A Model to support SPKI Federations management through XKMS

Michelle Wangham[12*], Emerson Ribeiro de Mello[1*], Joni da Silva Fraga[1*], Davi da Silva Böger[1*]

[1] Department of Automation and Systems
Federal University of Santa Catarina
Florianopolis, SC - Brazil

[2]Embedded and Distributed Systems Group
CTTMAR - UNIVALI
São José, SC - Brazil

wangham@univali.br,{emerson,fraga,dsborger}@das.ufsc.br

Abstract

The purpose of the XML Key Management Specification (XKMS) is to facilitate the use of a Public Key Infrastructure (PKI) by transferring the complexity associated with PKI to a trusted Web Service. Although this specification contains information on how compatibility with PKIs such as PGP and SPKI/SDSI can be reached, it is straight focused on X.509 PKI. This work uses XKMS to define a federated management model for SPKI/SDSI. In a web of SPKI Federations, the proposed model follows a peer-to-peer approach for discovering and establishing certificate chains. This model introduces a search algorithm and its effectiveness was verified through simulations.

1 Introduction

Several security standards, such as *XMLEncryption* [11], *XMLSignature* [3] and SAML [15], are used in order to provide security to the information expressed in XML and, consequently, to Web Services. These standards are based on the use of public key infrastructures (PKIs), through key manipulation and certificate issuance. Among the PKIs supported on these standards, one can highlight X.509, PGP and SPKI/SDSI [8, 18].

The *XML Key Management Specification* (XKMS)[10] aims to remove from application developers the complexity involved in using PKIs. When PKI management is transferred to the XKMS service, the information and operations related to this infrastructure become accessible through a standard protocol which is independent of the underlying security technology.

The XKMS Service is deeply focused on PKI X.509, whose global, hierarchical trust model places a few constraints as regards scalability and flexibility, despite its wide use. It should also be noted that the hierarchical enforcement of trust faces legislation-related issues in some countries, among other problems.

SPKI/SDSI (*Simple Public Key Infrastructure - Simple Distributed Security Infrastructure*) follows an egalitarian model, in which the principals[1] (subjects) are public keys and each public key is a certification authority [6]. In this model, there is neither a centralized entity for public key registration and certificate issuance, such as X.509's certification authority, nor a hierarchical global infrastructure. The issuer of an SPKI authorization certificate may enable the subject of the respective certificate to delegate the received permissions to other principals. The SPKI/SDSI delegation model enables, for example, the building of authorization chains which depart from a service provider and end in client keys of this service. In this trust model, the principal wishing to gain access to a resource is totally responsible for searching certificates chains which can grant rights to access the resource. However, the SPKI/SDSI specification lacks a way to perform those searches, which leaves the problem unresolved.

In [19] (see section 2), an extension of the SPKI/SDSI trust model was proposed, called SPKI Federation, which facilitates the publication and location of name and authorization certificates, and also provides support for the creation of new authorization chains. The model proposed in [19] aims to maintain the policy adopted in the SPKI/SDSI

*Supported by CNPq - BRAZIL

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

model, and makes the principals responsible for locating the chains of certificates they wish. However, the trust model proposed in [19] overloads the client, as the latter has not only to understand the PKI's complexity but also to be responsible for searching certificates on the federation web.

This article aims to describe an extension to the SPKI Federation Model [19] through XKMS. In the model proposed, the responsibility for discovering authorization chains is transferred to the XKMS service, which also becomes responsible for trust management among the members and associates of the SPKI Federation. Besides the federated management model for SPKI/SDSI, this study also presents a flexible algorithm (see section 5) for the discovery of authorization chains which follows the concept of flooding with different behaviors according to the level of trust existing among the federations.

2 SPKI Federations

A limitation of the SPKI/SDSI trust model lies in the identification, among a client's certificates, of trust paths (authorization chains) that binds the client to the desired server. In some cases, a client and a service may not be connected by an authorization chain. In [19], the SPKI/SDSI trust model receives proposals of extensions that override this drawback. SPKI Federations, which group principals with common interests, act as facilitators to discovery certificates and principals. Their main functions are centered in the federations' certificate manager, which provides mechanisms to store, recover and create authorization chains. Thus, sharing name certificates and authorization certificates in repositories becomes an alternative to the clients, given the lack of appropriate chains for the desired access.

A principal, when affiliating with an SPKI federation, is allowed to use the repository in order to find certificate chains. Nevertheless, an SPKI federation has a limited scope to reach a certain level of scalability on the web. To solve this problem, it is proposed in [19] that the federations, through their certificate managers, associate with other federations, and thus create mutual trust relationships which extend the trust model and form *webs of SPKI federations*. Federation web aids a client in the search for access privileges that link it to the desired service in large-scale networks.

3 XKMS

The XKMS specification [10] defines a Web Service for PKI management that, through protocols which are independent of the underlying security technology, provides distributed applications with a greater focus on business models, and thus leaves the PKI's management complexity and handling under a Web Service's responsibility. XKMS enables the use of different PKIs without the need to modify the distributed application.

In the XKMS specification, protocols are defined which enable(1) key pair generation; (2)storage, discovery and validation of public key information, and (3) signature validation. This specification comprises two parts: X-KISS (*XML Key Information Service Specification*), whose aim is to locate information associated with the keys, and X-KRSS (*XML Key Registration Service Specification*), responsible for registering this information.

The main goal of X-KISS is to aid applications in the use of signatures expressed in the *XML Signature* standard [3]. XKRSS enables a client to associate information with a key. This information can be a name, an identifier (e.g., e-mail) or specific attributes for certain kinds of applications. This mechanism also manages the information associated with the public key. XKRSS defines the following services [10]:

- register: associates information with a key;
- recover: enables the recovery of the private key previously associated and generated by XKRSS;
- reissue: enables the generation of new credentials in the underlying PKI (reissuance of a key's information);
- revoke: revokes information associated with a key.

4 A Federated Management Model to SPKI through XKMS

In the SPKI/SDSI trust management model, trust chains are uncontrolled paths, and their members are responsible for the storage and recovery of the sequence of certificates in order to verify the authorizations. This model is deeply focused on the client wishing to access a resource.

The federated management model proposed in this text is an extension to the SPKI federation model; the management and trust establishment functions among the elements of the federation, formerly assigned to the certificate manager, are transferred to the XKMS service. Moreover, in order to make client applications independent of the underlying security technology, the XKMS service is held responsible for locating certificate chains as well.

This way, in this new model, the XKMS service facilitates the interaction between the client and the application server by encapsulating the management functions of an SPKI federation and providing support for the navigation on the federation web. However, the service is not characterized as an intermediate key, as shown in Figure 1. The XKMS service of each federation has access to the repository of its federation and provides two services: XKISS and XKRSS (see Figure 1). XKISS is responsible for the

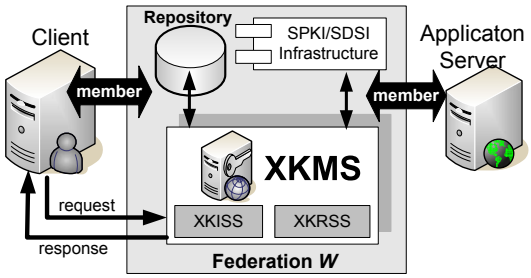

Figure 1. SPKI Federation Elements in WS-XKMS

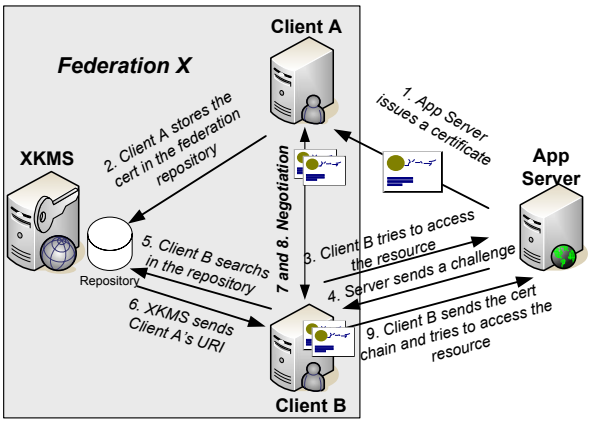

Figure 2. Integration between the XKMS service and the SPKI Federated Model

management of SPKI information and provides the locate and validate operations. XKRSS, in its turn, is used in the management of an SPKI federation to provide the register operation[2].

Although the XKMS specification supports different PKIs, the integration between the XKMS service and the SPKI trust model is not a simple one, because the XKMS specification is focused on the X.509 trust model. The next section provides a detailed description of this integration, as well as the management functions assigned to the XKMS services and the SPKI certificate search algorithm proposed in this study.

4.1 Integration between the XKMS service and the SPKI Federated Model

In the proposed management model, each XKMS service serves only the members of its SPKI federation and has no active participation in any authorization chain. The example in Figure 2 illustrates the manner and dynamics whereby the management model supported through the SPKI Federations is integrated into the XKMS service.

In the scenario in Figure 2, one can observe that *clients A and B* are members of Federation X. In **step 1**, the application server issues and sends an authorization certificate that can be delegated to *client A*. This client now has access rights to the resource protected by the guardian, and can also delegate these rights. In **step 2**, *client A* stores this certificate in the repository of its federation so that it is available to the other members. In **step 3**, *client B* attempts to access the application server's resource, but the service guardian verifies that *client B* has no access rights and sends a challenge to this client to prove that it has the required permissions to access the resource (**step 4**). In other words, the client should present a signed request and a chain of certificates which enables the guardian to verify the granted authorizations. As the challenge is posed, *client B* first performs a search in its local repository. As *client B* does not find the chain of certificates, in **step 5**, it sends its XKMS service a locate request message, so that the service can carry out the search in the repository of its federation. The XKMS service finds the certificate stored by *client A* in step 2 and returns *client A's* URI (*Uniform Resource Identifier*)(**step 6**). With the URI available, *client B* can negotiate the delegation of access rights with *client A* (**step 7**)[3]. After the negotiation, *client A* issues an authorization certificate to *client B* (**step 8**) and, finally, *client B*, having the chain of certificates that binds it to the desired service, signs this chain and sends it to the application server(**step 9**), which, in turn, grants access to the resource(**step 10**).

4.2 Creating SPKI Federation Webs through Associated XKMS Services

In the model proposed in this study, an SPKI federation can associate with other federations, forming a web of SPKI federations through its XKMS Services. A web of SPKI Federations is formed in an arbitrary way[4] and creates trust paths among the associated federations so that a member can perform queries to the XKMS service of the original federation. This service, in its turn, performs the search — in the name of its member — in the XKMS Services of associated federations.

[2]The reissue and recover operations in XKRSS are not used in the proposed model, as they are particularly aimed at private key management. In the SPKI policy, only the owner of the private key is allowed to manipulate these operations. Neither the revoke function was used, as the use of certificate revocation lists is not recommended for SPKI/SDSI.

[3]The negotiation of rights can be conducted in a simple way (through delegation), as the principals are members of the same SPKI federation. However, depending on the application semantics, more complex negotiation may be demanded.

[4]Since they are established and removed in a dynamic, random way.

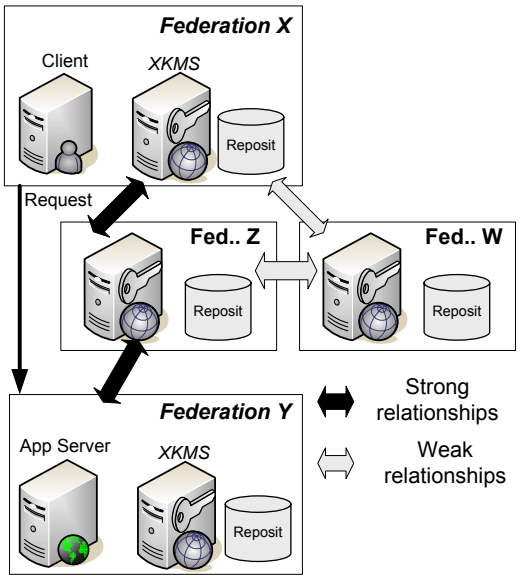

Figure 3. Creating Federation Webs through XKMS

In the scenario illustrated in Figure 3, *client A*, which belongs to *federation X*, wishes to access an application server in *federation Y*. *Federation* X's XKMS service is associated with the XKMS services of *federations W and Z*, and *Z's* XKMS service, in its turn, is associated with *federation Y's* XKMS service. In this example, *client A* may ask for the aid of the XKMS service of its *federation X*, so that the service accomplishes the search through the SPKI federation webs. When a chain of authorization certificates is located, the member can negotiate privilege concession with the privilege owner.

In the proposed model, a trust relationship can be of two kinds: weak or strong. The type of trust relationship is defined through the business rule. For instance, a *strong* trust relationship may occur between a federation of airline companies and a federation of hotels. These federations may present strong trust relationships because of common interests - for example, a joint sale of airline tickets and hotel accommodation. On the other hand, federations without many common interests - for example, because they belong to different countries - might establish *weak* trust relationships.

In the proposed management model, the XKMS service associating with another SPKI federation becomes a member of the latter. This association is made effective through the issuance of an SDSI group certificate of associates in each federation involved. The XKMS service is, thus, responsible for maintaining the information concerning the members and associates of its SPKI Federation, removing or adding members and associations with other SPKI Federations, but avoiding conflict of interests. These functionalities of the XKMS service are described in detail in the next section.

4.3 Operations in the XKMS Service

XKRSS: Membership, Association and Certificate Registration Processes

As defined in [19], a principal can affiliate with as many SPKI federations as desired; however, the principal must provide an endorsement in the form of a *threshold certificate*[5] signed by *k-of-n* members defined by this specific federation. To each new member of the SPKI Federation, a new group certificate is issued, expressing participation in the Federation, so that (*membership*) can be proven. Likewise, an XKMS service associates with another federation when providing the endorsement signed by *k-of-n* associates.

In our federated management model, the affiliation process takes place through the XKRSS register operation. As shown in Figure 4, this same operation is used to affiliate members, to submit certificates (that can be delegated) in the SPKI federation repository and to establish associations among federations. Therefore, an XKMS service - upon receipt of a registration request, and prior to identifying which task it should accomplish - verifies whether or not there is a *threshold certificate* or a chain of authorization certificates in the message content.

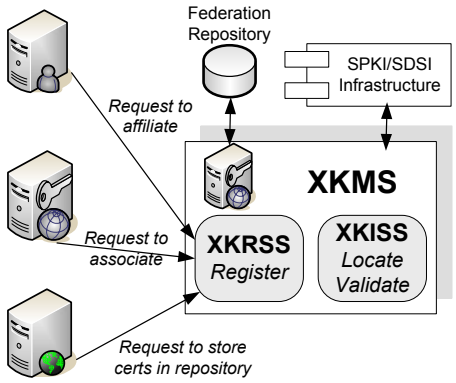

Figure 4. XKMS Service Operations

XKISS: Certificate Locate and Validate in the Federation Webs

The locate operation of an XKMS service is used by the members of the federation and by the associated XKMS services to request a search for chains of authorization certificates in their federation. A member can also request an

[5]The resource's owner can delegate rights over its resource to a threshold subject - which can delegate these rights to other principals. The threshold subject is composed of two numbers: K and N ($0 < K <= N$). This means that at least K principals of N present in this threshold subject have to sign the rights delegation [8].

XKMS service to validate a chain of certificates, for example, authorization ones, using the *validate* operation.

XKMS services provide facilities to discover authorization chains through searches carried out in the webs of SPKI federations. The heuristics for the navigation in the federation webs are defined and described in detail in the following section.

5 Search algorithm – *Diff-Trust*

As stated in section 2, principals affiliate with one or more federations and federations create associations among themselves, in an arbitrary way, aiming to increase their presence on the web. These affiliations and associations result in the creation of a web, known as *web of trust*.

Webs of trust, in a way, share some features with unstructured peer-to-peer networks used for file sharing. These networks present a distributed configuration which does not depend on any centralizing part. Each node of the network only knows of a subset of nodes of the whole network. Thus, the similarity between both networks suggests that unstructured P2P network search algorithms, such as Gnutella [9], can be used to discover certificate chains.

The Gnutella protocol was developed to locate files in a distributed network using message flooding. Previous experiments [13] have already shown that even though it presents positive answers for queries in most cases, the flooding technique is very costly as it generates too many messages in the network. This paper introduces a search algorithm, called *Diff-Trust*, which follows the concept of flooding and which assumes different behaviors according to the level of trust existing among the federations. For each trust relationship, there is an associated weight, which signals the level of trust among the federations taking part in the relationship.

Strong relationships enable a federation to delegate its query to another federation and the latter is held responsible for querying the other federations with which it has trust relationships. Such functioning is identical to the Gnutella protocol; upon receipt of a query, a node verifies whether or not it owns the desired resource. If it does not, it propagates the query to all its neighbors. When relationships are weak, and the queried federation does not have the desired resource, the querying federation receives the list of federations with which it has a trust relationship. Hence, the federation that received the list is responsible for querying the federations on the list. Therefore, it is this federation's role to query the other federations from the list.

Figure 5 shows the path taken by a search through paths composed of weak and strong relationships. In this example, federation"A" wishes to find a certificate chain that connects it with federation "J". Thus, "A" propagates the search to all its neighbors (step 1). There is a strong relationship

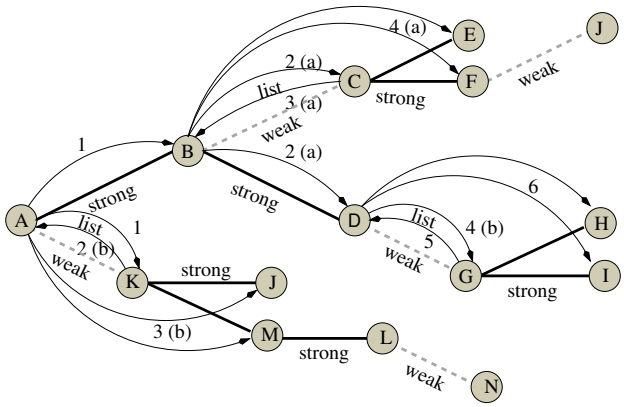

Figure 5. Diff-Trust algorithm behaviour

between "A" and "B"; thus, "B" takes on the role of propagating the search to its neighbors (step 2a). There is a weak relationship between "A" and "K"; thus, "K" returns to "A" a list with the federations with which "K" has trust relationships (step 2b). "A" is now in charge of propagating the search for the federations present in this list, namely "J" and "M" (step 3b). This behavior can also be observed between "B" and "C" as well as between "D" and "G". Finally, when querying "F", "B" receives a positive answer which is passed along the inverse search path until it reaches the federation that originated it, in this case until it reaches "A".

The *weak relationship* also acts as a search depth limiting factor. When the search goes through a path that has a *weak relationship*, this search will only be propagated to one more level, that is, the only federations to be queried are the ones present in the list that was returned by the federation with which a *weak relationship* existed. For example, there is a weak relationship between "A" and "K" and then when "K" is queried, it sends "A" the list of trusted federations (step 2b). "A" then queries the federations present in this list (step 3b), and the search ends there since they do not own the desired resource. In other words, neither "L" nor "M" will propagate the search over its neighbors or even send "A" a list. On the other hand, for paths composed of subsequent *strong relationships*, the depth limit is determined by the *time to live* (TTL)value, which can be defined according to the requirements of each application.

This kind of control is exerted by the federations being queried. When queried, a federation verifies its level of trust with the one performing the query. If the level is weak, the federation that originated the query must also provide the corresponding chain of associated certificates for the search, thus signaling that there is a chain of trust linking the two federations.

The main idea behind levels of trust is to limit the number of messages in the network but without decreasing the efficiency of flood techniques. That is, strong relationships

still work like flood techniques and weak relationships help limit the number of messages in the network.

5.1 Experiments & results

The effectiveness of the proposed algorithm was verified through simulations, whose results were compared with the results obtained by simulating the flooding algorithm, employed by Gnutella. Peersim [17] was the simulator used; it was designed to operate on large-scale simulations of P2P networks.

In the webs of trust present in this work, each federation has a partial view of the whole network. A web of trust can thus be represented as a graph, where the federations correspond to the vertices (nodes) and trust relationships correspond to the arcs. In this study, the federations establish mutual trust relationships, that is, if "A" trusts "B", then "B" trusts "A", which results in a non-directional graph.

The topology of P2P networks heavily influences the effectiveness of various algorithms. The topology is determined by the number of neighbors each node knows about (the degree of a node), and in our study we consider one class of topology: *scale free* [2]. The *scale free* topology follows the *power law* distribution where many nodes have few connections and few nodes have many connections. Such distribution complies with the concept of *small world* [14], which has already been observed in different areas, webs of trust included [5], a fact that brings the simulation environment closer to a real environment.

The simulations were carried out in a network composed of 20.000 nodes, labeled from 0 to 19.999. The minimum degree of the graph was set on 2; the average degree was set on 4. In other words, the minimal number of trust relationships that a federation is allowed to have is 2, and the average of trust relationships is set on 4. Weights were associated with the arcs so as to represent weak and strong relationships. The choice for the weight associated with each arc followed a pseudorandom distribution. The network, where the simulations were made, owned 39.996 arcs, where 21.635 are weak relationships and 18.361 are strong relationships.

Our interest lies in finding out how successful the search is, how many positive answers it returns, and how many messages it uses. The search always starts from node 0 and is propagated to other nodes, according to the behavior of the algorithm in question (*Diff-Trust* or *flooding*). As soon as one finds a chain of certificates linking the node that originated the query to the target node, the query answer is sent through the inverse path that had been taken by the search. For the target node, a choice was made to carry out simulations for three different cases, considering the distance until the source node of the query: the closest, distance being 2; the furthest, distance being 6, and average distance being 4.

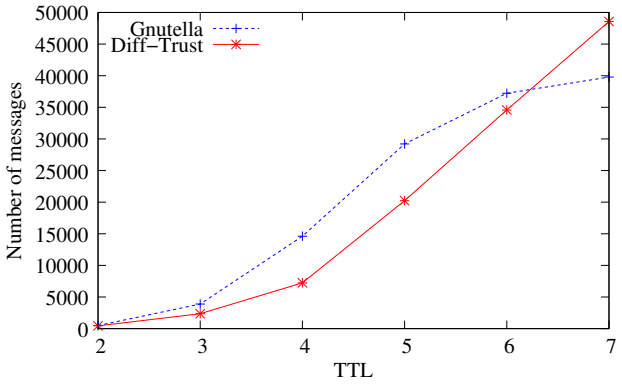

Figure 6. Number of messages under different TTL values

For all the cases simulations were carried out by having the initial TTL set to 2 and incremented up to 7 (the Gnutella specification standard value is $TTL = 7$). The graph in Figure 6 shows the amount of messages that were propagated along the network by each of the algorithms (y-axis), taking into account the TTL value(x-axis).

As regards the amount of positive answers obtained, the two algorithms presented pretty close values; neither algorithm reached answers when $TTL = 2$. Figure 6 shows that in most cases the *Diff-Trust* algorithm, proposed in this paper, presented better results than the traditional flooding algorithm, except for the case when $TTL = 7$, where the total number of messages propagated in the network was bigger.

At first, *Diff-trust*'s behavior seems to generate a larger amount of messages than Gnutella because nodes (with weak relationships) will send the answer – providing the list of neighbors – even if they do not own the resource (see section 5). In Gnutella, a node will only provide an answer if it owns the resource. However, the algorithm compensates for this extra exchange by limiting the search that passes through a path that has a weak relationship.This behavior complies with the principles of trust, i.e.: the further you are from me, the less I trust you. In Webs of trust, the distance between two principals could be measured by the amount of intermediate principals.

Taking into account the amount of messages sent *vs* the amount of answers obtained, one can conclude that the *Diff-Trust* algorithm showed the best performance; it was possible to find positive answers propagating a smaller amount of messages in the network.

6 Implementation

In order to prove the proposed model's flexibility as well as feasibility of use in distributed applications based on

a service-oriented architecture, a prototype involving the model was defined and implemented. In this prototype, we made use of the *Apache Axis framework* and the *Apache WSS4J* and *Apache XML Security* libraries.

The XML Signature specification [3] presupposes that the `<ds:KeyInfo>` element contains a subelement named `<ds:SPKIData>`, which must be used to transmit key pairs, as well as certificates or other data associated with SPKI. Nevertheless, this specification does not define how these SPKI data are inserted in this element. Thus, this paper proposes an extension to the *XML Signature Schema* so as to define which elements of the SPKI/SDSI infrastructure can be inserted into an XML signature. The `<ds:SPKIData>` element contains the `<ds:SPKISexp>` element which then contains the subelements, defined in Table 1. These extensions were implemented in the *XML Security* library, following the recommendations for extensions defined by the *Apache Foundation*.

`KeyValue`	Public key value
`tag`	Real expressions that can transmit authorizations
`uris`	URI List
`authorization-cert`	SPKI Authorization Certificate.
`name-cert`	SPKI Name Certificate
`sequence`	SPKI Authorization Chain

Table 1. Subelements of `<ds:SPKISexp>`

All the XKISS and XKRSS operations, defined in the proposed model - including the search algorithm in the federation web - were implemented. The XKMS specification defines different processing modes among XKMS services and their clients. In this work, for all the message exchanges among the XKMS services and between a member and a service, a two-phase protocol with asynchronous processing — defined in[10]– was implemented.

7 Related Works

Most studies described in the review of the literature — which use an XKMS service — resort to PKI X.509 only, thereby inheriting the problems of a centralized authentication approach. Among these studies, one can highlight the certificate validation service for computational *grids* presented in [16]. This service introduces a certificate validation module (CVM), using the XKMS service, in which several protocols are used, such as OSCP (*Online Status Certificate Protocol*), SCVP (*Standard Certificate Validation Protocol*) and LDAP(*Lightweight Directory Access Protocol*). On certificate validation tests, CRL verification must also take place. Another relevant study described in [4] presents a health information *grid* known as *HealthInfo Grid* which owns a component named *Medical Exchange Agency* (MEA). This component makes use of PKI X.509 and takes on the role of certification authority (CA) for the remaining *grid* components and for each component, MEA issues a web certificate. All the management of these certificates is done through the XKMS interface. Other studies which focused on XKMS, along with PKI X.509, to provide facilities for service-oriented applications are [12, 1, 7]

The studies mentioned above prove the relevance in using the XKMS service. However, only X.509 is taken into consideration in all those studies. Every XKMS implementation, independent of a distributed application, for example *SQLData XKMSServer*[6], *Markup Security Project*[7],*XKMS Prototype Server*[8] and the implementation of Verisign[9] do not support SPKI/SDSI. The prototype described in this study is the first implementation of XKMS to manipulate SPKI/SDSI certificates.

The SPKI/SDSI infrastructure requires a management model so as to aid the principals to discover an authorization chain required for the access to a given resource. In [19], a management model was described; however, in this model there is an overload on the client, which not only has to understand the complexity of PKI, but also takes responsibility for locating chains of certificates. Besides, the access to the functionalities provided by the model does not follow a standard exchange message model. The proposal described in this article encompasses the management model proposed in [19]. Nevertheless, the XKMS service, in lieu of the certificate manager, takes responsibility for the dynamic discovery of the trust paths. The solution for the use and federated management of SPKI, through a Web Service, provides the standardization for the access to the functionalities offered in an SPKI Federation.

8 Conclusion

The ability to define groups and to delegate authorizations, as well as the facilities to develop security, scalable, distributed systems, make SPKI/SDSI a good option for the development of distributed applications based on webs of trust. This study presented an approach to integrate the SPKI/SDSI trust model into the standards of Web services and XML extensions. From the concept of SPKI/SDSI Federations, the proposed model provides the federated management of SPKI certificates, through service-oriented technology, via XKMS. The discovery of certificates attributed to the XKMS service spares the client this costly task. Because this model provides an XKMS Web Service, it

[6] http://www.sqldata.com/xkms.htm
[7] http://markupsecurity.com/info/xkms/index.htm
[8] http://www.wingso fhermes.org/xkms.html
[9] http://www.xmltrustcenter.org/xkms/index.htm

presents a standard protocol for the access to the management and trust establishment functions.

In the literature, there is no attempt to use XKMS together with SPKI. Thus, this article proposes an original model. The lack of competing approaches resides in the fact that although the XKMS specification owns elements in which the use of different PKIs is possible, in practical terms, the specification was totally defined aiming at a hierarchical PKI such as X.509. Integrating XKMS into the SPKI trust model was not a simple task; however, there was no modification or extension that could compromise the conformity with the XKMS specification [10].

Acknowledgements

This work has been developed within the scope of the "Security Infrastructure for Service Oriented Distributed Applications" project (CNPq 550114/2005-0). The authors would like to thank the financial support received and the members of this project for their contributions.

References

[1] C. Adams and S. Boeyen. Uddi and wsdl extensions for web service: a security framework. In *Proceedings of the 2002 ACM workshop on XML security*, pages 80–89, November 2002.

[2] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.

[3] M. Bartel, J. Boyer, and B. Fox. *XML-Signature Syntax and Processing*. W3C, February 2002. `http://www.w3.org/TR/xmldsig-core`.

[4] I. Bilykh, Y. Bychkov, D. Dahlem, J. H. Jahnke, G. McCallum, C. O. A. Onabajo, and C. Kuziemsky. Can grid services provide answers to the challenges of national health information sharing? In *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, pages 01–15, 2003.

[5] S. Capkun, L. Buttyan, and J.-P. Hubaux. Small worlds in security systems: an analysis of the PGP certificate graph. In *Proceedings of the 2002 New Security Paradigms Workshop*, pages 28–35, September 2002.

[6] D. E. Clarke. Spki/sdsi http server/certificate chain discovery in spki/sdsi. Master's thesis, Massachusetts Institute of Technology - MIT, September 2001.

[7] R. T. Daniel J. Polivy. Authenticating distributed data using web services and xml signatures. In *Proceedings of the 2002 ACM workshop on XML security*, pages 80–89, November 2002.

[8] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. *SPKI Certificate Theory*. IETF RFC 2693, September 1999.

[9] Gnutella. *The Gnutella Protocol Specification v0.4*. Clip2, 2001.

[10] P. Hallam-Baker and S. H. Mysore. *XML Key Management Specification (XKMS 2.0)*. W3C – Proposed Recommendation, May 2005.

[11] T. Imamura, B. Dillaway, and E. Simon. *XML Encryption Syntax and Processing*. W3C, December 2002. `http://www.w3.org/TR/xmlenc-core`.

[12] R. Kraft. Designing a distributed access control processor for network services on the web. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):36–52, 2002.

[13] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. *Proceedings of the 16th international conference on Supercomputing*, pages 84–95, 2002.

[14] S. Milgram. The small world problem. *Psychology Today*, 1:61, 1967.

[15] OASIS. *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. Organization for the Advancement of Structured Information Standards (OASIS), March 2005. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`.

[16] N. Park, K. Moon, and S. Sohn. Xml security: Certificate validation service using xkms for computational grid. In *Proceedings of the 2003 ACM workshop on XML security*, pages 112–120, October 2003.

[17] Peersim p2p simulator, 2004. `http://peersim.sourceforge.net`.

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

[19] A. O. Santin, J. da Silva Fraga, F. Siqueira, and E. R. de Mello. Federation web: A scheme to compound authorization chains on large-scale distributed systems. In *22nd Symposium on Reliable Distributed Systems (SRDS'03)*, Florence - Italy, 2003.