Access Control for Healthcare Data Using Extended XACML-SRBAC Model

A. A. Abd EL-Aziz
Research Scholar
Dep. of Information Science & Technology
Anna University
Email: zizoah2003@gmail.com

A.Kannan
Professor
Dep. of Information Science & Technology
Anna University
Email: kannan@annauniv.edu

***Abstract*—In the modern health service, data are accessed by doctors and nurses using mobile, Personal Digital Assistants, and other electronic handheld devices. An individual's health related information is normally stored in a central health repository and it can be accessed only by authorized doctors. However, this Data is prone to be exposed to a number of mobile attacks while being accessed. This paper proposes a framework of using XACML and XML security to support secure, embedded and fine-grained access control policy to control the privacy and data access of health service data accessed through handheld devices. Also we consider one of the models, namely Spatial Role-based access control (SRBAC) and model it using XACML.**

Keywords: XACML, SRBAC, XML encryption, XML signature, XML security, mobile

I. Introduction

Health services are a major part of the national infrastructure and hence are a very critical sector of the nation. The information and data related to health services are very confidential and needs to be maintained and accessed with improved levels of security. The information in health services relates to sensitive and confidential information of the patients. These could include the patients history, personal details, and any other secret information. It needs to be made available to the authorized doctors. Privacy is the main concern here and therefore data access control plays a very critical role in health services. Doctors and nurses get access to data and information using their handheld devices. Security of these handheld devices is very important and so is the security of the data coming into the mobile device and going out of the mobile device. The first step that takes place when a patient visits the doctor is that the doctor requests the central health repository for the patients information using his handheld device. The central health repository will send the information to the doctors device after performing initial trust negotiation and ensuring that it is the right request coming from the correct source. The information is sent to the doctors device not in plain text but in an encrypted format. After a few requests and responses between the doctors device and the central health repository, the data will be finally available for the doctor to use. The main parts of the process used in order to retrieve the actual information from the encrypted data involves the use of XACML policy which is used to verify the policy of the mobile device and make policy decisions to send the key that is used to encrypt the information. Based on some computations performed using XACML policy and some cryptographic algorithms, the real time key that is used to encrypt the patients information can be obtained. The real time key will be used in the doctors handheld device to decrypt the information and get access to the actual information[8]. XACML was used not only for defining access control policies. The standard XACML languages and processing models were also extended to allow the access control policies be embedded with the digital content of any type or format in the same XACML document, which serves as a persistent container for both embedded access control policies and the content to be protected. In addition, the original content can be further divided into multiple parts, each of which encapsulated by its own access control policy, to provide finer grained access control. Since then, we have incorporated XML encryption and XML signature into the XACML policy document to further protect the confidentiality, authenticity, and integrity of the content and the access control policy, both embedded in the same XACML document and both can be sensitive information [7]. Many models have been suggested to extend the Role-based access control (RBAC) model to provide location aware access control. In this paper we consider one of the models, namely Spatial Role-based access control (SRBAC) and model it using XACML [2].

II. Related Work

In [11], the authors described a complete composite access control model for mobile agents where authors presented two key aspects of the role-based access control for mobile agents:

1) authorization infrastructure.
2) the structure of role-based access control policies.

The policies map agent role to user role and provide a composite policy for policy Decision point (PDP) decisions regarding access control applied to mobile agents.

The traditional XACML polices, used for user access control in distributed environments, can be used for mobile agents access control [5]. Such polices are used to manage delegation of access rights from users to agents while at the same time following the core principles of the XACML standard. [5] proposed a combination of policies that map users to their mobile agents and make access control decisions for mobile agents by evaluating complex policy sets.

In [6], the policy embedding approach that we use is similar to that of the Enterprise Rights Management (ERM) which is defined as a digital document-based security model that enforces access, usage, confidentiality, and storage policies. However, the method in [7] is based on XACML which is an open standard; while ERM is built upon Microsofts generally proprietary Windows Rights Management Services (RMS) technology that works with RMS-enabled applications to help safeguard digital information from unauthorized use. In [3], the XML-based security standards will be used more and more in terms of an integrated security system, and the possible interaction of different standards was a basic goal in the evolution of XML-based security standards that include XACML and XML Security. The combined use of XACML, XML-ENC, and XML-DSIG in [7] is one more example of constructing an integrated security system focusing on secure, embedded, and fine-grained access control.

III. XACML

XACML [4] is an XML-based language for access control that has been standardized by OASIS (Organization for the Advancement of Structured Information Standards). XACML is an XML encoded language that describes both an access control policy language and a request/response language [1]. The policy language is used to express access control policies (who can do what when). The request language expresses queries (requests) about whether a particular access should be allowed, and the response language describes answers (responses) to those queries. In the response, the answer about the request is available and it is one of four possible values Permit, Deny, Intermediate, or Not Applicable. XACML provides an extensible, flexible, highly expressive, standards-based, and general-purpose access control policy language that can be used for controlling access to any type of resources [1], not just XML documents. In addition, there are other XML based security standards that can be leveraged to enhance the confidentiality, integrity, and authenticity of information. These include the XML Encryption Syntax and Processing [9] and XML Signature Syntax and Processing [10], both from the W3C (World-Wide Web Consortium). Integration of these technologies (access control, encryption, and signature) becomes significantly easier with XML being the common base language for them all. XACML supports an architectural model of separating the policy decision logic from the policy enforcement logic. Three key logical functions are defined by XACML. A Policy Decision Point (PDP) is an entity that evaluates applicable policy and renders an authorization decision. A Policy Enforcement Point (PEP) is an entity that performs access control by making decision requests to a PDP and enforcing the authorization decisions returned by the PDP. The third XACML logical function, Policy Administration Point (PAP), is an entity that creates and manages access control policies. In a typical XACML usage scenario, a subject (e.g., user, application, or process) wants to take some action on a particular resource [2], [7]. The request is the XACML code corresponding to the access request made by the subject to a certain resource. The subject submits its query to the PEP that is responsible for protecting the requested resource (e.g., file system, or web server). The PEP translates the Access request into the corresponding XACML code. This request consists of four parts: Subject, Resource, Action and Environment. Those four elements help in the process of matching the Request with the corresponding Policy(s) in the PDP. The subject is the entity associated with the access request. An example for a subject can be the human requesting the service or the piece of code responsible for creating the request. The Resource element specifies information about the resource(s) for which access is requested. The Action element specifies the action to be performed on this resource. The environment element provide information regarding the environment. This information is not related to the Subject, Resource, or Action. For example, environment element could include an attribute describing the time of access or the place from which the request was initiated. The Response element encapsulates the authorization decision produced by the PDP after evaluating the Request against the existing Policies. The response is composed of a sequence of one or more results each corresponding to a requested resource. Each result element represents an authorization decision for the resource specified in the ResourceId attribute. The result can take one of the four values (Permit, Deny, Intermediate, or NotApplicable). The Status of the Response is an optional element which states the errors (if any) that were encountered during the evaluation of the request. access by the subject [6]. The base construct of all XACML policies is a Policy element. Each XACML document can hold exactly one Policy or PolicySet. The PolicySet is a container that can hold other Policies, PolicySets or other reference found in remote locations. A Policy element must contain a ⟨Target⟩ element which is used for determining whether a policy is fit for controlling a specific request. The Target defines a set of Subjects, Resources, Actions, and Environments. ⟨Rule⟩ is the smallest element among the elements that expresses access control policy semantics. Each rule is composed of Condition, Effect, and Target. The condition is composed of logical expressions defining attribute restrictions of entities and is used for evaluating the request. If the condition evaluates to *True*, the rule is applicable and the effect of the rule takes place. If the condition evaluates to false, the effect of the rule is *NotApplicable*. If the condition evaluates to *Intermediate*, the rules effect is *Intermediate*. A Rule cannot exist alone and must be contained in a policy. A policy can contains 0 to n ($n \geq 1$) rules and provides standard rule combination algorithm to deal with the conflict that occurs when more than one rules make evaluation to one request.

IV. Embedding XML Security in XACML document

This framework is designed with XACML as the base mechanism for embedding fine-grained access control policies and content. The resultant XACML document is further protected with XML encryption [7].

A. Extending XACML

The most significant extension was made to the XACML policy language to allow a ⟨ResourceContent⟩ element within a ⟨Resource⟩ element which is contained within a ⟨Rule⟩ element. The ⟨ResourceContent⟩ element is used to embed the original content encoded into the Base64 format. Access to the data embedded within the ⟨ResourceContent⟩ element is regulated by the policy expressed in the encompassing ⟨Rule⟩ element. Fine-grained access control is supported by allowing multiple ⟨Rule⟩ elements in an XACML document with each ⟨Rule⟩ element specifying its own access control policy for the content embedded within this ⟨Rule⟩ element.

B. Applying XML Encryption

If the entire policy document is to be protected as a single unit, then the element content of the ⟨Policy⟩ element can be encrypted to produce an encrypted version of the XACML document as shown in Figure 1. On the other hand, the element content of each ⟨ResourceContent⟩ element can be encrypted to replace the original content by an ⟨EncryptedData⟩ element. Furthermore, different parts can be encrypted using different keys and thus preserving the fine-grained control capability. Figure 2 shows an example XACML document containing an embedded policy section that having its original content encrypted with a different key.

```
 ⟨ Policy ⟩
  ⟨ EncryptedData Id? Type? MimeType?... ⟩
     .....
  ⟨ /EncryptedData ⟩
⟨ /Ploicy ⟩
```

Fig. 1. XACML with entire policy and content encrypted

```
⟨ Policy RuleCombining...”...:permit-overrides” ⟩
    ....
 ⟨ Rule RuleId=”Rule1”, Effect=”Permit” ⟩
   ⟨ !– Policy & Content for First Part – ⟩
   ⟨ Target ⟩
       ....
     ⟨ Resource ⟩
       ⟨ ResourceContent ⟩
         ⟨ EncryptedData ⟩
           ⟨ EncryptionMethod/ ⟩
           ⟨ KeyInfo ⟩
             ⟨ KeyName ⟩ Key1 ⟨/KeyName ⟩
           ⟨ /KeyInfo ⟩
           ⟨ CipherData ⟩
             ⟨ CipherValue ⟩Data encrypted⟨/CipherValue⟩
           ⟨ /CipherData ⟩
         ⟨ /EncryptedData ⟩
       ⟨ /ResourceContent ⟩
     ⟨ /Resource ⟩
        ....
   ⟨ /Target ⟩
```

Fig. 2. XACML with encrypted element content of ⟨ResourceContent⟩ element

C. Applying XML Signature

The flexibility of the XML signature standard allows our framework to support fine-grained signatures as well. In our framework, only enveloped signatures are used. In this mode, a signature is generated over some portion of the XACML content, and the resulting signature is contained as an element in the same XACML document. The XML encryption and The XML signature can be used together to sign and encrypt the XACML document. Figure 3 shows an example of an enveloped XML signature generated over the element content of the ⟨Policy⟩ element which contains the encrypted data and access control policy.

```
 ⟨ Policy RuleCombining ... ”...:permit-overrides”⟩
  ⟨ EncryptedData Id? Type? MimeType?... ⟩
     .....
  ⟨ /EncryptedData ⟩
  ⟨ Signature ID? ⟩
     ...
  ⟨ /Signature ⟩
⟨ /Ploicy ⟩
```

Fig. 3. XACML with entire content encrypted and signed

The SRBAC model is an extension of the RBAC model. It specifies the spatial restrictions on permission assigned to roles. The SRBAC models consists of five basic components: sets Users (U), Roles (R), Permissions (PRMS), Sessions (S), and Locations (L) to represent the set of users, roles, permissions, sessions, and the spatial location, respectively. Users are considered to be mobile users who establish a wireless connection with the system to access a certain resource. Roles represent the set of permissions to access system resources. Permissions are approvals to execute some operations on one or more resource. Locations describe location domains identifiable by the system. Hierarchies in SRBAC define an inheritance relationship between different roles. Thus, role ri inherits role rj , if and only if all permission that are available for role rj are also available for role ri. Since SRBAC model depends on the location of the users, the inheritance of the SRBAC model should also depend on the location of the users. Thus, if role ri inherits role rj in Location L, this means that ri has all the permissions that rj has in Location L [2].

V. THE PROPOSED ARCHITECTURE

Our proposed architecture based on [7] and it concerns with the implementation of XACML in the mobile environment. XACML policy is used to give access to the data requested by the mobile device from the Central health repository (service provider). The steps begin with the doctor requesting the patient’s information from the Central health repository (service provider). When the service provider receives the request it will send a challenge request and integrated XACML policy based on SRBAC with the key policy decision checks. The integrated XACML policy will contain the encrypted patient’s information by using XML encryption standard and the environment can be the location where the doctor is located. On receiving these, the mobile device will parse through the XACML policy and provide a response in the form of a hash value of the values requested by the XACML request policy from the service provider. The Central health repository receives the challenge response and the hash value. An initial set of secret computation techniques is decided and both the service provider and the mobile devices provider are aware of it. These computations are performed on the hash value and the resulting value is used as a key to encrypt the real time key on the service provide (Central health repository) side. This new encrypted key is then sent to the mobile device. Since the device is also aware of the same secret computation techniques, it will perform the same computations on the hash value which it already has. The resulting value is used to decrypt the encrypted real time key. The real time key is then used to decrypt the encrypted data. The proposed architecture can be summarized in the following points:

1) Doctor requests for the patients information.
2) Embedded XACML policy based on SRBAC with policy decision checks is sent containing the encrypted patient's information.
3) Challenge response and hash of the policy decision values returned to the service provider.
4) Real time key is encrypted using modified hash value and is sent to the device.
5) Real time key is decrypted using the modified hash value. Real time key is used to decrypt the encrypted patient information.

VI. Conclusion

In this paper, we propose an architecture for controlling the access to the data in the health services sector. The main Contribution of the paper is the approach used to control privacy of the data using integrated XACML within the mobile environment. The integrated XACML document, containing both access control policy and the patient's information, is further protected by XML encryption for confidentiality and XML digital signature for authentication and integrity control. The resultant encrypted data and digital signatures are also embedded in the same XACML document and applied in fine granularity. Also we have shown how to model SRBAC model that extends RBAC to incorporate location information in access control decisions to be able to determine the permissions a role encompasses at a given location using XACML.

References

[1] Sun's XACML Implementation Programmer's Guide for Version 1.2. July 11, 2004 http://sunxacml.sourceforge.net/ guide.htm.

[2] M. Aburahma and R. Stumptner. Modeling Location Attributes Using XACML-RBAC Model. *In Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, pages 251–254, 2009.

[3] A. Ekelhart, S. Fenz, G. Goluch, M. Steinkellner, and E. Weippl. Xml security a comparative literature review. *The Journal of Systems and Software*, vol. 81(10):1715–1724, Oct., 2008.

[4] eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, 1 Feb 2005, http://docs.oasis-open.org/xacml/2.0/ access_control-xacml-2.0-core-spec-os.pdf.

[5] A. Giambruno, M. A. Shibli, S. Muftic, and A. Lioy. MagicNET: XACML Authorization Policies for Mobile Agents. *In Proceedings of the International Conference on Internet Technology and Secured Transactions (ICITST)*, pages 1–7, 2009.

[6] G. Hsieh, K. Fostera, G. Emamalia, G. Patricka, and L. Marvelb. Using XACML for Embedded and Fine-Grained Access Control Policy. *In Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, pages 462 – 468, 16-19 March, 2009.

[7] G. Hsieh, R. Meeks, and L. Marvel. Supporting Secure Embedded Access Control Policy with XACML+XML Security. *In Proceedings of the 5th International Conference on Future Information Technology (FutureTech)*, pages 1–6, 21-23 May, 2010.

[8] M. Rajarajan S. Arunkumar. Healthcare Data Access Control using XACML for Handheld Devices. *In Proceedings of the Developments in E-systems Engineering (DESE)*, pages 35 – 38, 6-8 Sept, 2010.

[9] XML Encryption Syntax and Processing. W3C Recommendation, 10 Dec 2002, http://www.w3.org/TR/Xmlenc-core.

[10] XML-Signature Syntax and Processing. W3C Recommendation, 12 February 2002,http://www.w3.org/TR/xmldsig-core.

[11] J. Zhang, Y. Wang, and V. Varadharajan. Mobile Agent and Web Service Integration Security Architecture. *In Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 172–179, June, 2007.