# Credential-Based Policies Management in an Access Control Framework Protecting XML Resources

Konstantina Stoupa, Zisis Simeoforidis, and Athena Vakali

Department of Informatics, Aristotle University of Thessaloniki, Greece
kstoupa@acn.gr, zsimaiof@csd.auth.gr, avakali@csd.auth.gr

**Abstract.** XML has been widely adopted for Web data representation under various applications (such as DBMSs, Digital Libraries etc). Therefore, access to XML data sources has become a crucial issue. In this paper we introduce a credential-based access control framework for protecting XML resources. Under this framework, we propose the use of access policy files containing policies concerning a specific credentials type. Moreover, we propose the reorganization of the policies in these files based on their frequency of use (the more frequently it is used the higher in the file it is placed). Our main goal is to improve request servicing times. Several experiments have been conducted which are carried out either on single request or on multiple requests base. The proposed framework is proven quite beneficial for protecting XML-based frameworks such as digital libraries or any other data resources whose format is expressed in XML.

## 1 Introduction

Web large-scale adoption has resulted in massive amount of data exchanged daily, and in this context, large scale access control mechanisms have been considered essential in order to prevent unauthorized access to critical resources. Thus, several research efforts and various access control models have been proposed (a description of the most well-known access control models is given in [6], [8]). In an effort to reduce the increasing number of policies required for protecting resources from the wide and heterogeneous clients accessing modern Internet-based environments, modern access control models such as Credentials-based access control model [9] have been proposed. In such models, each subject is associated with some properties (forming credentials) and thus we express policies of the form: "*clients under 18 years old can gain access to documents belonging into the topic* `non-adult staff`".

Among the various languages proposed for expressing credentials and generally protected resources over the Web, XML has become the most popular standard for both modeling resources and expressing security policies. XACML is a standardized framework using an XML-based language for expressing policies. In this paper, we adopt XML to define and organize credentials and access control policies over sets of XML-structured documents (typically stored in large-scale repositories).

A. Levi et al. (Eds.): ISCIS 2006, LNCS 4263, pp. 603 – 612, 2006.

### 1.1 Related Work

Several credentials-based access control frameworks have been designed to protect XML documents. Adam et. al. in [1] propose a credentials-based access control module appropriate for high volume libraries. This model authorizes subjects with specific properties (e.g. age>18) to gain access to protected resources related to a specific concept. Access control authorizations are stored altogether in a *policy base* (as it is called) which is a unique file containing policies and the servicing of a request demand the scanning of the whole policy base leading to high response times.

Author-X [2] is a CBAC framework able to control access to XML documents. Again subjects are characterized by credentials and authorizations are assigned to credentials and not to subject identities. Author-X specializes its protection mechanisms on XML documents. It is an innovative system that introduces methods for reducing the number of policies that have to be specified for a specific file such as the positive and negative policies. Author-X defines as a *policy base* an XML file that contains the access policies for a single file. Each file under system's protection must have a corresponding policy-base file.

Additionally the MaX mechanism proposed in [3] is a credentials-based system for enforcing access control, specifically tailored to both Digital Libraries (DLs) and Web environments. Key features of MaX are the support for credential and content-based access control to DL and Web documents, and its full integration with standard Internet rating systems. The access policies regard User Groups and the total of the policies is stored inside a policy base which is typically a file. MaX was also the first framework that made use of the credential type hierarchy. This hierarchy is based upon the idea that to make the task of credential specification easier, credentials with similar structures are grouped into *credential-types*, organised into a *credential-type hierarchy*.

### 1.2 Paper's Contribution

Although several research efforts focus on introducing access control modules (adopting various models) protecting XML resources, few proposals have been made in order to optimize the access request evaluation process. Murata et. al. [5] have introduced the use of static analysis in improving access queries and thus decreasing time delays in request servicing. Ferrari et. al. [4] propose the use of Access Control XML (AC-XML) documents containing for each object (either XML document or DTD) the associated policies.

*In this paper* we propose an innovative CBAC access control utility protecting XML documents and supporting credential type hierarchy where great concern is given in adopting such policy organization in order to improve the request servicing time. A wide range of Web-based non-CBAC security systems also exists which use either embedded Operating System's techniques, such as the NTFS or databases to control the denial or granting of access to resources. These systems offer very small response times but lack ease of administration as they use single file or directory

access control. We try to provide a framework with response times competitive to those of non-CBAC systems, without abolishing the benefits of CBAC models at the fields of administration and policy definition. The effectiveness of our proposal is proved by experimental results. The proposed system can be *used* for the protection of XML-based digital libraries accesses through Internet and generally in order to protect XML-defined resources. For example, web services, image files whose structure is given in XML, etc.

The proposed CBAC framework uses XML to represent resources, credential types (the properties that a subject belonging in this type should have, e.g. student, secretary, etc.), credentials (instances of credential types, e.g. a specific student with specific values in the student properties) and policies but our main focus is to use such policy organization in order to improve response time. The proposed improvements concern the organization of the policy warehouse:

- *Grouping of policies according to credential types they refer to*: our policy warehouse consists of so many XML *access policy files (AP files)* as the number of credentials types are. Each file contains only the policies that refer to the specific type. Therefore, when a request is received, only the associated AP file is scanned. The frameworks discussed in [1] and [3] do not use such an organization and all of the policies are stored into the same file. Thus, the policy file can be extended endlessly and in order to service a request all of this huge policy file should be scanned. On the other hand, Author-X builds one policy file for each protected object. As the protection of files is directed on policies regarding a single file and not a group of physical entities, the majority of the CBAC model benefits is lost. As a result of this architecture, changing the policies by the administrator becomes a very difficult task.
- *Grouping policies according to the topic of the object they refer to*: moreover to the previous grouping, the policies inside an XML access policy file are grouped according to the topic of objects they refer to. Therefore, by receiving a request, the access control module should search only a specific part of the associated policy file and not all of it.
- *Adoption of credential type hierarchy:* Each AP file also contains the policies related to the super credential types. By this the credentials hierarchy parsing does not take place during access request servicing phase (as it is in [3]) but during idle times when the access policy files are built.
- *Reorganization of policies in the AP file according to the frequency of their use*: since some policies are triggered more often than others, in intervals we reorganize policies in each category of AP files according to the frequency of their use. Thus, the policies that are more likely to be triggered again will be at the beginning of the part of the AP file that should be scanned.

The rest of the paper is organized as follows: Section 2 introduces the basic concepts and definitions involved in credential-based access control. Section 3 describes the proposed framework and the functionality of each involved module is discussed. Section 4 has the results of several experiments which have been carried out to prove the effectiveness of our proposal for optimizing request evaluation time. Finally, Section 5 has the conclusions and future work ideas.

## 2 Credential Based Access Control: Basic Issues

In a typical Credentials Based Access Control System, the system controls which users have access to resources based on their credentials. Access rights are grouped by credential types names, and access to resources is restricted to users who have been authorized to assume the associated role. Here, we formally define the basic components on which our access control module relies. These components are: the protected resources (*objects)*, the *subjects* which are characterized by their credentials and the *access modes* supported by the implemented system. Protected resources are valid XML documents following a specific Document Type Definition (DTD).

*Objects* are the protected resources and they are XML documents and XML Document Type Definitions (DTDs). Therefore, an object can be represented as a 2-tuple (*oid*, *t*) where *oid* is the identity of the object and *t* is the topic where the object belongs. Each object is associated with one topic. Moreover, a finer-grained access control can be achieved if we also denote as objects specific parts of an XML document. In the implemented framework we have not considered such a case but it is in our future plans.

Since the proposed framework is a CBAC one, each subject is associated with a list of properties building his *credentials* assigned during subscription. Each user is associated with a *credentials type* which consists of definition of several properties. Credentials are instances of credential types, i.e. having specific values on the properties. Credentials of a user are stored into XML files whose DTD depicts the credential type they belong to.

In order to service a request, the system should scan an access policy file consisting of several policies. Each policy is a 3-tuple of the form *(subject, object, access_mode)* where *subject* can be a credentials identifier, a credentials type or a credentials expression, *object* can be a whole XML document, or a DTD or part of a document and *access_mode* $\epsilon$ {read, write} is the access privilege.

Policies are triggered by requests sent by the client. Requests can be of the form: *(subject, access_mode, object)* where *subject* is a credentials identifier, *access_mode* can be read or write and *object* is an object identifier.

*X-Univ - A Case study Scenario :* in the following of the paper we will refer to a scenario simulating a university. A university consists of university members which can be students, secretaries or professors. Moreover, the protected resources can be XML files or DTDs which may be grouped into several topics, e.g. courses staff, payroll documents, professors CVs etc. We assume that each object is associated with an identifier known as *object id.* This identifier remains unchanged through life of the document. Often objects belonging into the same *topic* have common protection requirements. Therefore, XML documents are categorized according to their topic. As an example consider the case of having to protect various documents related to the courses, e.g. timetables, exercises, manuals, etc. It is quite sure that subjects sharing similar credentials will have similar authorizations to every object belonging in this topic. Each object is associated with a specific topic but every subject may be interested in various topics. The following is an example of credential type:

```
(student, {(name, string), (type, (postgraduate |
 undergraduate)), (semester, (1…8))}).
```

Fig.1 depicts a credential types hierarchy related to our scenario.

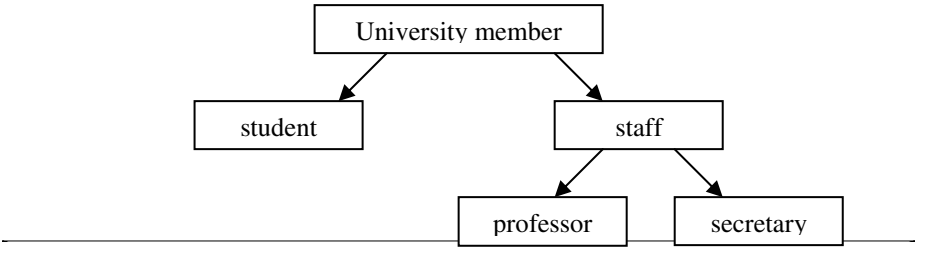


**Fig. 1.** A credential types hierarchy for the university scenario

According to the scenario described some examples of credentials are:

```
(student, 1, (name: Ntina, dept: computer science,
                    semester: 5, type: undergraduate))
(professor, 10, (name: Athena, dept: physics, course:
                    Nuclear Physics, type: postgraduate))
```

In our model only positive policies are defined and in case no policy is expressed servicing a specific request, access is denied. Some examples of policies are the following:

```
Policy1=(student/dept[.="Physics"],
                                physics_courses.xml,read)
Policy2=(professor,courses,read)
```

The first policy defines that every student belonging into the physics department is allowed to read `physics_courses.xml`, while the second one defines that all professors are allowed to read documents belonging into the `courses` topic.

## 3 The Proposed Framework

We propose a credentials-based framework which controls access to XML files and whole groups of files belonging into the same topic, in a quick and secure way, by assigning credentials to system's clients (i.e subjects).

The proposed framework (Fig. 2) consists of several XML-based LDAP directories storing required information about the subject, the objects and the policies. In the credential types LDAP, the XML-based definitions of all credential types are stored (e.g in the X-Univ example, student, secretary, etc.). The credentials of each client following one of those types are stored into credentials LDAP. Since our aim is the improvement of response time, access control policies are organized into XML files according to the credentials type they refer to. Each AP file is further organized according to the topic of the object a policy refers to. The reason we have adopted LDAP directories as storage model is due to the fact that in LDAP, data are organized as a tree like XML documents. Therefore, the transformation from the XML data model to the LDAP data model is not a complex task.

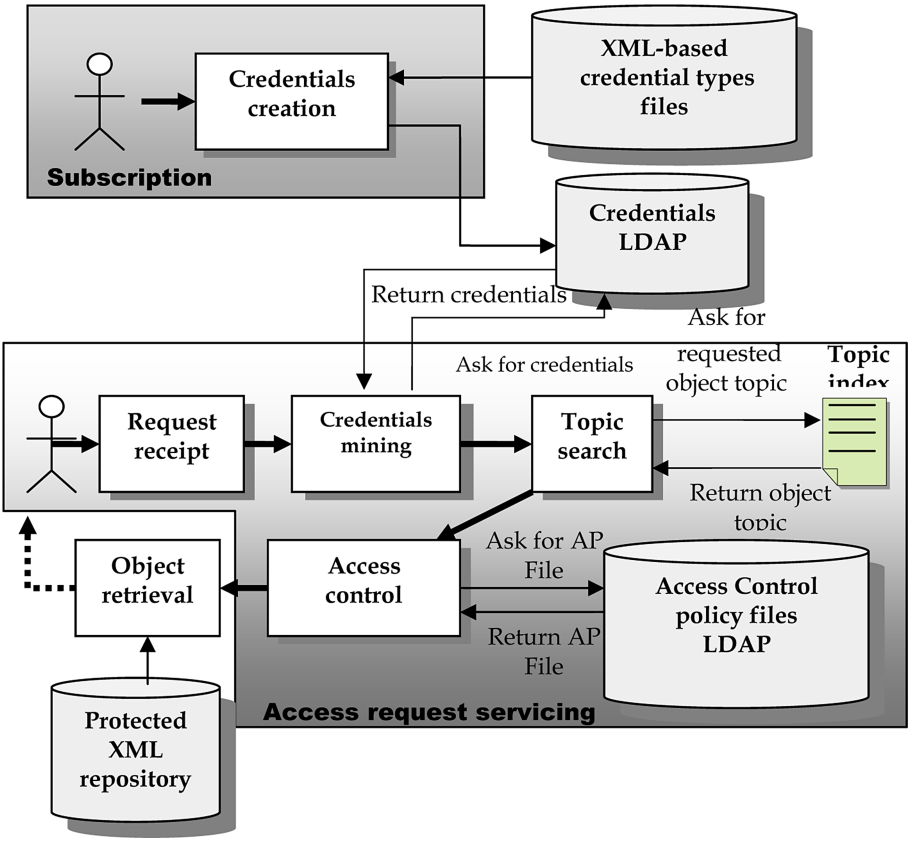


**Fig. 2.** The architecture of the optimized access control utility

Fig. 3 depicts the AP file related to the credential type "`student`". The policies in this file are organized according to the topic of object they refer to (in this example one topic is shown: `courses`) and each category contains both specific and general policies. The object in the specific policies is an object identifier, i.e. a specific XML file, while in the general policies the object is a topic identifier. In order to avoid the parsing of the credential type hierarchy during the access control phase (something that is done in [3]), this file also contains the specific and general policies related to the super credential types (in our example the type "`university members`"). Thus, upon the receipt of a request originating from a subject belonging into a specific credential type, the associated AP file is opened and the part associated with the requested object topic is scanned.

Each policy has an attribute "`frequency`" which is increased by one every time a policy is triggered. During idle times, an administrative module scans each policy file and reorganizes policies according to the frequency of their use. Thus, the most frequently used policies will be forwarded at the beginning of the file in order to optimize the response time.

```
<policy_file type="student">
<topic value="courses">
   <policy frequency="23">
       <subject>…</subject> <object>…</object>
       <access_mode>…</access_mode>
   </policy>
   </topic>
</policy_file>
```

**Fig. 3.** An example of an Access Policy (AP) File

Using the previously defined resources, the function of our framework is divided into two sections: (a) the subscription phase and (b) the access request servicing phase. During the subscription phase the client communicates into the system in order to define its credentials. According to the credential type the client follows, the framework ask him to give values to the properties associated with this type. The subject credentials are then stored into a separate XML file stored into the XML-based credentials warehouse.

The main functionality of the environment is presented during the access request servicing. After the subject has logged into the system, it sends an HTTP access request to Access Control Server. Each HTTP request has to comply with the following schema: *http:// ServerIPAddress : ActivePort / RequestedDocumentName.* From the time a request is received until the time the subject will receive an answer (positive or negative) the following modules are triggered:

- *Credential mining*: according the subject identity, the appropriate file containing its credentials is opened.
- *Topic search*: this module finds the topic that the requested object belongs to.
- *Access control*: this module opens the access policy file associated with the credential type the subject owns and goes to that part of this file containing policies related to the topic the requested object belongs to. Of course the policies are organized according to the frequency of their use and therefore it is very possible to find the appropriate policy at the beginning of the scanned part. If a policy is triggered, a positive answer will be returned to the subject otherwise access will not be allowed. The object retrieval module will be triggered only if a positive answer is returned by the access control module.

## 4 Experimentation

In order to check the effectiveness of the proposed framework we have implemented it using Java. The parsing of XML files depicting credentials and policies has been done using Xerxes and SAX. The experiments were conducted using both synthetic and real access requests. Synthetic requests were stored into a single file from where they were retrieved while real requests were received through http port. According to the X-Univ scenario we have built three access policy files (AP files) (one for every credential type, student, secretary, professor). Each access policy file contained 100 policies of the form discussed in section 2. The AP files followed the format shown in

Fig. 3. Moreover, we have also built a separate unified policy file containing all of the policies totally unorganized, as they were the policy bases in [3] and [1]. We have conducted experiments using both synthetic workload consisting of 100 requests stored into a file and real workload consisting of requests coming to the server through http messages. In this section we will refer to our proposal as *"AP Files Approach"* and to the generally used method were only one policy file is used as *"Unified Policy File Approach"*.

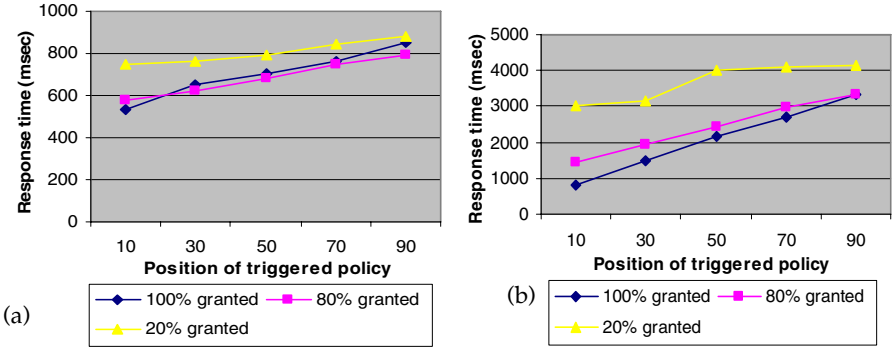


**Fig. 4.** Response times against the position of the policy that is triggered. In both cases three lines are depicted, when 100% of the requests were granted, 80% were granted and 20% were granted. (a) For AP Files Approach, (b) For Unified Policy File Approach.

*Synthetic workload*

Since the response time depends on the position in the policy file of the policy that is triggered, the experiments depicted in Fig. 4(a) and (b) show the response times of both approaches when the triggered policy is found at the 10% first policies, at 50%, 70% and 90%. Moreover, we have depicted the response times when all of the requests were granted, when 20% of them were granted or 80%. Comparing Fig. 4(a) and (b) we realize that the AP Files Approach achieves much better response times. (e.g. if all requests are granted and the triggered policies are found at the middle of the total of policies that are to be scanned, the AP Files Approach response time is 701 msec while in the other approach we have 2163 ms.). In all cases our proposal is three times faster.

Fig.5(a) depicts the response times for servicing one request for both approaches. Again the x-axis contains the position of the triggered policy. In case of the AP File approach the response time remains unchanged independently of the policy position and it is at least two times lower than that of the Unified Policy File Approach.

*Real workload*

We have also tested the behaviour of the two approaches in case or receiving real requests. Fig. 5(b) depicts the behavior of the AP file approach in case it receives one request through http message and the triggered policy is at beginning of the policies that should be scanned, 20% after the beginning, 40%, 60%, 80% and at the end. It is obvious that when using synthetic workload we achieve lower response times.

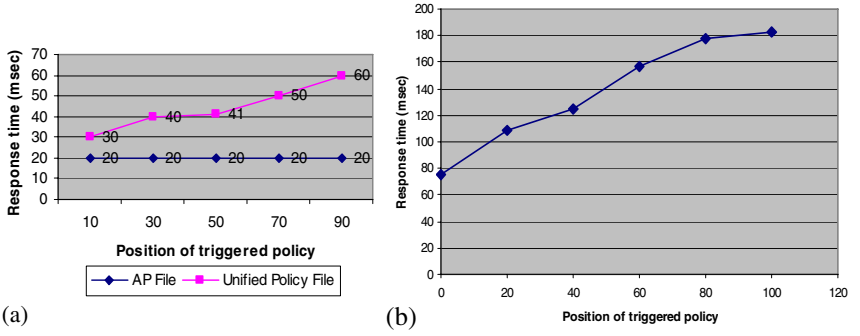


**Fig. 5.** Response time for servicing one request in both approaches against the position in the file of the triggered policy (a) when request originates from synthetic workload, (b)when the request is received through http.

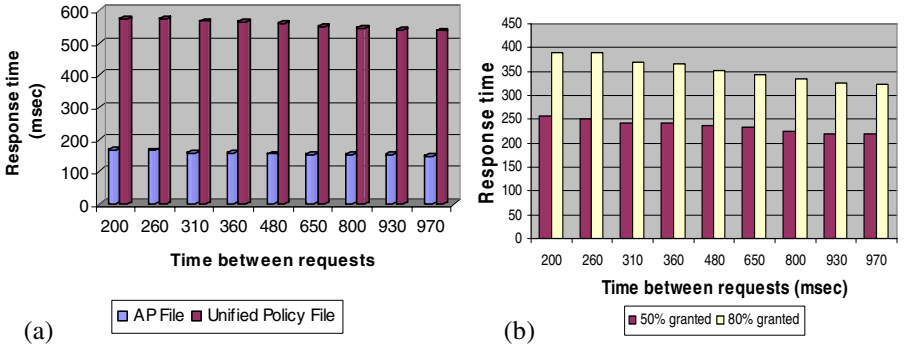


**Fig. 6.** (a) Response time for both approaches according to the time interval between two requests (real workload), (b) Response time for the AP File approach against the time interval between two incoming requests. Each line refers to a different percentage of granted requests.

Commenting Fig. 6(a) we shall notice that in the AP File approach shows much better response time than the Unified Policy File approach. It is quite obvious that in the AP file approach there is an almost linear increase of the response time as the time interval increases. Nevertheless, that is of minor importance because the increase is of a few milliseconds, which is an amount of time that could be barely noticed by a human user. So we conclude, that this slight time increase is very satisfactory, compared to the large intervals.

Studying the behaviour of the AP File Approach when requests are received through http, Fig. 6(b) depicts the framework's average response time compared with the time interval between two requests, in three different situations. The purple column represents the framework's behavior when 50% of the received requests are serviced by a policy at the end of the scanned area and the white one when 80% are serviced by a policy in this part of the area. Satisfactory times are achieved even in the third situation, where the 80% of the requests trigger policies at the end of a policy file.

## 5 Conclusions

In this paper we have introduced a credentials-based access control framework focusing on an optimized organization of access policy base in order to improve request servicing time. By presenting the results of several experiments we have proven that by changing the organization of the access policy base and by reorganizing access policy files according to the frequency that policies are triggered, we can achieve improved response times, either on a single request or multiple requests base.

Our future plans include the extension of our model in order to protect even parts of specific XML documents and generally protect any type of resource whose format can be expressed in an XML file. Moreover, we will experiment on several scheduling algorithms to find which is the most appropriate for reorganizing access policy files according to the frequencies of the triggered policies. Finally, we plan to modify this framework to support Internet-accessed environments where users are unknown and their credentials are sent along with their request signed by a trusted authority.

## References

1. R. Adam, N. R., Atluri, V., Bertino, E., Ferrari, E.: A Content-based Authorization Model for Digital Libraries. IEEE Transactions on Knowledge and Data Engineering, 14(2) (2002), 296-315
2. Bertino, E., Castano, S., Ferrari, E.: Securing XML Documents with Author-X. IEEE Internet Computing, May-June (2001), 21-31
3. Bertino, E., Ferrari, E., Perego, A.: "MaX: An Access Control System for Digital Libraries and the Web", Proceedings of IEEE Int. Computer Software and Applications Conference, Oxford, England, (2002)
4. Carminati, B., Ferrari, E.: AC-XML Documents: Improving the Performance of a Web Access Control Module. Proceedings of the 10th ACM Symposium of Access Control Models and Technologies, Stockholm, Sweden, (2005)
5. Murata, M., Tozawa, A., Kudo, M., Hada, S.: XML Access Control Using Static Analysis. Proceedings of the 10th ACM Conference on Computer and Communications Security, Washington D.C., U.S.A., (2003)
6. Pallis, G., Stoupa, K., Vakali, A.: Storage and Access Control Issues for XML Documents. In: Taniar, D., Rahayu, J. W. (eds.): Web Information Systems. Idea Group Publishing, (2004), 104-140
7. Sandhu, R. S., Coyne, E. J., Feinstein H. L.: Role-Based Access Control Models. IEEE Computer (1996) 38-47.
8. Stoupa, K., Vakali, A.: Policies for Web Security Services. In Ferrari, E., Thuraisingham, B. (eds.): Web and Information Security, Idea Group Publishing (2006), 52-72
9. Winslett, M., Ching, N., Jones, V., Slepchin, I.: Using Digital Credentials on the World-Wide Web. Journal on Computer Security, 5 (1997), 255-267