Mediation of Trust across Web Services

Emerson Ribeiro de Mello*and Joni da Silva Fraga
Department of Automation and Systems
Federal University of Santa Catarina
CP 476 – 88040-900 Florianopolis, SC - Brazil
{emerson, fraga}@das.ufsc.br

Abstract

This work presents the use of security proposals in the Web Services architecture aiming to provide an environment that guarantees authentication and authorization transfer between different security domains. The model described facilitates the access of rights owners into an environment with different security technologies. This model is based on the federation web concept, which allows scalable and flexible rights management solutions. This work illustrates the model properties through examples involving different security technologies.

1. Introduction

Transparent to platforms and with a loosely coupled model are characteristics that make Web Services ideal as integrative technology, allowing distributed applications to be flexible and rapidly built in large scale environments, such as the Internet. This integrative characteristic allows existing applications to be available and visible without a great cost being involved. Also the firewalls boundaries can now be crossed. Applications cross several administrative domains, performing multiple hops, involving multiple operations in many services.

These are some of the facilities provided by the Web Services Architecture. But, facilities such as crossing the filtering of packages, considered prohibitive to distributed applications developed in CORBA for instance, are also considered a high risk factor, when security is a requirement to be considered.

Another aspect related to Web Services Model which offers some security difficulties is that server applications are not restricted to direct interactions with their clients. Web services may serve as a proxy of requests from clients which would be routed to other services, the real operations providers. The intermediate accesses are performed on behalf of the principal[1] which originated the initial request.

In order to make the use of Web Services safe and thus guarantee its ample adoption, many security proposals are being submitted to organizations such as W3C (*World Wide Web Consortium*)[2], OASIS (*Organization for the Advancement of Structured Information Standards*)[3] and WS-I (*Web Services Interoperability Organization*)[4].

The proposals above with XML security specifications are intend to cover several security areas. The underlying layers technologies may also be used together to provide greater security. For instance the SSL (Secure Sockets Layer) [8].

The management of distributed applications built according to the Web Services model is also a great challenge, since the management boundaries are crossed, these applications will be under several management models, integrating several technologies of implementation and also involving several mechanisms and security models. Each security domain crossed by a distributed application can provide its own set of security credentials, having as a basis its underlying security technology and policies.

This paper presents our experience in the development of a trust model for service-oriented distributed applications. The model assumes authentication and authorization premises crossing several management and security domains. This model must serve as a mediator within trust schemes of different security domains, involving clients and services laid as distributed application. Therefore, the mechanisms for locating the rights in heterogeneous environments, must deal with different security technologies. These technologies usually express rights and controls in a varied and non-interoperable way. This model´s role is to allow, for instance, the interaction between a client in a domain which uses X.509 [9] and a server whose con-

*Supported by CNPq

[1] Authorized user, process or host by the security policies.

[2] http://www.w3.org

[3] http://www.oasis-open.org

[4] http://www.ws-i.org

trols had been implemented based on SDSI/SPKI certificates [[4], [15]].

This work is organized as follows. Section 2 introduces the idea of grouping the principals through the concept of federations and the trust model for Web Services. Details of the implementation are in section 3. In Section 4 related works are discussed. Finally in section 5 conclusions are made on this work.

2 Trust Model for services oriented environments

In this section we introduce the trust model defined for distributed applications developed through Web Services. The model, since it has, as a base, an integrative technology, must assume authentication and authorization premises crossing several management domains, which the distributed application might reach. Considering this, a fundamental premise is that this model must serve as a mediator between trust models of different security domains, involving clients and services laid out according to the concept of distributed application

The WS-Trust [23] and WS-Federation [21] provide concepts, services and protocols which form the basis for this trust model developed to cross management boundaries and security domains. However, these proposals are omissive as to the dynamics in the establishment of trust relantionships in environments usually heterogeneous and complex. Our model is based on the concept of federations [16], [21] and [11], as a way to group users and in order to facilitate scalability of our solutions in the management of trust relationships. Each federation in our model characterizes a security domain.

2.1 Federations

The management in an environment composed of very different types of individuals, each with different interests, is very difficult, considering an environment of large scale. The classic way to facilitate the administration is always to group them according to abilities and interests. In environments such as the Internet the problem is how to organize these groups and the relationships among them. To make it even more complex, individuals may belong to more than one group. For instance, the clients of a book shop, organized in a group, could pay their bills by check or an electronic card, these belonging to another group, in this case the group of clients of a bank. So, to get scale and reach all the individuals and services, groups must communicate and have trust relationships established among them. These groups may be described as federations. In [16], [21] and [11] federations are proposed, whose objective is to group individuals whom may have interests in common.

The trust model proposed in [16] is based on SPKI federations, and its objective is to resolve chains of authorization certificates SPKI/SDSI [[4], [15]] and the dynamic establishment of new chains of certificates which are the basis for authorization and authentication in large-scale distributed applications. Each federation works similarly to a passive repository of certificates. A principal entering the federation, supplies all authorization certificates which he desire to delegate, so that other (principals) may use the same permissions as his. Scalability in this environment is achieved through associations among the federations (webs of federations). Such associations allow principals to carry out searches through these webs of federations, without having to join numerous federations. It is a equalitarian trust system which does not impose key hierarchy to gain in scale as those formed by X.509's PKI (Public Key Infrastructure).

The grouping of entities (services and clients), through federations, presented in the specifications [21] and [11] aims to reduce the complexity in the management of entities (names) of clients and service providers, however without requiring a central repository for storing these entities.

2.2 Structural aspects of the WS Domain

The proposed model is based on the concept of federation, introduced in [16], that in the text we call WS Domain. The associations of federations allow the construction of "Web Federations" which are used in the our model and are called "Web of WS Domains". In the proposed trust model, each domain is composed of a manager whom groups his various affiliates through their security attributes (credential, certificates, etc). The characteristics of these managers will depend upon the underlying security technology. For instance, if this manager is encapsulated in the infrastructure SDSI/SPKI, he becomes a simple repository of authorization certificates and names of this PKI. If this, on the other hand, corresponds to a Kerberos server [10], then the Authentication and Ticket Granting Services of this server will be available through this manager. In other words, any PKI or security technology is represented by the manager of a WS Domain.

In any of these security technologies, the manager has control over the members, managing their ingression and egression, as members of the domains, as well as queries performed by them. The manager's interfaces with the Web Services World go through the STS (Security Token Services) and XKMS [6].

The STS is part of the WS-Trust specifications and forms the basis of the trust model of Web Services. Since a STS is responsible for issuing or granting the attributes which are valid for all participants in the trust relationship, the establishment of this trust relationship must be characterized through our proposal. The STS service plays a fundamental

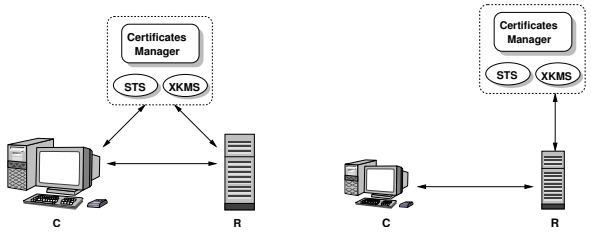
role in our model, mainly in the mediation of trust relationships involving two different security domains. A principal may request, through the STS, security attributes (which according to the technology used, might be: tickets, chains of certificates, etc) needed to access a service within another security domain.

The XKMS allows the localization and validation of keys, and functions as an agent which seeks to take complexity off the client, in dealing with public key infrastructure. The model may be used in the search and validation of chains of certificates SPKI, for instance. The specification is not concerned with which public key infrastructure will be made available by the XKMS interface, it does not describe how key, certificate, etc, might be recovered or validated. When X.509 [9] is used the infrastructure could simply be a repository of keys and certificates accessed through by XKMS, in which trust is achieved through hierarchic relations among valid Certification Authorities, available through STS communications. In the SPKI/SDSI, Certification Authorities are not considered, each principal is apt to issue and sign certificates and the trust relationships are established according to the business policy of each principal. Being a distributed model, the SPKI/SDSI model's major difficulty is in the localization of rights owners. To overcome this difficulty, we use a heuristic (section 2.5) which describes the navigation in the web of domain in an attempt to locate the owner of the desired right for its likely delegation if that is the case.

2.3 Trust relationships in the WS Domain

As described earlier (section 2.2), the WS Domains are composed of a manager and several affiliates (principals), thus there is only a simple trust relationship between the affiliates and the manager of the domain. Security Credentials, issued by the manager's STS, are considered trustworthy by all the affiliates.

Figure 1 illustrates some forms of rights delegation involving only one WS Domain. In the case shown on figure 1(a), a client "C" wishes to call a service provided by "R". Upon receiving the request, "R" verifies whether it is accompanied by the necessary SAML [12] assertions. If not, client "C" is informed of the access policy applied to the service – expressed in WS-Policy [22] – indicating which attributes are necessary to carry out the service (the access). With the challenge at hand the client calls (message `wstrust:RequestSecToken`) the manager's STS service of his domain, which, through a business policy, provides him the necessary assertions (message `wstrust:RequestSecTokenResp`). In possession of the assertions, client "C" responds the challenge to "R" and, this after verifying that assertions presented are valid and enough, grants access to the service[5].



(a) Active client in the requests

(b) Passive client in the requests

Figure 1. Inter-Domain trust relationships

In the case shown in figure 1(a), "R" has a trust relationship with the manager of his domain. Therefore, the assertions issued by the manager are said to be trustworthy. This kind of trust relationship may be classified as "Fixed Trust Root". It is the case in which the client is active, that is, apt to handle challenges posed by the service.

In Figure 1(b), "R" upon receiving and verifying that it does not contain the assertions needed, calls the domain manager, instead of sending a challenge to "C", and request that he generates the assertions needed by "C" this way guaranteeing access to the service. In this case, the client application plays a passive role, not being apt to handle challenges, and requiring that authentication and authorization mechanism be transparent to it.

```
<wsp:Policy xmln:wsse="..." xmlns:wsp="...">
 <wsp:ExactlyOne>
  <wsse:SecurityToken wsp:Usage="wsp:Required"
  wsp:Preference="1">
     <wsse:TokenType> wsse:X509v3 </wsse:TokenType>
  </wsse:SecurityToken>
 <wsp:ExactlyOne>
<wsp:Policy>
```

Figure 2. Message WS-Policy sent in the challenge

Both cases (Passive or Active Client) could be applied in a closed domain, which is composed of several clients and several service providers. The domain manager would be playing the role of Certification Authority, guaranteeing to each client (principal), an authenticated identity associated with access rights. For any client of a domain to communicate with the interfaces STS and XKMS of his manager,

[5]To guarantee protection against replay attack, a random number which will be used only once (nonce) accompanies the protocol.

he will have to import corresponding stubs which allow the execution of the respective protocols.

Details about the delegation of rights and other more complete cases, involving several Web Domains will be explained in section 2.4 through an illustrative example.

2.4 Trust relationships crossing WS Domains

In this section a more thorough example will be presented, which illustrates the steps involved so that a client "C" may access a service provided by "R" (figure 3). The proposal presented in this paper aims at the interoperability among different WS Domains, which may use different security technologies. The following case is composed of three WS Domains. Domain 1 uses SPKI/SDSI for security technology. Domain 3 is based on X.509. Domain 2's manager is able to work with SPKI/SDSI as well as with X.509 and, in the example presented here, will mediate the trust between domains 1 and 3, since there is no trust relation between 1 and 3.

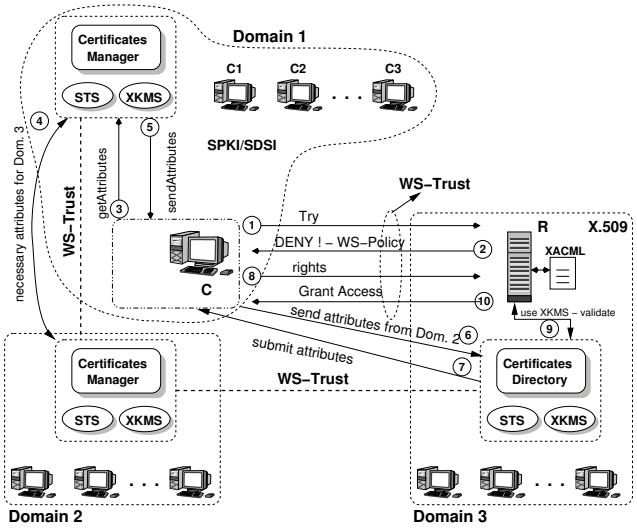


Figure 3. Interaction among participants in the proposed model

Client "C", in domain 1, is requesting access to the service provided by "R" (step 1), which is present in domain 3. The challenge placed by "R" could be of the type "accept assertions issued by principal *X*'s key" (step 2). Client "C" must seek the principal *X*, for this that calling the service XKMS of his manager so that he might localize the respective principal. With this localization, "C" may through a direct communication with the holder of rights (principal *X*), get the delegation of such rights. This is a typical model of the SPKI/SDSI. Through a central rights repository within the domain, the manager helps the members find the rights.

There are cases in which the principal owner of rights does not belong to the SPKI/SDSI domain and thus, is not apt to delegate authorization certificates. Supposing that "R" only accepts security attributes issued by the manager of his own domain. In the case of the example in figure 3, the challenge sent by "R" may contain the location of his manager (domain 3's manager) (step 2). However client "C" will not be able to call him, since the client, in our example, is a principal SPKI/SDSI and does not understand how X.509 works, a PKI which is the basis for the manager of "R". This way, client "C" calls the manager of his domain to try and find a "trustworthy path" which guarantees authorization to service "R" (step 3).

The Managers of "C" (Domain 1) and of "R" (Domain 3) do not have a Trust relationship between them. The manager of "C" must perform a search (heuristic, section 2.5) throughout the domains with which it has trust relationships, in order to find a trustworthy path which will lead him to the manager of "R". In our proposal this search performed by the managers and it is similarly to Gnutela [3] protocol, which facilitates the navigation through the web of associates, this way achieving scale without requiring the manager to know all other likely partners in the web.

In the example in figure 3 the path between domains 1 and 3 is intermediated[6] by domain 2. The manager of "C" requests to STS[7] of domain 2 the necessary attributes to establish communication with the manager of "R" (step 4).

Finally, the manager of domain 1, through his STS service, provides client "C" with all the necessary credentials (issued by him and by intermediate managers) so that he may communicate with the STS service of the manager of "R" (step 5). Assuming a simple negotiation, the manager of "R" analysis the informed credential (in step 6), verifies if they are valid and then supply the credentials (step 7) requested by the client (Figure 4 illustrates the message for the request of credentials to the STS.). Now in possession of the rights, the client sends a challenge response to "R" (step 8), who confronts it with the policies applied to it and, the validation of the signatures using the XKMS service of his domain (step 9), thus guaranteeing access (step 10). Details about access control, applied to resource "R", are shown in section 2.6.

In the case presented above the manager of domain 1 has a fundamental and active role. The manager is responsible to searches for the associate managers, requests the credentials and provides the client with all the necessary means to communicate directly with the owner of rights, in this

[6] Knowing that there might be several intermediate domains, and considering that the greater this number of intermediate domains, less likely it will be for one to get the desired rights.

[7] The business policy involved in the issuing of these attributes is directly related to the application, and it might be simple, in which each request generates a response free of costs, or complex, which would involve the payment of taxes.

```
<soap:Envelope>
  <soap:Header>
    <ws:Security>...</ws:Security>
  </soap:Header>
  <soap:Body>
    <wstrust:RequestSecurityToken>
     <wstrust:TokenType> SAML </wstrust:TokenType>
     <wstrust:RequestType>
       ReqExchange
     </wstrust:RequestType>
     <wstrust:OnBehalfOf>
       <ws:BinarySecurityToken
       id="OriginalToken"  ValueType="SPKI">
         AngqvCfA0cGB+ /WRhiy+9rJ02YHh1C
       </ws:BinarySecurityToken>
     </wstrust:OnBehalfOf>
    </wstrust:RequestSecurityToken>
  </soap:Body>
</soap:Envelope>
```

Figure 4. Message for the request of credentials

case the manager of "R". This kind of behavior makes the implementation of the members of each Web domain less complex, making this search transparent to them. However, another plausible and interesting logic would be the case in which the manager would find the path of associations and would return this information to the client. So the client is the only one responsible for negotiating with each intermediate manager until he gets the rights to communicate with the service requested.

The reliability of the channel, in terms of integrity and confidentiality, is achieved through WS-Security [14] as well as the SAML assertions, not requiring (though not forbidding) the use of mechanisms in the lower layers, for instance the use of the HTTPS protocol.

2.5 Search-for-rights Heuristic

In the case of the example illustrated in figure 3, the manager of domain 1 wishes to locate, among the domains with which it has trust relationships, a trustworthy path which will lead him to the manager of domain 3. A Heuristic is presented here which in our work was conceived for the (peer-to-peer) interaction model. The protocol which follows this model has two messages: *query*, which is used in the search for rights; and the *queryHit*, which informs that the resource has been found. The algorithm below describes how the search for rights is performed, in this case the message "*query*".

The message *query* is composed of four variables: **source** – from where came the request; **resource** – saying which resource to be searched; **P** – a set containing the reverse sequence of all nodes through which the request passed; and **ttl** – which indicates a lifetime for a search, preventing it from extending indefinitely, thus limiting its propagation.

Algorithm 1 $query(source, resource, P, ttl)$

Require: $T = \{$ Table with all nodes where there are trust relations$\}$
Require: $D = \{$ Local directory with information about security domain's members$\}$
1: **if** $(resource \subset D)$ **then**
2: $queryHit(source, resource, P, p)$
3: **else**
4: **if** $(ttl > 0)$ **then**
5: $N \longleftarrow T$
6: **while** $N \neq \varnothing$ **do**
7: $x \longleftarrow getElement(N)$
8: $P \longleftarrow source \cap P$
9: $query(actualNode, resource, P, ttl - 1)$
10: $N \longleftarrow N \setminus \{x\}$ {Removes the element x from the set N}
11: **end while**
12: **end if**
13: **end if**

A node (p), in our case a domain manager in the web, upon receiving a "*query*" message verifies in his local repository – a set D of algorithm 1 – if it has the searched "*resource*" and, if it does, send a "*queryHit*" message to the node which originated the "*query*" message. Otherwise, a "*query*" message is sent to all the nodes (domains) – set T – with which it has trust relationships (lines 6-11). For each new level the "*query*" message descends, the value of "*ttl*" is decremented, preventing the message to propagate indefinitely.

With this heuristic it is possible to cover a great variety of nodes in search for rights without requiring a central repository to tell who has the right over what. The performance of the algorithm may be improved through the implementation of local indexes, which retains information about which resources are provided by which nodes. The maintenance of the indexes is performed through the very queries to that node, leaving stored there for instance, the *n* last queries.

2.6 Access control

This section details step 9 from figure 3, illustrating the process of access control to the resource. The system access policies are defined in XACML [13] and the decision and the enforcement are done through PDP (Policy Decision Point) and PEP (Policy Enforcement Point) [24], respectively. Figure 5 illustrates the access control's steps.

Whenever a client makes a request without authentication assertions issued by a reliable authority, this request is intercepted by PEP, which forwards it to the Authentication Authority (step 1) which in turn queries the X-KISS service – provided by the XKMS interface of the domain's manager – in order to validate the signature (step 2). In step 3 a SAML authentication assertion is generated (step 3).

In possession of the SAML authentication assertion, the

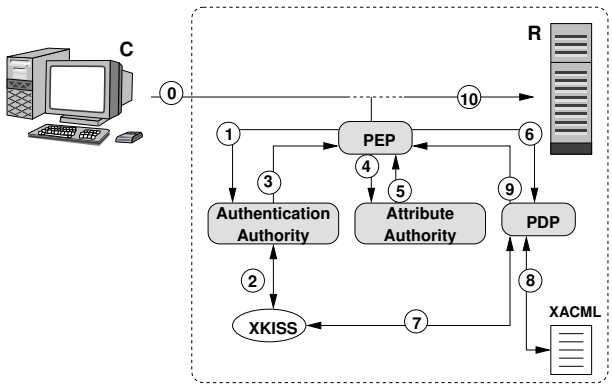


Figure 5. Access control to the resource

PEP supplies it to the Attributes Authority (step 4) in order to obtain a SAML attributes assertion. In the assertion, all the attributes that are related to the SAML authentication assertion provided by PEP are inserted. Finally, it issues an SAML attributes assertion (step 5).

In step 6 the PEP forwards the SAML authentication assertions and attributes assertions to PDP, so that it may determine whether access must be granted or not. The PDP confronts the attributes supplied with the system policies (step 8) and returns a SAML authorization assertion to PEP (step 9). The PEP in possession of the SAML authorization assertion, determines what information the client may obtain from the protected resource (step 10).

3 Implementation

For the implementation Java was the programming language used and as an application server TomCat 5 was adopted. Apache Axis[8] was used as implementation SOAP and to work with encryption and signing of documents XML, the Apache XML Security API[9] was used. And as support to SAML the OpenSAML API[10] was used.

The prototype implemented consists of Web Services in which a domain with a SPKI manager was defined, which allows that trust relationships be established dynamically, providing means to locate and negotiate rights with their respective owners. The services STS and XKMS of the manager of the domain were also implemented, meeting the requirements of interoperability, after all the STS services of our prototype will have to communicate with different managers which encapsulate different security technologies (in the prototype we are initially support the PKIs X.509 and SPKI/SDSI), to allow intermediation among technologies. The development of the XKMS service was necessary since the current implementations, such as TSIK (Trust Service Integration Kit) from VeriSign [18], do not support SPKI/SDSI certificates. The access control mechanisms in the prototype were implemented in the same machine of the application server; but the model supports other configurations, where for instance the PDP could be present in a different machine than that of the application server.

As an example application we use a previous work: the "Papers Search Service". This application addresses the current need for a central repository for academic papers. Usually, each organization has its own paper repository and search engines, what requires users to know how to execute may different queries for each service.

The example application acts as a central repository of academic papers with interfaces for queries about specific papers selected by author, title and others. Therefore, users authenticate once and make queries to obtain results from several organizations (see figure 6).

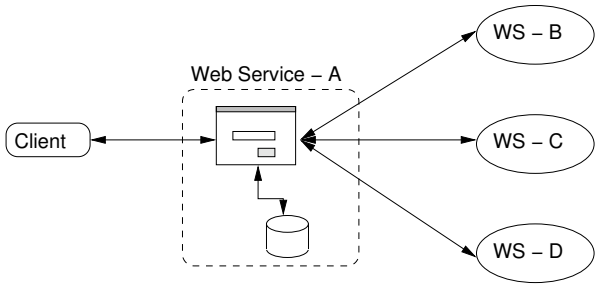


Figure 6. Dynamic of "Papers Search Service"

The application consists of only one simple interface which gathers information derived from different service providers. For the system user, this distribution is transparent and the access control mechanisms among the different systems collaborate through trust relationships among the domains, thus allowing a client's security attributes in a domain to be acknowledged in other domains. The environment comprised by the prototype consists of three domains, in which the trust relationship between the client's domain and service provider's domain is intermediated through a third domain (as illustrated in figure 3).

4 Related Work

The work of Welch [20] describes how to allow the dynamic creation of services as well as trust domains, having his application geared towards Globus' tool kit (a platform for grid computing architecture) [7]. The security infrastructure specification for grid computing architecture, assume the integration with Web Services and benefits from the security standards, such as SAML and WS-Security.

[8] http://ws.apache.org/axis/
[9] http://xml.apache.org/security
[10] http://www.opensaml.org

Some security challenges present in grid computing architecture are shown, being the dynamics of the environment the main challenge, since the service (resources) may be activated or deactivate dynamically during the life cycle of a resources-allocation session. This type of environment congregates several management and security domains, consequently different security technologies. In the proposal, security is provided as services, being *The Credential Conversion Service* responsible for enabling different domains to communicate.

The security services described by Wlech [20] work similarly to the services used in our proposal, however Wlech does not describe how trust relationships are established, nor how to locate, if necessary, possible rights owners. In our work such questions are addressed and its use in grid computing architecture could be adopted without great modifications.

Work [19] proposes a kind of architecture for the establishment of trust relationship among strange parties through gradual unveiling of credentials. A typical case would use a trustworthy third party so that the negotiation may take place.

Such solution becomes a bottle neck in large scale environments. According to Winslett [19], it is difficult to implement zero-knowledge tests efficiently. Therefore, the method "what needs to be known" was adopted, in which the parties involved supply their policies only when necessary. The proposal manages to protect the credentials of the involved parties, however without forbidding the communication among them. But, if the negotiation involves several parties, for each party it will be necessary to establish an authentication, creating a problem in an environment of large scale.

The partial unveiling proposed by Winslett in thesis is similar to the architecture we propose. Each principal is the holder of several rights, described through SPKI/SDSI certificates, or through other forms. The rights are only revealed according to each service. These rights are expressed through SAML assertions, allowing them to transpose domains and operate in large scale environments.

In [17] a model geared towards the negotiation of trust for Web Services is presented. The Model defines services which must allow interaction among attributes authorities and public key infrastructure. State Machines are used in the management of policies' life cycle, associated to the resources.

Like the work of Winslett [19], the proposal of Skogsrud [17] aims to establish trust gradually, using state machines, in which different states would be associated to different rights. The proposal allows changes in the current policies without interrupting the ongoing negotiations. Skogsrud's approach using state machines is interesting since it predicts the path an authentication might follow. In our work the interactions among the principals may assume several paths and a state machine mapping would bring some formalism to our proposal.

In work [5] a security infrastructure for heterogeneous middlewares is presented. To coordinate the trust relationships among the different systems the Keynote [1] was adopted, however the infrastructure also provides support to SPKI/SDSI. The authorization policies of each middleware are coded in Keynote certificates and vice-versa. This allows heterogeneous security domains to be crossed, serving as the basis for the decentralized support of security policies. The work details the advantages of the systems which are based on the concept of trust management [2] on systems which use the X.509.

Foley's objective is to cross limits imposed by technologies through Keynote certificates [5]. In our model we propose to overcome such limits through the use of standards for Web Services, in this case the WS-Trust, which seems more adequate since it is a standard being defined. The crossing of limits brought problems to the localization of rights needed by each domain and that way we describe how to overcome such problem through the concept of federations and the navigation heuristic.

5 Conclusions

The Web Services appeared, through open standards, in order to be an integrative technology, overcoming difficulties present in earlier models, such as the possibility to cross packages filter and the use of XML in the exchange of messages.

Solution proposals to Web Services must guarantee interoperability, since without this characteristic the use of Web Services is worthless. We described in this work a way to integrate applications which use different security technologies. The security proposals to Web Services along with XML security standards were adopted to form the basis of the proposed model. This model provides then, confidentiality, integrity and authenticity, and also a means to locate security attributes, thus enabling the creation of trust relationships dynamically.

In this work the confidentiality of clients' services issue was not addressed. Specifications such as the Liberty Alliance [11] and the proposal WS-Federation [21] propose pseudonyms services which guarantees such confidentiality. As future work we would have the adoption and adaptation of the use of pseudonyms services, meeting the requirements of a niche of applications.

References

[1] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. *The keynote trust-management system version 2.* Internet

Engineering Task Force RFC 2704, September 1999.

[2] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. Technical Report 96-17, AT&T, 28, 1996.

[3] Clip2. *The Gnutella Protocol Specification v0.4*, 2001. Doc. rev. 1.2.

[4] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. *SPKI Certificate Theory*. Internet Engineering Task Force RFC 2693, September 1999.

[5] S. N. Foley, T. B. Quilinan, M. O'Connor, B. P. Mulcahy, and J. P. Morrison. A framework for heterogeneous middleware security. In *18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, 2004.

[6] W. Ford and P. Hallam-Baker. *XML Key Management Specification (XKMS)*. W3C, March 2001. http://www.w3.org/TR/xkms.

[7] I. Foster and C. Kesselman. *The grid: blueprint for a new computing infrastructure*, chapter A Toolkit-Based Grid Architecture, pages 259 – 278. Morgan Kaufmann Publishers Inc., 1999.

[8] A. O. Freier, P. Karlton, and P. C. Kocher. *The SSL protocol - v.3*. Internet Draft, March 1996.

[9] ITU-T. ITU-T recommendation x.509, 1993. http://www.mcg.org.br/mirrors/97x509final.doc.

[10] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (v5)*. Internet Engineering Task Force RFC 1510, September 1993.

[11] Liberty Alliance. *Liberty Architecture Overview v1.1*, January 2003.

[12] OASIS. *Security Assertion Markup Language (SAML)*, November 2002. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[13] OASIS. *eXtensible Access Control Markup Language(XACML)*, February 2003. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[14] OASIS. *Web Services Security: SOAP Message Security 1.0*, March 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

[15] R. L. Rivest and B. Lampson. SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession, 1996.

[16] A. Santin, J. Fraga, F. Siqueira, and E. Mello. Federation WEB: A scheme to compound authorization chains on large-scale distributed systems. In *22nd Symposium on Reliable Distributed Systems (SRDS 2003)*, Florence, Italy, 2003.

[17] H. Skogsrud, B. Benatallah, and F. Casati. Modelo-driven trust negotiation for web services. In *IEEE Internet Computing*, pages 45– 52. IEE Computer Society, December 2003.

[18] VeriSign. *VeriSign Digital Trust Services: Enabling Trusted Web Services*, February 2002.

[19] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. Negotiating trust on the web. In *IEEE Internet Computing*, pages 30–37, December 2002.

[20] C. Wlech, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for grid services. In *12th IEEE Int. Symp. on High Performance Distributed Computing*, 2003.

[21] *Web Services Federation Language (initial draft)*, July 2003. http://msdn.microsoft.com/ws/2003/07/ws-federation.

[22] *Web Services Policy Framework (initial draft)*, September 2004. http://msdn.microsoft.com/ws/2004/09/policy/.

[23] *Web Services Trust Language (initial draft)*, May 2004. http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Trust.asp.

[24] R. Yavatkar, D. Pendarakis, and R. Guerin. *A Framework for Policy-based Admission Control*. Internet Engineering Task Force RFC 2753, January 2000.