each SOA are also described in these documents.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAD xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xsi:noNamespaceSchemaLocation="SOAD.xsd">
  <SOA_ID>ACS</SOA_ID>
  <ACDeclarations>
    <SOAAttribute>
      <AttributeName>Member</AttributeName>
      <AttributeValue>ACS</AttributeValue>
    </SOAAttribute>
    <SOAAttribute>
      <AttributeName>Subscription</AttributeName>
      <AttributeValue>ACSPortal</AttributeValue>
    </SOAAttribute>
  </ACDeclarations>
  <!-- To be subscribed to the ACS Portal implies that you are
  a Member of the ACS and the subscription to the TOSEC -->
  <ACRelations>
    <SOARule>
      <AttributeSet>
        <SOAAttribute>
          <AttributeName>Subscription</AttributeName>
          <AttributeValue>ACSPortal</AttributeValue>
        </SOAAttribute>
      </AttributeSet>
      <Relation>Implies</Relation>
      <AttributeSet>
        <SOAAttribute>
          <AttributeName>Member</AttributeName>
          <AttributeValue>ACS</AttributeValue>
        </SOAAttribute>
      </AttributeSet>
    </SOARule>
    <SOARule>
      <AttributeSet>
        <SOAAttribute>
          <AttributeName>Member</AttributeName>
          <AttributeValue>SIGSEC</AttributeValue>
          <SOA_ID>SIGSEC</SOA_ID>
        </SOAAttribute>
      </AttributeSet>
      <Relation>Implies</Relation>
      <AttributeSet>
        <SOAAttribute>
          <AttributeName>Member</AttributeName>
          <AttributeValue>ACS</AttributeValue>
        </SOAAttribute>
      </AttributeSet>
    </SOARule>
  </ACRelations>
</SOAD>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="SOAD.xsd">
  <SOA_ID>SIGSEC</SOA_ID>
  <ACDeclarations>
    <SOAAttribute>
      <AttributeName>SIGMember</AttributeName>
      <AttributeValue>SIGSEC</AttributeValue>
    </SOAAttribute>
    <SOAAttribute>
      <AttributeName>Subscription</AttributeName>
      <AttributeValue>SIGSECNewsLetter</AttributeValue>
    </SOAAttribute>
    <SOAAttribute>
      <AttributeName>Subscription</AttributeName>
      <AttributeValue>TOSEC</AttributeValue>
    </SOAAttribute>
  </ACDeclarations>
  <ACRelations>
    <SOARule>
      <AttributeSet>
        <SOAAttribute>
          <AttributeName>SIGMember</AttributeName>
          <AttributeValue>SIGSEC</AttributeValue>
        </SOAAttribute>
      </AttributeSet>
      <Relation>Implies</Relation>
      <AttributeSet>
        <SOAAttribute>
          <AttributeName>Subscription</AttributeName>
          <AttributeValue>SIGSECNewsLetter</AttributeValue>
        </SOAAttribute>
        <SOAAttribute>
          <AttributeName>Subscription</AttributeName>
          <AttributeValue>TOSEC</AttributeValue>
        </SOAAttribute>
      </AttributeSet>
    </SOARule>
  </ACRelations>
</SOAD>
```

Fig. 8. (a) SOAD of the ACS (b) SOAD of the ACS TOSEC

7 Conclusions and future work

The system presented in this paper, XSCD-DL, combines an external PMI, a software protection mechanism (*SmartProt*) and a modular language (*Semantic Policy Language*, SPL) for the specification of the access control policies in order to provide distributed access control management and enforcement and

secure content distribution in digital libraries. XSCD-DL allows policies to be dynamically changed by the owner or originator of the resource in a transparent manner.

An important feature is the extensive use of XML metadata technologies to facilitate the security administration in digital libraries and other complex environments. It also enables interesting functionalities of the system such as the contextual validation of policies. In our system, XML-metadata technologies are applied at different levels to express the semantic information. On one hand, metadata are used for the creation and semantic and contextual validation of access control policies. Likewise, metadata about the objects included on the DL enables dynamic policy allocation and parameter instantiation. The higher expressiveness of SPL specifications along with the additional semantic information in the form of XML metadata allows a full integration of the external PMI in the access control system., enabling interoperability among access control mechanisms of different DLs.

To summarize, XSCD-DL represents a solution applicable in different distributed scenarios, is flexible, provides persistent protection, can be applied regardless of the attribute certification scheme, offers distributed access control management and enforcement mechanisms, incorporates secure content distribution as well as an integrated payment mechanism, and allows the dynamic modification of policies transparently and efficiently.

A prototype of this system has been implemented for a Digital Library scenario. In such environment, PCOs are implemented using Java applets. The *e-gate Cyberflex*™ USB Java smart cards are used as secure coprocessors. The high capacity and the transfer speed of these cards makes possible that the performance of the PCO is very good. A set of techniques, such as temporary authorizations, is used to improve the performance. Current work is focused on the extension of SPL for Digital Rights Management application. Additionally, we are interested in different enhancements to the SOAD metamodel, such as support for authorization delegation. Finally, we are considering different interface definition languages to provide an integrated functional and security description of the objects.

References

Baldonado, M., Chang, K., Gravano, L.P and Paepcke, A. (1997), "The Standford Digital Library Metadata Architecture", in *International Journal of Digital Libraries*, vol. 1(2).

Bertino, E., Castano, S. and Ferrari, E. (2001), "Securing XML documents with Author-X", in *IEEE Internet Computing, vol. 5(3):21-31*.

Chadwick, D. W, (2002), "RBAC Policies in XML for X.509 Based Privilege Management", in *Proceedings of IFIP SEC'02, Kluwer Academic Publishers*, available at: http://www.permis.org/

Coleman, A. (2002), "Metadata Rules the World: What Else Must We Know About Metadata?", available at: www.alexandria.ucsb.edu/~acoleman/mrworld.html

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P. (2001), "Controlling Access to XML Documents", in *IEEE Internet Computing, vol. 5, n. 6*, pp. 18-28.

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P., (2002), "A Fine-Grained Access Control System for XML Documents", in *ACM Transactions on Information and System Security (TISSEC), vol. 5, n. 2, pp. 169-202*.

Fayad, A. and Jajodia, S. (2001), "Going Beyond MAC and DAC Using Mobile Policies", in *Proceedings of IFIP SEC'01, Kluwer Academic Publishers*.

ITU-T Recommendation X.509, (2000), "Information Technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks".

Janée, G. and Frew, J. (2002), "The ADEPT Digital Library Architecture". In *Proceedings of the 2nd Second ACM/IEEE-CS Joint Conference on Digital Libraries*.

Ketchpel, S., Garcia-Molina, H. and Paepcke, A.(1996), "Shopping Models: A Flexible Architecture for Information Commerce", in *Proceedings of the 2nd ACM International Conference on Digital Libraries*.

Kudo, M. and Hada, S., (2000), "XML Document Security based on Provisional Authorization", in *Proceedings of the 7th ACM Conference on Computer and Communications Security.*

López, J., Maña, A. and Yagüe, M.I. (2002a), "XML-based Distributed Access Control System", in *Proceedings of EC-Web'02*, LNCS 2455, Springer-Verlag.

López, J., Maña, A., Pimentel, E., Troya, J.M. and Yagüe, M.I. (2002b), "An Infrastructure for Secure Content Distribution". In *Proceedings of the 4th. International Conference On Information and Communications Security 2002*. LNCS 2513, Springer-Verlag.

Maña, A. and Pimentel, E., (2001), "An Efficient Software Protection Scheme", in *Proceedings of IFIP SEC'01. Kluwer Academic Publishers*.

McCollum, C.J.; Messing, J.R. and Notargiacomo, L. (1990), "Beyond the pale of MAC and DAC - Defining new forms of access control", in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 190-200.

M.Thompson, A. Essiari and S. Mudumbai, (2003), "Certificate-based Authorization Policy in a PKI Environment", submitted to *ACM Transactions on Information and System Security,* Aug 2003.

Yagüe, M. I. (2002), "On the suitability of existing access control and DRM languages for mobile policies", in *Department of Computer Science Technical Report nb. LCC-ITI-2002/10, University of Málaga*.