Constrained delegation in XML-based Access Control and Digital Rights Management Standards

Guillermo Navarro[*]
Universitat Autònoma de Barcelona
gnavarro@ccd.uab.es

Babak Sadighi Firozabadi
Swedish Institute of Computer Science
Babak.Sadighi@sics.se

Erik Rissanen
Swedish Institute of Computer Science
Erik.Rissanen@sics.se

Joan Borrell
Universitat Autònoma de Barcelona
Joan.Borrell@uab.es

ABSTRACT

In access control and digital rights management, delegation introduces the ability to decentralize the management of the privileges in a system. Constrained delegation presents a new approach to delegation, where the authority to create a permission and the permission itself is clearly differentiated. This allows the use of delegation for scenarios where one may have the authority to create a permission, but without having the permission for himself. In this paper we examine some of the most popular XML standards for access control and digital rights management, and how constrained delegation can be supported by them. Specifically we take a look at the Secure Assertion Markup Language (SAML), the eXtensible Access Control Markup Language (XACML), and the eXtensible rights Markup Language (XrML).

KEY WORDS
Delegation, Access Control, Digital Rights Management, SAML, XACML, XRML

1 Introduction

Delegation introduces the ability to decentralize the management of access control privileges to various resources in a system. A new approach for delegation and constrained delegation is developed in [1, 6]. In this approach one distinguishes between the authority to create a permission and the permission itself. This means that one may have the authority to create a permission, for certain principals to access a resource, without having the permission for himself or without having the authority to create that permission for himself. This is different from the approach in trust management [2, 4] in which the right to delegate a privilege can be given only to those that have the privilege for themselves.

The constraint delegation model allows one to specify how a privilege can be distributed in several steps. The source of authority for a privilege is able to define how an access permission can be delegated by a number of principals to a specific group of principals by expressing constraints on the groups and time intervals of each delegation in a delegation chain.

An example for the use of constrained delegation is when one organization outsources the administration of its information system to another company. The outsourcing company may have the authority to create access permissions to employees of the organization, but not to any other people. This means also that the outsourcing company does not have the right to create the access privileges for its own employees. However, it may have the authority to delegate the delegation authority for that access permissions within its own organization, hence it has a way of organizing the administration as it wish. This is possible because the organization that owns the information system is able to put constraints on how the outsourcing company can manage access permissions to the information system.

In this paper we will investigate how the constraint delegation model can be encoded in a number of XML standards for access privileges. We have chosen SAML, XACML, and XrML because these are currently the most popular standards for access control and digital rights management. We have not considered other XML approaches out of the standardization efforts. Most notably, models like the Semantic Access Control (SAC) model [10], which present interesting novel approaches are out of the scope of our study.

2 Delegation Model

In this section we provide an overview of the constrained delegation model. A formal definition of the constrained delegation model is given in [1] and [6].

The model uses a certificate based authority management framework with only two types of actions: the issuing of certificates and revoking of certificates. A certificate is represented as:

$$certifies(issuer, p[I], timestamp, id).$$

The certificate expresses that the *issuer* makes an attempt at the time *timestamp* to bring about that privilege p holds for the time interval *[I]*. The time interval is called *validity interval*, and *id* is a unique identifier for the certificate.

The privilege p can be an access-level permission or a management level authority:

[*]Work carried out during a stage at the Swedish Institute of Computer Science.

- *Access-level permission*: defines an access-level action such as read or write file. Denoted as *perm(s,a,o)[I]*, where s is a subject or agent, a is an action, o is an object, and *[I]* is a time interval (validity of the permission).

- *Management-level authority*: defines the authority to declare an access-level permission, or another management-level authority. It is denoted as *pow(s,ϕ)[I]*, where s is a subject, ϕ is either an access-level permission, or another management-level authority, and *[I]* is a time interval (validity of the authority).

And the revocation of certificates can be expressed as:

revokes(issuer, id, [I], timestamp)

The *issuer* revokes the certificate with the identifier *id* during the time interval *[I]*, called the *disabling interval*. The revocation is issued at time *timestamp*.

The model provides the ability to determine whether a privilege p holds at a given time, based on a historical database of certificates and revocations. Informally: a privilege p holds at a time-point t when there is a certificate c declaring that p holds for some interval $[I]$ containing t; the certificate c moreover must be *effective* at t, in the sense that it was issued by s at a time when s had the authority to declare p to hold for interval $[I]$. The authority of s, in turn, requires a certificate that was effective at the time c was issued – and so on, in a chain of effective certificates back to some source whose authority can be accepted without certification (as determined by the organizational structure).

2.1 Delegation example

To show how a delegation can be expressed we give a simple example. Consider two different corporations: *CompanyA*, and *CompanyB*. Alice, who is the project leader of *CompanyA*, has the authority to assign access-level permissions to the project database (*DB*). For instance, consider the following certificate:

$$certifies(Alice, perm(Bob, read, DB)[I_0], t_0, id_0) \quad (1)$$

It declares that Alice authorizes, at time t_0, Bob to $read$ the DB during the time interval $[I_0]$.

Now consider that Carol is a project leader of *CompanyB*, whose employees need to access the resource DB in *CompanyA*. Alice decides to delegate the right to issue access-level permissions to Carol. This way Carol can manage the access to DB for her employees, and Alice does not need to do it. Alice can delegate to Carol the right to issue read permissions for DB to *CompanyB_employees* (see certificate (2)).

$$certifies(Alice,\ pow(Carol,\\ perm(CompanyB\_employees,\ read,\ DB)[I_1])[I_2],\\ t_1, id_1) \quad (2)$$

Note that Alice limits the scope of the delegation, only employees of *CompanyB* can receive an access-level authorization from Carol. And by the other hand there is also a time constraint imposed by the validity interval $[I_2]$, which determines the valid period of time when Carol can issue the access-level permission; and the interval $[I_1]$, which limits the validity interval of the access-level permission.

The scope of the delegation can describe a whole delegation chain. For example, consider that Alice wants to delegate to Bob the right to delegate to the head of engineering of *CompanyA* the right to delegate access-level permissions to engineers of *CompanyA*.

$$certifies(Alice,\ pow(Bob,\\ pow(head\_engineering\_CompanyA,\\ perm(engineers\_CompanyA,\ read, DB)[I_3])[I_4])[I_5],\\ t_2,\ id_2) \quad (3)$$

With certificate 3, Alice defines a valid delegation chain, which says that Bob can delegate the right to delegate the access-level permission only to the head of engineering of *CompanyA*. Each delegation step and access-level permission is limited by a time interval.

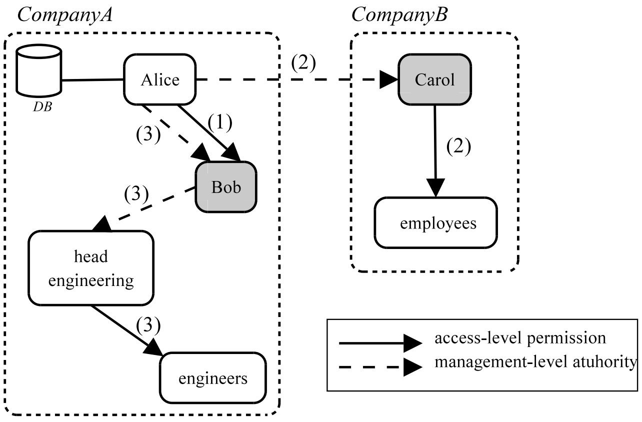

Figure 1. Delegation example

Figure 1 shows the example described with certificates (1,2,3). Solid arrows denote access level permissions and dashed arrows denote management-level authority (the right to delegate). Each arrow is numbered with the associated certificate number. Note that, although employees of *CompanyB* have the access-level arrow, they cannot access to DB. In order to do that, Carol has to issue the access-level permission directly to the employees.

If an engineer of *CompanyA* wants to access to DB, she will need a full delegation chain. That is, a part from certificate (2), there must be another certificate issued by Bob, giving the right to delegate the access-level permissions to the head of engineering; and finally the head of engineering has to issue the access-level permission to the engineer. These three certificates, will form a valid delegation chain for the engineer to access resource DB. It is valid because the first certificate is issued by a source of authority for the resource, it is called a *rooted chain*. If a chain is not rooted it is called *dormant chain*. Dormant chains are possible and may become rooted in the future if the source of authority issues the root certificate for that delegation chain.

In order to reason about the privileges, or in other words, to determine if a given subject can access a given resource, the *calculus of privileges* is used. The calculus of privileges, presented in [6] and [5], assumes there is a database which stores issued certificates and revocations. The database may be stored in a distributed form: the only requirement is that the reasoning engine for determining whether a certain privilege holds, has access to the information.

3 Constrained Delegations for Access Control

In this section we analyze the SAML and XACML standards. Both are seen from the access control perspective.

3.1 Secure Assertion Markup Language

The *Secure Assertion Markup Language* (SAML) is a framework for exchanging authentication and authorization information [8]. SAML information is expressed through security *assertions*. An assertion is defined as a statement (or declaration of facts) about a subject made by an issuer.

SAML provides three different kinds of assertion statements: *Authentication* (the subject has been authenticated by some means at a given time), *Attribute* (the subject is associated with the given attributes and values), and *Authorization Decision* (response to an access request, whether the access has been granted or denied).

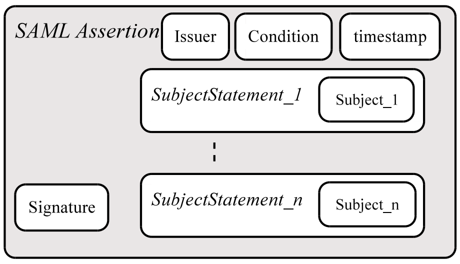

Figure 2. SAML assertion

The basic information contained in a SAML assertion[1] is: the *issuer* of the assertion, a *condition* that must be fulfilled, and an unbounded *choice* of *statements* (see Figure 2). Since it is unbounded, in one assertion we can put several statements of the same type. The statement can be: *Statement*, *SubjectStatement*, *AuthenticationStatement*, *AuthorizationDecisionStatement*, and *AttributeStatement*.

[1]Actually a SAML assertion provides more information like version, time of issue, ... We only consider now the basic information relevant to our work.

The assertion has a timestamp or *IssueInstant*, which determines when the assertion was issued. And may include a signature. As we can see, a SAML assertion is very similar to what is normally called a certificate.

Delegation is not directly supported (not even mentioned) by the current SAML specification and the last draft of version 1.1 published by the time of writing.

In order to encode delegations as SAML assertions we need to extend the current specification of SAML. This extension can be implemented by extending or rewriting the SAML Assertion Schema [7].

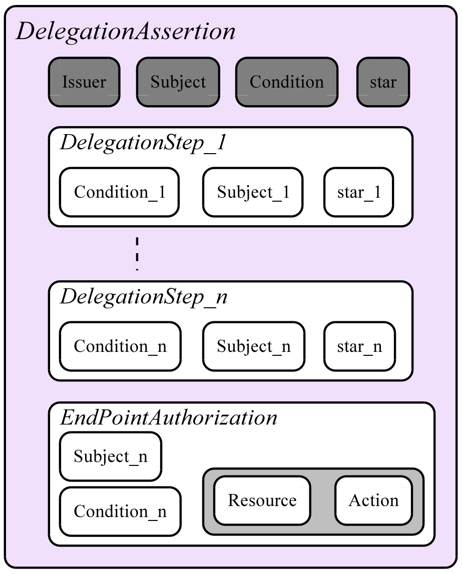

Figure 3. Delegation assertion

In order to express a delegation statement we extend the *SubjectStatement*, since the statement is made about a specific subject. To do that, we extend the *SubjectStatement* with a new *SubjectDelegationStatement*, which can be a *DelegationStep*, or a *EndPointAuthorization*. Figure 3 shows the main elements of a delegation assertion:

- *Issuer*: the issuer of the assertion.
- *Subject*: subject for whom the statement of the assertion holds.
- *Conditions*: conditions and constraints to be enforced.
- *DelegationStep*: the delegation step allows to constraint the delegation chain, by defining each of the following steps to be applied to the delegation. Each delegation step includes the following information:
  - *Subject*: subject of this delegation step.
  - *Conditions*: conditions and constraints to be enforced in this delegation step.

- *EndPointAuthorization*: defines the final authorization which can be delegated. It includes the following elements:
  - *Subject*: subject of the authorization.
  - *Conditions*: conditions and constraints to be enforced in the authorization.
  - *Resource*: resource to which the authorization applies.
  - *Action*: action allowed over the object.

The subject of the delegation assertion can delegate the authorization specified by the *EndPointAuthorization* to the subject specified in the same *EndPointAuthorization*. The *DelegationStep* elements allow to define intermediate subjects in the delegation chain.

The delegation assertion could be included in the SAML assertion schema as:

```
<element name="SubjectDelegationStatement"
         type="SubjectDelegationStatementType"/>

<complexType name="SubjectDelegationStatementType">
  <complexContent>
    <extension base="saml:SubjectStatementAbstactype">
      <sequence>
        <element name="star" type="xsd:boolean"/>
        <element name="saml:DelegationStep"
                 type="saml:DelegationStepType"
                 maxOccurs="uboundend"/>
        <element name="saml:EndPointAuthorization"
                 type="saml:EndPointAuthorization"/>
      <sequence>
    </extension>
  <complexContent>
</complexType>

<complexType name="saml:DelegationStepType">
  <sequence>
    <element ref="saml:Subject"/>
    <element ref="saml:Conditions" minOccurs="0"/>
    <element name="star" type="xsd:boolean"/>
  </sequence>
</complexType>

<complexType name="saml:EndPointAuthorization">
  <sequence>
    <element ref="saml:Subject"/>
    <element ref="saml:Conditions" minOccurs="0"/>
    <element ref="saml:Action"/>
  </sequence>
  <attribute name="Resource" type="anyURI"
             use="required"/>
  <attrubute name="Decision" type="saml:DecisionType"
             use="required"/>
</complexType>
```

Listing 1. SAML Assertion

3.2 eXtensible Access Control Markup Language

The eXtensible Access Control Markup Language (XACML)[9], is an XML-based language, which describes both an access control policy language and a request/response protocol. The current specification of XACML does not explicitly support delegation. In this case we propose to use the existing language constructs and semantics of XACML to express constrained delegations.

XACML provides three top-level policy elements: *rule*, *policy*, and *policy set*. Each one, with a *target* element, which includes *subject*, *resource*, and *action*.

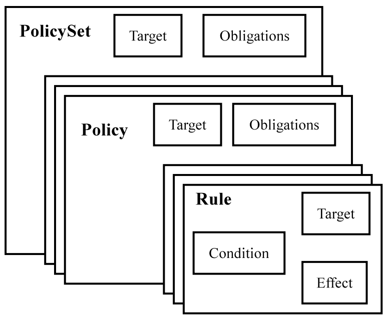

Figure 4. XACML top-level policy elements

We consider the delegation itself as an *action*. Thus, we consider a delegation action over a resource in the *target* element. The resource is the authorization to be delegated, expressed as another *target*.

It is important to note that XACML is a language to specify policies. The issuer of the policy is the *owner* of the policy, and it is not explicitly specified in the policy.

4 Combining XACML and SAML

In this section we propose the use of a combination of XACML and SAML. One of the main limitations of XACML to express the constrained delegation model is that it is difficult to express the idea of a certificate, and all the rules have to be composed in a single policy with one *issuer* or administrator authority. On the other hand, while SAML supports the concept of a certificate, the language provided to express conditions and constraints is more limited than XACML. XACML also provides richer functions and mechanisms to reason about access control decisions.

We propose the use of XACML and SAML in distributed environments to support constrained delegations. The main idea is based in a *context handler* engine responsible for translating SAML assertions and protocol elements into XACML policies. Since XACML and SAML are XML-based languages the context handler may use XSLT to map SAML into XACML and vice versa.

All access control decisions are computed by a PDP (Policy Decision Point) using XACML. And all the communication between several entities of the system is done by using SAML assertions.

A sample query to a PDP is outlined in Figure 5. When a PDP receives a request to access a given resource it queries the PDP using SAML. The SAML query is transformed by the context handler into an XACML context query with the addition of relevant SAML assertion. This SAML assertions will normally be attributes such as groups or roles, but can also contain delegation assertions.

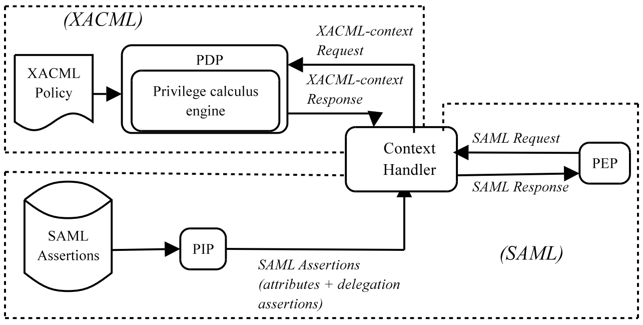

Figure 5. XACML-SAML Request/response

The PDP is responsible for calculating privileges based on XACML policies and attributes. This calculations can be performing by implementing the *privilege calculus* described in [6].

This schema can be applied to highly distributed systems. Each resource of the system may be associated with a PDP. Since the decision of an access request may depend on attributes issued by several authorities and stored in distributed databases, the PIP should include a SAML based protocol to find all the relevant information for a PDP to take an access decision.

5 Constrained Delegations for DRM

The eXtensible rights Markup Language (XrML) is an XML-based language for specifying and managing rights, to control the access to digital content and services[3].

5.1 Overview of XrML

The basic construct of XrML is the *grant*. A grant is used to express that a right has been granted to a given entity, it is composed of:

- Principal: entity receiving the grant. Principals are normally represented as public keys, individually or as a group.
- Resource: object to which the right may be applied.
- Right: the specific right granted.
- Condition: conditions that must be satisfied for the right to be exercised.

A grant is issued by a given principal and both are contained in a *license* element. Figure 6 shows the simplified structure of an XrML License, which may include several issuers, one or more grants, and additional information.

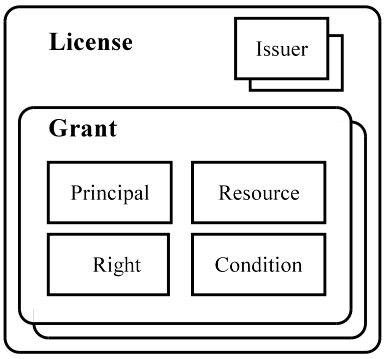

Figure 6. XrML License

5.2 Delegation in XrML

XrML supports delegation of grants from one principal to another. The delegation is controlled by an optional element of the grant called *DelegationControl*. If a grant includes the delegation control, the principal authorized by the grant may issue another license with the given grant to another principal.

It is important to note that in order to delegate a grant to another principal, the former one has to be in possession of the grant. Thus, constrained delegation is not supported by the *DelegationControl*.

One of the rights described in the XrML Core Schema is the *issue* right. This right may be used to specify that a given principal has the right to issue a given right, but the former principal does not necessary hold the right.

A license with an issue right, authorizes the granted principal to issue another license with the right enclosed in the issue right. We can see the issue right as a management level authority (see Section 2). A right that specifies a specific action over an object may be seen as an access level permission.

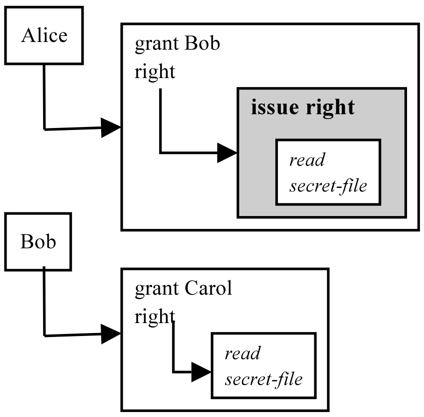

Figure 7. XrML issue right example

For example, imagine that Alice issues a license to Bob with the right to issue the right to read *secret-file*. Figure 7 shows the license issued by Alice with the previous issue right. Then, Bob can issue another license, which grants Carol the right to read *secret-file*.

In this case, Bob is authorized to issue the right to read *secret-file*, but it is important to note that Bob is not authorized to read the file. In other words, Alice delegates to Bob the management level authority to issue the right *read secret-file*, and Bob delegates to Carol the access level permission read *secret-file*.

Another important thing to note, is that the issue right specifies the right, which can be issued, as a full *grant* object. This way, the delegation can be restricted by means of the subjects that may receive the access level permission.

By using the *issue right* we can achieve one step of delegation. That is, the principal receiving the issue right cannot directly delegate the management level authority, the principal can only issue an access level permission. In order to allow a full delegation chain, we can use the *delegation control* of the grant.

5.3 Notes on constrained delegation in XrML

The first thing to note is the expressiveness of the XrML language. This expressiveness allow us to encode constrained delegations without having to redefine or even extend the current specifications. For instance, the delegation path can be constrained in several ways:

- Groups: delegation can be scoped to a given group, but we can not specify the scope of each step of delegation.
- Time: XrML provides validity intervals, issuance time, and several conditions related to time. Fine grained time constraints can be implemented using XrML.

This expressiveness may have some drawbacks. For instance, in the example of Figure 7, the grant issued by Alice with the issue right, could also have a delegation control. Then, the access level permission could also be delegated. It is important to note that this issue may introduce complexity to the management.

6 Conclusions

In this paper we have outlined how constrained delegation can be used in several XML-based standards. We have seen the use of SAML and XACML standards for access control and the XrML standard for digital rights management.

XrML provides a complex and rich language, which allows the encoding of constrained delegations. On the other hand, neither SAML nor XACML directly support delegation. We propose the extension of the SAML specification to support constrained delegations, and the use of the existing XACML specification to express delegations as actions.

7 Acknowledgments

The work of Guillermo Navarro and Joan Borrell has been partially funded by the Spanish Government Commission CICYT, through its grant TIC2003-02041 and Catalan Government Department DURSI, with grant 2001SGR 00219.

The work of Babak Sadighi Firozabadi and Erik Rissanen has been funded by the Swedish Agency for Innovation Systems in the Policy Based Network Management project.

References

[1] O. Bandmann, M. Dam, and B. Sadighi Firozabadi. Constrained delegation. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 131–140, Oakland, CA, May 2002. IEEE Computer Society Press.

[2] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The KeyNote Trust-Management System Version 2. RFC 2704, The Internet Society, September 1999.

[3] ContentGuard. XrML 2.0 Technical Overview, March 2002.

[4] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693, The Internet Society, September 1999.

[5] B. Sadighi Firozabadi and M. Sergot. Revocation schemes for delegated authorities. In *Proceedings of IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, June 2002.

[6] B. Sadighi Firozabadi, M. Sergot, and O. Bandmann. Using authority certificates to create management structures. In *Proceedings of Security Protocols, 9th Internatinal Workshop*, April 2002.

[7] OASIS. SAML Assertion Schema. SAML 1.0 Specification.

[8] OASIS. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.0. OASIS Standard, November 2002.

[9] OASIS. eXtensible Access Control Markup Language (XACML) Version 1.0. OASIS Standard, February 2003.

[10] M. I. Yagüe and J. M. Troya. A semantic approach for access control in web services. In *EuroWeb 2002*, pages 25–33, December 2002.