Web-based Delegation using XML[*]

Konstantina E. Stoupa
Computer Science Department
Aristotle University
Thessaloniki, Greece
kstoupa@acn.gr

Athena I. Vakali
Computer Science Department
Aristotle University
Thessaloniki, Greece
avakali@csd.auth.gr

Fang Li
Jiao Tong University
Shangai, China
li-fang@cs.sjtu.edu.cn

George Andreadis
School of Engineering
Aristotle University
Thessaloniki, Greece
andreadi@eng.auth.gr

Abstract - *Existing access control mechanisms should be extended in order to authorize external (and possibly unknown) clients, when entering distributed environments. This paper proposes the structure and issuing of appropriate authorization certificate to support the delegation process under a role-based access control environment. The proposed processes aim to enhance accessing automation and to avoid (central administrator) bottlenecks (in cases of altering an authorization or a policy). The delegation requests and the certificates are expressed according to the XML syntax for enhancing the interoperability of the delegation processes, which is highlighted in a step-by-step algorithmic fashion using flowcharts.*

Keywords: Delegation, access control, XML.

1 Introduction

The purpose of our work is to introduce a distributed delegation/revocation module (which is part of the access control mechanism) able to serve an Internet-accessed environment whose resources are stored into a distributed fashion. *Delegation* is the passing over of ones authorizations or roles to another. If all requested delegations should be controlled by a single person, then such a central administration point would become a bottleneck in wide environments where such and other requests arrive in large frequency. Those environments demand self-administration mechanisms and one such mechanism is the delegation of roles (or authorizations)

The proposed distributed modules may be integrated into the environment shown in Figure 1. The protected environment consists of several subnetworks supported by a local delegation module (DM). The function of the local modules is supported by rules which have only local effect. Moreover, every local server has access to some global bases containing information recognizable by all servers. Each client should send a delegation request and his XML-based Authorization Certificate (AC) to the connected server. Such a certificate contains access control and delegation information concerning the owner. Those certificates can be extensions of identity certificates Here we use X.509v3 certificates. Servers are categorized into *master* and *slave* ones. The master server receives the requests originating from both internal and external users (through Internet) while the slave ones support only the connected internal users. In [7] we have described how an external request can reach the protected environment.

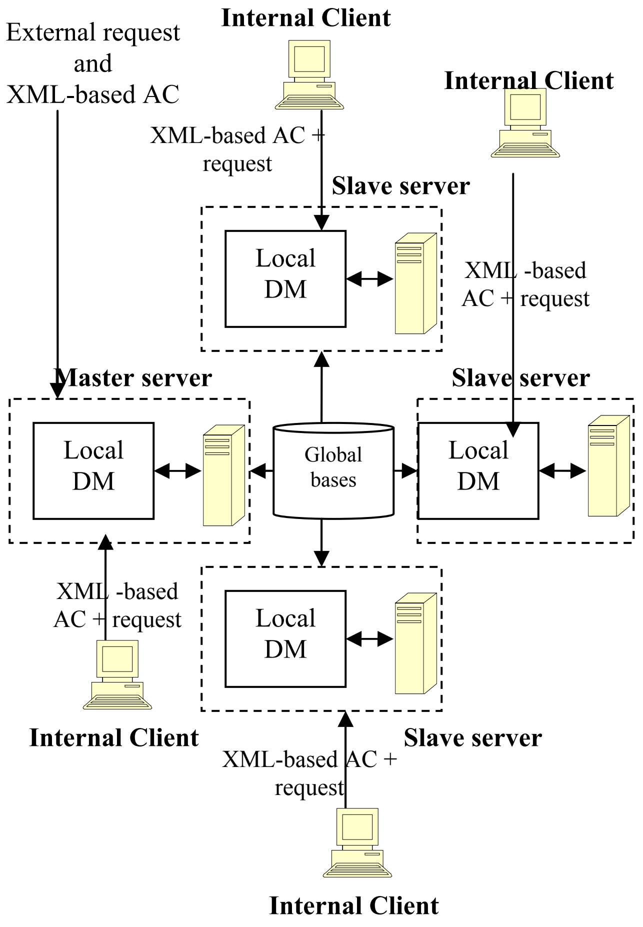


Figure 1: The Distributed Delegation (DM)

Due to space limitations, only the delegation function is analyzed. Thus, the main contribution of the paper is summarized in the following:

[*] **0-7803-8566-7/04/$20.00 © 2004 IEEE.**

- Introduction of XML-based authorization certificates.
- The function of the proposed distributed delegation module able to be incorporated into distributed Internet-accessed environments.
- The supporting of both user-to-user and role-to-role delegation by employing the idea of administrative roles which can modify the features of regular roles.
- Extension of the idea of delegated object to include both roles and authorizations.
- The format and the syntax of the delegation and the revocation requests are identified and an algorithm for the delegation process is given. This algorithm identifies all the issues involved in the delegation process and highlights all the aspects emerged when controlling and completing the delegation process.

The remainder of the paper is structured as follows: In Section 2 the basic issues governing the delegation process are described while in Section 3 the function of the delegation module is discussed. The structure of the delegation request along with their XML-syntax Document Type Definitions (DTD) is discussed in Section 4. In Section 5 the ACs structure is given in XML and in Section 6 the delegation procedure is analyzed in a step-by-step fashion through flowcharts. Finally conclusions are summarized in Section 7.

1.1 State of the Art and Contribution

In the delegation context, RDM2000[1] is a centralized role-based delegation model supporting hierarchical and multilevel delegation [6]. The main idea of this system is that it allows users acting in a specific role to delegate roles to other users. The series of PBDM[2] models extends this idea [9]. PBDM0 also allows the user-to-user delegation of permissions while PBDM1 and PBDM2 supports role-to-role delegation. In order to satisfy such a need, PBDM uses a central security administrator controlling the permission flow by defining separately delegatable roles. All of the above models are centrally administered and therefore, not adequate for large-scale distributed environments.

We believe that our contribution is significant since there is little been done in distributed delegation of authorizations or roles. Moreover, we have decided to use XML to express the major entities of our models since there are already standardized XML-based access control languages, a feature that will help us in integrating our module into existing access control frameworks. Our work advances the current state of the art since we introduce the idea of distributed delegation into Internet-accessed distributed protected networks. Moreover, we have tried to design a both user-to-user and role-to-role delegation of authorizations or roles in order to complete the functionality of such a module. Finally, we have tried to improve our proposal made in [8] where besides the local delegation modules there was also a global one. The reason we have changed the topology is that we wanted to exclude every central point which could become a bottleneck.

2 Delegation Issues

Delegation and revocation are functions of the general access control service. Therefore, we can adopt access models to implement them. In the proposed environment, each subject is associated with an AC which contains its roles and authorizations. Each role has three attributes: (a) *type* defining whether it is regular or administrative (i.e. it is able to modify existing regular roles, or create others), (b) *origin* whether it is global (recognizable by all subnetworks) or local, and (c) *scope* defining the identity of the role hierarchy it belongs to. Both regular and administrative roles can be organized into hierarchies. In regular roles hierarchies a role in an upper level has all of the authorizations related with all roles in lower levels and also some more, while in administrative ones a role can modify the authorizations of roles below it in the hierarchy. Therefore, when a certificate is newly issued (both by the local and external authorization authorities) should include the following information:

- *Licensee*: containing the name and the id of the subject
- *Issuer*: containing the signature of the Authorization Authority which has issued the certificate.
- *Valid period*: This depicts the life duration of the certificate.
- *Regular roles*: a list of the regular licensee's roles.
- *Administrative roles*: a list with the administrative roles of the licensee. This field is blank when the certificate is issued and it is completed by the access control mechanism if needed.
- *Extension fields*: these fields are filled later by the access control mechanism and include delegation information.

3 Delegation Procedure

Figure 2 depicts the format of the proposed delegation module (DM). For the module to work, some information is needed which is stored to the following databases:

<u>Local bases (accessed only by the associated local DM):</u>

[1] Role-based Delegation Model 2000
[2] Permission-based Delegation Model

- *Local roles definition base*: it contains the definition of local roles (which have effect only in the specific subnetwork)
- *Local domain server*: it contains the identities of all users belonging to the specific subnetwork.

Global bases (accessed by all local DMs):

- *Global domain server*: it associates the identity of each user with the subnetwork it belongs to.
- *Global roles definition base*: it contains the definitions of all global roles (roles which are identified by every local DM.
- *Copies of ACs base*

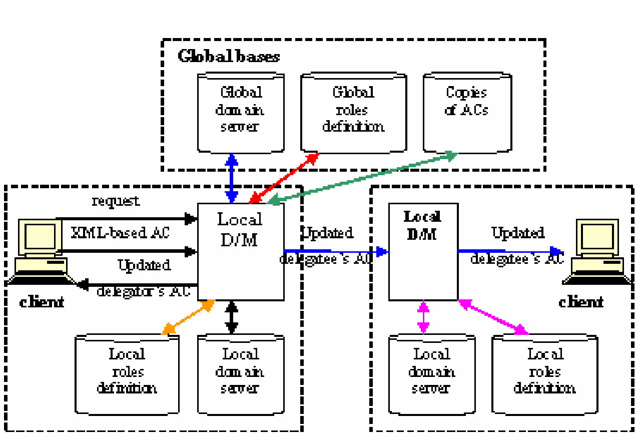


Figure 2 : Function of delegation module

In case of *user-to-user delegation*, we need a delegation request and the delegator's AC (which are sent to the local DM by the delegator), and the delegatee's AC (which is retrieved by the database with the copies of ACs). Afterwards, the local delegation mechanism asks the local domain server if the delegatee is a user supported by the local subnetwork.

- In case, (s)he is a local one, the local delegation mechanism satisfies the request and updates the two parties authorizations certificates which are sent back to their owners (black routes).
- If the local domain server cannot identify the delegatee, the local DM scans the global domain server to find out the domain of the delegatee. If the delegatee is identified the local DM satisfies the request, the delegator receives his updated AC and the delegatee's AC is passed over to the local DM that supports him (black and blue routes).

In each case a copy of the updated ACs is stored to the appropriate base (green route).

In case of role-to-role delegation, after the delegation takes place, if the delegatee is a:

- *Local role*: the associated entry in the local roles definition base is updated (black and orange routes).
- *Global role*: the associated entry in the global roles definition base is updated (black and red routes).

4 The Delegation Requests

The delegation request specifies who wants (*delegator)* to delegate what (*delegation object*), to whom (*delegatee*) under which constraints (*delegation constraints*). Thus, each delegation request involves the :

- *Delegation structure*: defining the delegator and the delegatee, as well as the roles (or authorizations) that are to be delegated (i.e the delegation object).
- *Delegation constraints*: describing the features of the delegation.

4.1 Delegation structure

The *delegation_structure* element is depicted by the following tuple.

(delegator, delegator_role, delegatee, delegated_object, rh_identity, ah_identity)

According to the values that those elements may take, the following cases arise in a delegation request:

- *User-to-user delegation*: when a user acting under a regular role r_1 delegates his regular role r_2 or an authorization to another user (of course r_2 is below r_1 in the role hierarchy).
- *Role-to-role delegation*: when a user acting in administrative role r_1 delegates a regular role or an authorization to regular role r_2 (of course r_1 and r_2 should belong to the same administrative hierarchy).

Since an organization may contain many regular roles and administrative roles hierarchies, we should define the *scope* of the delegation, i.e. to which hierarchies it refers. Thus, *rh_identity* defines the unique identity of the role hierarchy where the regular roles participating in the request belong to, and *ah_identity* defines the identity of the administrative hierarchy where the administrative role belongs to. Of course, this last field may be blank.

4.2 Delegation Constraints

Delegation constraints are related to :

- *Scope*: the scope of its validity which is given by the identity of a role hierarchy.
- *Permanence*: in case a delegation is permanent, the delegator permanently passes on his(her) authorizations to the delegatee.
- *Monotonicity*: this feature refers to the "power" that the delegator possesses after the delegation. In a monotonic delegation, the delegator maintains his(her) authorizations
- *Totality*: this feature refers to the extent with which authorizations assigned to a role are delegated to another. In case of a total delegation the delegator passes over all of his/her authorizations.
- *Levels of delegation*: it defines whether a role can be further delegated and for how many times.
- *Activation/de-activation condition*: every delegation should take place when a condition is fulfilled and it should be cancelled according to a de-activation condition.

A delegation request may or may not contain constraints, or it may contain a part of them. Therefore, the final format of the delegation request tuple is:

(delegator, delegator_role, delegatee+, delegated_object+, srh_identity, arh_identity?, (permanence, monotonicity, delegation_levels, activation_condition, deactivation_condition)?)

The delegation requests (described above) are structured according to the XML syntax which is given in [7].

5 Authorization Certificates and Delegation Procedure

We consider both delegation and revocation (of authorizations and roles) when conducted by both clients and roles. For that purpose, the extension fields of the AC are used, which will now include the following data:

- *Denied authorizations*: this list is expanded every time the licensee acting in a role delegates monotonically an authorization.
- *Delegated objects*: this is a list consisting of the roles and authorizations that have been delegated to the licensee. It consists of tuples of the form:
  (delegator, delegator_role, delegated_object_type, object_id, scope, levels)
- *Revocable objects*: a list containing the roles and authorizations that the licensee has delegated but (s)he has the right to revoke at some time. Of course, this list contains only those subjects that have been temporarily delegated. It consists of tuples of the form:
  revocator _role, revocable_object_type, object_id, scope, levels

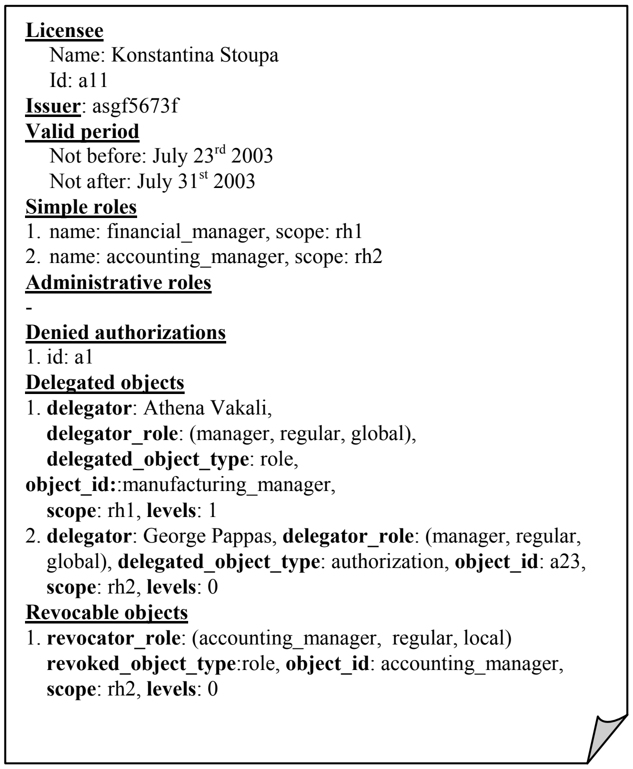
According to the authors knowledge there is no unified AC standard defined yet, so we consider ACs to be defined in a syntax which will facilitate their adoption to Internet-accessed data resources. For this reason, the XML standard is used in order to increase the ACs interoperability and flexibility of use in different (and often heterogeneous) protected resources frameworks.

An example of the syntax of our considered ACs is given in Figure 3 which highlights the definition of an AC tailored for the considered delegation process.

```
<!ELEMENT authorization_certificate  (licensee, issuer,
      valid_period, regular_roles, administrative_roles,
      extension_fields?)>
<!ELEMENT licensee (name)>
<!ATTLIST licensee id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT issuer (#PCDATA)>
<!ELEMENT valid_period  (not_before,not_after)>
<!ELEMENT not_before  (date, time?)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT not_after  (date, time?)>
<!ELEMENT regular_roles(role+)>
<!ELEMENT role (name, scope)>
<!ATTLIST role type (regular | administrative)>
<!ATTLIST role origin (local | global)>
<!ELEMENT scope (#PCDATA)>
<!ELEMENT administrative_roles(role?)>
<!ELEMENT extension_fields (denied_authorization?,
                delegated_object?, revocable_object?)>
<!ELEMENT denied_authorizations EMPTY>
<!ATTLIST denied_authorizations id ID #REQUIRED>
<!ELEMENT delegated_object (delegator, delegator_role,
                                  scope, levels)>
<!ATTLIST delegated_object type (authorization|role)>
<!ATTLIST delegated_object id ID>
<!ELEMENT delegator (#PCDATA)>
<!ELEMENT delegator_role (role)>
<!ELEMENT scope (#PCDATA)>
<!ELEMENT levels (#PCDATA) )>
<!ELEMENT revocable_objects(revocator_role,
                       revoked_object,  scope, levels)>
<!ATTLIST revoked_object type (authorization | role)>
<!ATTLIST revoked_object id ID>
<!ELEMENT revoked_object empty>
<!ELEMENT revocator_role (role)>
```

Figure 3 : DTD for the Authorization Certificate

Moreover, Figure 4 depicts an extended AC. Thus, according to the denied authorizations list, Konstantina Stoupa acting in a role has delegated monotonically authorization with identity a1. Moreover, this delegation was permanent since the delegated object is not included in the revocable objects list. It is interesting to focus on the revocable object which is the accounting manager role. Since this role is also present in the regular roles list, it is obvious that the subject has delegated this role non-monotonically.



Licensee
Name: Konstantina Stoupa
Id: a11
Issuer: asgf5673f
Valid period
Not before: July 23rd 2003
Not after: July 31st 2003
Simple roles
1. name: financial_manager, scope: rh1
2. name: accounting_manager, scope: rh2

Administrative roles
-
Denied authorizations
1. id: a1

Delegated objects
1. **delegator**: Athena Vakali, **delegator_role**: (manager, regular, global), **delegated_object_type**: role, **object_id:**:manufacturing_manager, **scope**: rh1, **levels**: 1
2. **delegator**: George Pappas, **delegator_role**: (manager, regular, global), **delegated_object_type**: authorization, **object_id**: a23, **scope**: rh2, **levels**: 0

Revocable objects
1. **revocator_role**: (accounting_manager, regular, local) **revoked_object_type**:role, **object_id**: accounting_manager, **scope**: rh2, **levels**: 0

Figure 4: An example of an extended AC

6 The Delegation Algorithm

In this section the two types of delegation are analyzed through the use of flowcharts depicting their functions.

6.1 User-to-user delegation

This type of delegation involves request where the delegator is a user acting under a regular role and the delegatee is also a user. In order to follow the flowchart of this procedure shown in Figure 5, we should define the following processes:

- *Local check domain*: it takes as input the delegation request and asks the local domain server whether the delegatee belongs to the local subnetwork or not.
- *Global check domain*: it takes as input the delegation request and asks the global domain server whether the delegatee belongs to one of the supported subnetworks.
- *Update request*: it takes as input the original request and adds the domain of the user.
- *Check validity*: it takes as input the request and checks whether the scope is valid and whether the delegator owns the delegated object (either role or authorization).
- *Completion for users*: it takes as input the updated request and the ACs of the two parties and takes the following actions:
    - o Adds a record in the delegated objects part of the delegatee's AC.
    - o If delegation is temporary, it adds a record in the revocable objects part of the delegator's AC.
    - o If the delegation is monotonic and the delegated object is (a) a regular role, it deletes the object from the regular roles list of the delegator AC, (b) an authorization, it adds a record in the denied authorizations list of the delegator's AC.

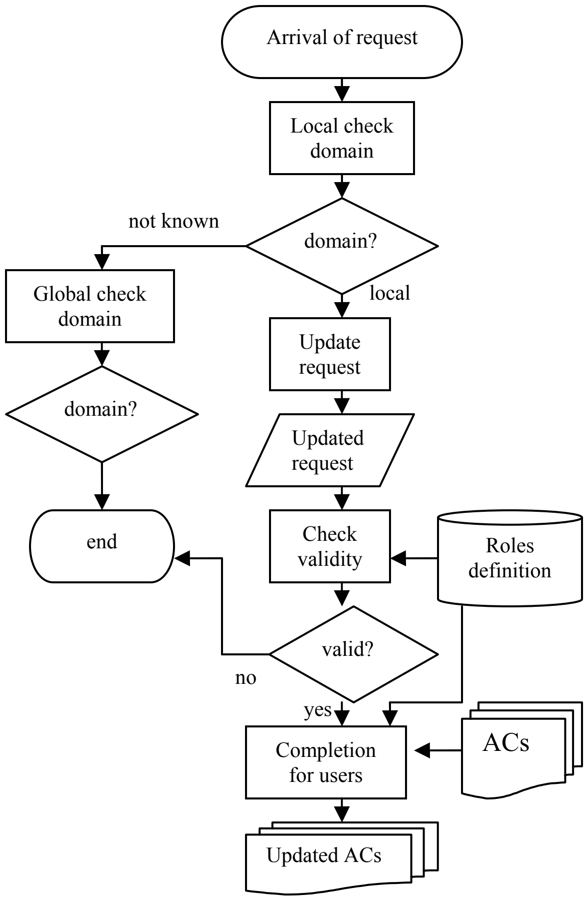


Figure 5 : User-to-user delegation flowchart

After the delegation is completed successfully both delegator's and delegatee's certificates are send back to their owners and a copy of them is stored in the copies of ACs base.

6.2 Role-to-role delegation

This type of delegation involves requests where the delegator is (a user acting under) an administrative role and the delegatee is a regular role. In order to follow the flowchart of this procedure shown in Figure 6, we can use some of the processes defined in Section 6.1 but we should also define the following one:

- *Completion for roles*: it takes as input the request and the delegator's AC and updates the delegatee's definition. According to the origin of the delegatee (local or global) the appropriate database is accessed.

After the delegation is completed successfully the both delegator's certificate is send back to him and its updated copy is stored in the copies of ACs base.

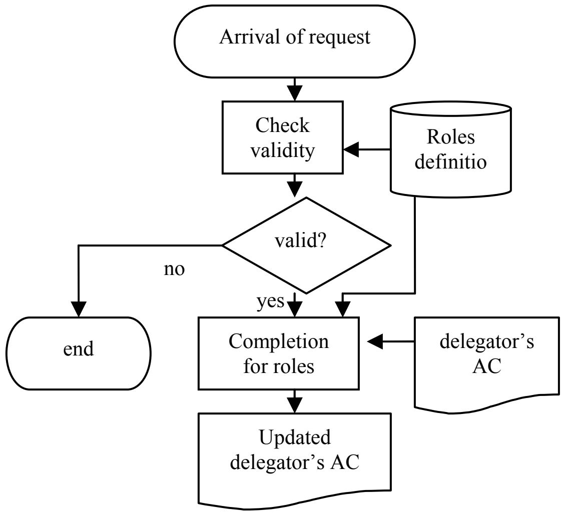


Figure 6 : Role-to-role delegation flowchart

7 Conclusions

Currently, there is a trend in integrating authentication and authorization in one certificate and it is expected that the use of authorization certificates will be adopted by most access control mechanisms. Since there is not yet a uniform standard format of such certificates, there is a need of appropriate mechanisms able to transform the incoming certificates into the ones recognizable by the underlying system. XML is a quite flexible and effective language in expressing such data and here it is used for defining and authorization certificates. This paper's aim is to present the delegation procedure tailored for an Internet-accessed role-based authorization certificates-issuing access control environment supporting distributed resources.

Our work elaborated more on the delegation process which are part of the overall access control mechanism. The syntax used to define delegation requests is given in order to define the appropriate structure to support delegation. The function of both user-to-user and role-to-role delegation is analyzed through flowcharts. The future goal is to implement the proposed structures and algorithms in a prototype authorization-certificates-issuing environment in order to evaluate its usage mainly over the Internet-accessed resources.

References

[1] E. Barka, and R. Sandhu, "Framework for Role-Based Delegation Models", Proc. 16th Annual Computer Security Applications Conference, pp. 168-176, December 2000.

[2] J. Dai and J. Alves-Foss, "Certificate Based Authorization Simulation System", Proc. 25th Annual International Computer Software and Applications Conference, pp. 190-195, October 2001.

[3] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor and Y. Ravid, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers", Proc. Symposium on Security and Privacy, pp. 2-14, May 2000.

[4] N. Li, B. N. Grosof and J. Feigenbaum, "Delegation Logic: A Logic-based Approach to Distributed Authorization*", ACM Trans. On Information and System Security*, Vol. 6, Issue 1, pp. 128-171, 2003.

[5] M. J. Moyer and M. Ahamad, "Generalized Role-Based Access Control", Proc. IEEE 21st Int. Conference on Distributed Computing Systems, pp. 391-398, April 2001.

[6] S. Na, S. Cheon, "Role Delegation in Role-Based Access Control", 5th ACM Workshop on Role-Based Access Control, pp. 39-44, July 2000.

[7] K. Stoupa, A. Vakali, "An XML-based Language for Access Control Specifications in an RBAC Environment", IEEE 2003 Conference in Systems, Man & Cybernetics, Washington, D. C., 2003.

[8] K. Stoupa, A. Vakali, F. Li, I. Tsoukalas, "XML-based Revocation and Delegation in a Distributed Environment", Proc. of the International Workshop on Database Tecchnologies for Handling XML Information on the Web, Heraklion, Greece, 2004.

[9] X. Zhang, S. Oh, R. Sandhu, "PBDM: A Flexible Delegation Model in RBAC", Proc. of ACM SACMAT, pp. 149-157, June 2003.