Federation Web: A Scheme to Compound Authorization Chains on Large-Scale Distributed Systems*

Altair Olivo Santin[1,3], Joni da Silva Fraga[1], Frank Siqueira[2], Emerson R. de Mello[1]

[1]Department of Automation and Systems (DAS) and [2]Department of Computer Science (INE)
Federal University of Santa Catarina (UFSC), Brazil
{santin, fraga, emerson}@das.ufsc.br, frank@inf.ufsc.br

[3]Graduate Program in Applied Computer Science – Exact Sciences and Technology Center
Pontifical Catholic University of Paraná (PUCPR), Brazil

Abstract

Traditional security systems are not easily scalable and can become single points of failure or performance bottlenecks when used on a large-scale distributed system such as the Internet. This problem occurs also when using a Public Key Infrastructure (PKI) with a hierarchical trust model. SDSI/SPKI is a PKI that adopts a more scalable trust paradigm, which is focused on the client and based on authorization chains. However, the task of locating the chain that links a client to a server is not completely addressed by SDSI/SPKI. Aiming to overcome this limitation, this paper proposes extensions to the SDSI/SPKI authorization and authentication model. The proposed approach introduces the concept of Federation Webs, which allow the client to build new authorization chains linking it to a server when a direct path does not exist. A prototype implementation of this proposal has shown promising results.

1. Introduction

Internet applications require authentication and authorization models in which the trust relationships can be established on a flexible, scalable, and distributed way. However, in traditional authentication and authorization system, while authorization service is generally implemented in a distributed way, the authentication service is centralized by the naming service. This approach limits the use of these mechanisms to the local naming domain, and consequently it is usually adopted in corporate networks. When employed in a large-scale distributed system such as the Internet, this approach grows in complexity, leading to poor scalability, and eventually creating single points of failure or performance bottlenecks. In order to overcome the scalability limitations, it is necessary to define inter-domains trust relationships, allowing the coverage of a global naming space. Under such circumstances, the management of these relationships may become a difficult task.

Public Key Infrastructures (PKI) offer means to carry out authentication on a global context. However, the most commonly used PKI are unsuitable for building trust relationships between principals spread through a large-scale distributed system.

The *X.509* PKI [1], for example, adopts a global naming system (X.500), which is based on a hierarchical trust model formed by Certification Authorities (CA). In this model, the authentication chains start from a root CA and lead to a principal (a user, for example). Although the X.509 PKI is widely used, its global model faces difficulties on adjusting to each country's legislation, and is difficult to use due to its complex and inflexible scheme. In addition, trust models based on a centralized entity (names / authentication service), besides representing critical points regarding faults and vulnerability, may constrain performance and scalability on large-scale environments [2].

Pretty Good Privacy (PGP) [3], employed to cipher and authenticate computer files and electronic mail, adopts a structure for key and certificate management based on a web of trust. Comparing to the X.509 hierarchies, the PGP *web of trust* – built up on an arbitrary way – is quite flexible and very well adapted to the characteristics of the Internet. On the other hand, the use of pondered trust decisions implies that multiple signatures may be necessary on a single certificate to assure credibility, leading to a complex authorization process.

* This project has been partially supported by the Brazilian Research Council – CNPq, under grant 552.175/2001-3.

Proceedings of the 22nd International Symposium on Reliable Distributed Systems (SRDS'03)
1060-9857/03 $17.00 © 2003 IEEE

This paper introduces a new approach to establish trust relationships for authentication and authorization in large scale distributed systems. The proposed approach makes use of the *Federation* notion, responsible for a certificate repository as well as for establishing trust relationships with other federations spread through the distributed system. Federations define domains in which trust relationships between principals are valid, creating *Federation Webs*. Therefore, in the absence of a given authorization chain, principals can locate certificates in the federation web and negotiate privileges in order to create new authorization chains.

This paper is structured as follows: section 2 overview trust management and describes systems that adopt this paradigm, giving particular attention to SPKI/SDSI, which will be employed in this work; section 3 describes the proposed model for building trust relationships between principals and explains how new authorization chains can be established; section 4 presents a prototype implementation of this model and evaluates its performance; and finally, section 5 outlines the conclusions resulting from this work.

2. Trust Management

Trust management is defined as a paradigm focused on authorization [4], unifying security policies, identification, access control and authorization in a single framework. This framework is based on an egalitarian trust model, which has the main purpose of adapting authentication and authorization models to the distributed worldwide network environment – i.e. the Internet.

Two different approaches are found in the technical literature that can follow this concept. In the first one, the trust management is based on a language for authorization and credentials description and on a compliance checking engine. *PolicyMaker* and *KeyNote* [5] are systems that use this approach.

The concept of trust management can also be implemented using a standardized information structure, which allows the description of credentials for authorization and security policies. The *Simple Distributed Security Infrastructure / Simple Public Key Infrastructure* (SDSI/SPKI) is a good example of this approach.

The SDSI/SPKI infrastructure employs an egalitarian trust model: principals are not only identified by public keys, but they can also sign and issue certificates. As a result, there is no need for a Certification Authority like the one existing in the X.509 environment. This approach eliminates the limitations and the complexity presented by X.509 resulting from the adoption of a global naming scheme.

SDSI and SPKI were developed separately, but have been combined due to their complementary features. SDSI [6] is a security infrastructure whose main purpose is to simplify the implementation of secure distributed applications. SPKI [7], on the other hand, is the result of an effort to design a simple and well-defined authorization model. After being merged, SPKI and SDSI provide, mainly, a simple infrastructure for authentication and authorization of distributed applications in large-scale environments.

2.1. Authentication and Authorization using SDSI/SPKI

Two different kinds of certificate are defined by SDSI/SPKI: a name certificate and an authorization certificate.

A name certificate links names to public keys or to other names. A principal always signs certificates issued by one with his/her private key. Names described on a name certificate are meaningful and unique only within the namespace of its issuer – i.e. they can have different meanings in other namespaces. The concatenation of the public key of the issuer with a local name is recognized as a SDSI/SPKI unique global identifier.

Names and naming chains are used by SDSI/SPKI to simplify the search for the actual identifiers: the public keys of principals. In order to resolve names, the whole chain must be examined until the corresponding public key is reached. This procedure is known as "naming chain reduction".

SDSI/SPKI authorization certificates grant access permissions to a name, to a special group of principals – called "threshold subjects" – or to a public key. Through these certificates, the issuer can authorize principals (i.e. other public keys) to access a resource or a service that it provides or controls.

Authorization certificates held by a principal may also be delegated to other principals. A "public" authorization certificate (with the delegation bit on) allows a principal not only to access the resource but also to delegate (grant) access to other principals – either as a whole or partially. Otherwise, when the delegation bit is off, the received privileges cannot be forwarded. In such case, the authorization certificate is "private", i.e. only the principal holding the certificate can use it [8].

For the access control procedures, the rights granted through consecutive delegations (authorization chains) must be summarized into a single certificate containing the intersection of all the privileges granted to that

subject, in a procedure called "authorization chain reduction".

The trust model adopted by SDSI/SPKI is said to be focused on the client due to its authorization procedure. According to this procedure, authorization chains are built through the delegation of access privileges, ensuring trust paths between a server and its clients. The authorization flow is illustrated by Figure 1, in which client *A* is receiving an authorization certificate (C_{SA}) from server *S* and delegating this privilege to client *B* (C_{SA}+C_{AB}). Clients *A* and *B*, after receiving their certificates, have authorization chains that allow them to access server *S* – in most cases, these authorization chains are built arbitrarily. The privilege owner must keep the corresponding certificate and present it to the server when accessing the protected resource. In this example, client *B* (Figure 1) has to present a signed access request and the corresponding authorization certificate chain (C_{SA}+ C_{AB}) for having access to the resource.

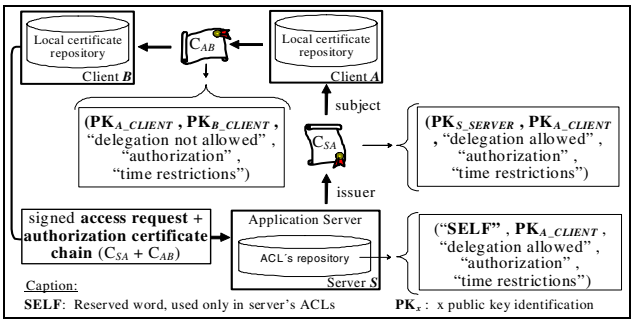

Figure 1 – SDSI/SPKI Authorization Flow

2.2. Discovery of SDSI/SPKI Authorization Certificates

The main difficulty faced by a principal (client) using SDSI/SPKI is to find an authorization chain which certifies that it has authorization to access an object or a service in the distributed environment. Several architectures and algorithms have been proposed in the technical literature to help a client to search for a certificate chain. However, none of these proposals offers alternatives to the client when a valid certificate chain is not found (i.e. the search for a certificate chain is unsuccessful).

In [9], Nikander and Viljanen have shown how the DNS service can be used for storing and retrieving SDSI/SPKI certificates. In that proposal, DNS extensions added by RFC 2065 have been used to allow the storage of certificate records by using entities that store identification and authorization certificates in DNS databases. In addition, the search algorithms include some filtering of the certificates being retrieved.

In [10], Aura describes the trust net built by the propagation of SDSI/SPKI authorization certificates as an oriented graph. The author considers that, in typical corporate environments, such graph is hourglass-shaped, due to the fact that there is much more client and server keys than intermediary keys. Therefore, starting from these premises, the author uses the DFS forward and backward algorithms, and their combination, to perform fast searches in a database having only one intermediary. Experiments using the distributed search algorithms proposed in [10] are reported in [11]. This work also reports some improvements in the DFS forward algorithm.

All previously described works have been conceived for preliminary versions of SDSI/SPKI, in which some aspects of the model still had not been solved. Some premises assumed at that time are now considered obsolete, no longer complying with the RFC 2693 specifications. However, these papers have valuable contributions in terms of system architecture.

The chain search algorithms and other aspects considered in [12], suggested by Clarke, are deeper refinements of RFC 2693 recommendations. The author of this work also presents an implementation of the current version of SPKI, quite rich in content, although no architectural solution for distributed systems has been proposed.

In [13], Li argues that SDSI/SPKI local names can be viewed as distributed groups of principals for name resolution. Based on this assumption, the author proposes algorithms based on logic programming, supposed to be more efficient in chain search when compared to conventional implementations. Since the main purpose of this paper was to define search algorithms based on logic programming, a new architecture has not been proposed. Nevertheless, the interpretation of local names as distributed groups can be considered a significant contribution.

3. A Proposal of Extensions to the SDSI/SPKI Trust Model

This section presents the proposed extensions to the SDSI/SPKI trust model, which allow building new authorization chains. The proposed trust model is based on the concept of Federations, which group principals with common interests. A federation assists its members on reducing names and on building new authorization chains.

By joining a federation, principals have access to the federation facilities, and new trust relationships among these principals can be established. In this sense, the SDSI/SPKI trust model is supported by a Certificate Manager (CM), which offers a certificate search alternative, either for reducing names or for creating new authorization chains.

Figure 2 shows a federation CM integrated to the SDSI/SPKI trust model, in which client *A* stores its public certificates in the federation certificate repository. Through a search on the CM repository, client *B* – which has no access to server *S* – can identify a principal in the federation (client *A*) holding such privilege (C_{SA}). Client *B* can then negotiate with *A* in order to receive (by delegation) this privilege (the authorization chain C_{SA}+C_{AB}).

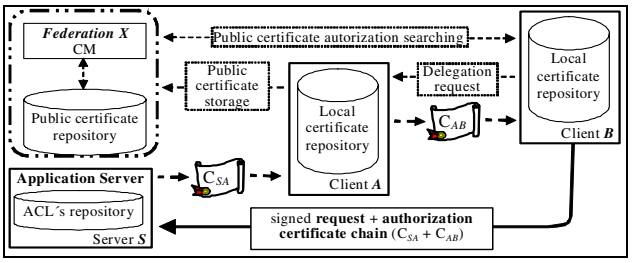

Figure 2 – SDSI/SPKI Extended Trust Model

The presence of a client at distinct federations allows this client to easily access the public authorization certificates held by members of these federations. However, the number of federations a client must join in order to have an acceptable visibility in the worldwide network can also be considered a scalability problem. The scalability requirements are achieved in the proposed model by associating federations. Certificate managers can then be associated to each other, linking those who, due to existing affinity, can better represent the needs of their members. Such associations are done through trust relationships which form Federation Webs (in Figure 3, for example, the CM of federation *X* is associated to the CM of federation *Y*). This approach frees clients and servers from joining a considerably large number of federations to achieve global scope.

Figure 3 illustrates how the entities that comprise a federation web are organized. Client authorization certificates – both private and public – are stored in a local repository under the responsibility of an agent that represents this principal in its local domain. Clients make name certificates issued by their corresponding principals and their public authorization certificates available through the CMs of the federations they belong. The certificates available through CMs are used in the search for potential issuers of delegable permissions.

The proposed trust model has no centralized entities or hierarchical arrangement, i.e. federation webs are arbitrarily formed, and do not play any active role in the authorization chains – they just carry out support roles in the authorization procedure.

A federation is basically composed of three entities: clients, servers and a certificate manager, which will be explained in the following topics.

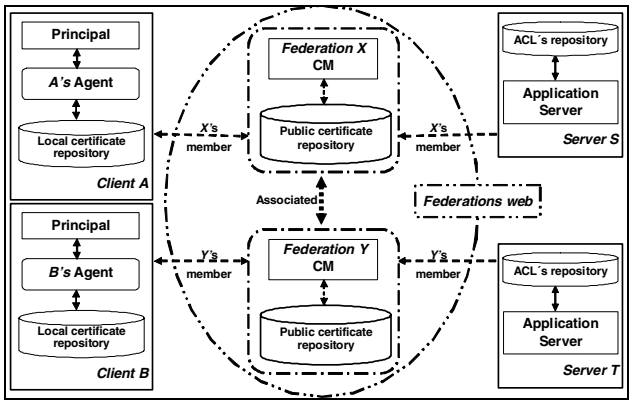

Figure 3 – Federation Web Overview

3.1. Clients and Application Servers

The client represents the principal who creates name certificates, propagates the authorization certificates by delegation, takes part in threshold certificates, requests access and composes new chains.

The storage and retrieval of certificates in the client naming space is responsibility of the client's agent (Figure 3). This agent is a software entity that manages the certificates available at the local repository. These tasks include checking and effecting signatures, searching for certificate chains, negotiating permission grants, issuing new authorization certificates and maintaining local names consistency. The agent must be instantiated during the client's lifetime, and it interacts with the client through a binding to its operational interface.

The application server implements the service objects, which are protected by SPKI ACLs, kept by a guardian (reference monitor). In order to perform delegations and negotiations to propagate permissions, the server can also make use of an agent. In the certificates reduction procedure, the server can issue authorization certificates to clients that present new delegation chains and/or include the public keys of these clients in the guardian's ACLs.

3.2. Certificate Manager

The main purpose of the certificate manager is to facilitate the interaction between clients and servers. A certificate manager only serves the principals that belong to its federation. The public keys of its members are seen as a SDSI group [7]. As the CM does not actively participate on any authorization chain, it is not considered a public key, and therefore is not seen as a principal – it is mainly a repository of certificates.

In order to any ordinary principal join a federation, an endorsement in the form of a *threshold certificate* is demanded [14]. The threshold certificate signature depends on "*k* out of *n*" federation members. Each federation defines the number of members (*k*) that must sign the endorsement request.

Upon joining a federation, the principal's name certificate is included in the federation repository. The federation's certificate manager will store name certificates in order to ease principal identification (this procedure will be explained in more detail in section 3.3).

For every new member joining the federation, a name certificate stating SDSI group inclusion it is issued in order to prove membership. The creation of associations among federations (federation webs) is also interpreted as membership of the SDSI groups of each federation involved. In this case, the new member – i.e. the other federation – is recognized as a group defined and administered within another naming space, according to the definition of SDSI groups.

Besides managing the information related to the members and associations of its own federation, the CM has the ability to include or exclude members and associations to other federations, observing any conflicts of interest. Procedures for storing and retrieving name and authorization certificates are made available to federation members through standard interfaces offered by the federation CM.

3.3. Authentication, Authorization and Auditing

The authentication of SDSI/SPKI principals is performed using public keys instead of names. The authentication of principals is done by checking their digital signatures. In order to check the digital signature on the destination, the principal's public key must arrive there securely. Since there is no entity responsible for public key distribution in the SDSI/SPKI infrastructure, the public keys demanded by an authentication procedure are available through authorization certificate chains.

Mutual authentication is achieved with SDSI/SPKI on an authorization chain basis. The client making a request to a server must sign it and send it along with the authorization chain that grants the required access privileges. The authorization chain sequence associated to a request is checked by the resource guardian upon its arrival. The guardian makes use of the last key in the authorization chain (the client's key, in the subject field) to check the digital signature on the request. Having this check been successful, then client's authenticity is confirmed.

Every authorization certificate carries the public key of the principal signing that certificate (the issuer field). Therefore, to authenticate a server (always expressed as a public key starting an authorization chain), the client should require the server's name certificate, retrieved from a federation web. After that, the client uses the certificate's public key for validating the server's signature in the first chain's authorization certificate. When all the mentioned procedures are successfully done, then the server identity can be assured.

All accesses by public keys to the server are locally logged, and these log records can be used for auditing purposes. If needed, the searching of the corresponding name certificate can be performed on the federation web to identify the principal corresponding to the public key that performed a given access.

The whole authentication and authorization procedure described in this paper is in compliance with the current SDSI/SPKI specifications.

3.4. Creation of New Authorization Chains

There are several experiences in the technical literature regarding procedures for searching SDSI/SPKI certificates, such as the ones that were presented in section 2.2. However, in all these approaches, if a certificate chain is not found, the search reports an exception (failure), and the client is unable to access the server. This work proposes a schema based on the use of federations that enables a client to locate in a federation web a certificate holding the needed authorization. Later on, the client can negotiate with the privilege holder such grant to build an authorization chain that allows one to access the server.

In order to show the chain creation process, consider the example illustrated by Figure 3. At first, an authorization certificate is stored in the CM of federation *X*, after been propagated from the server *S* to the client *A* (*A* is a member of the federation *X*).

Figure 4 shows the messages exchanged between a client and two associated federations when an

authorization chain between client *B* and server *S* does not exist. Client *B*, member of federation *Y*, starts by requesting an access to server *S* (message *m1*). Server *S* replies by sending a challenge message back to *B*. In this challenge message (*m2*), server *S* reports the ACL protecting the requested object and asks for client *B* to prove that there is an authorization chain allowing the requested access. In this case, SDSI/SPKI ACL data is effective to accelerate the searching process.

Having the ACL, *B*'s agent performs a local search for an authorization chain allowing the requested access that links client *B* to server *S*. This search must retrieve all the authorization chains that include the required permission, and have the requested server *S* as the issuer. Supposing that the local search turns to be unsuccessful, *B*'s agent asks the CM of the federation it belongs (*Y*) to search for authorization certificates holding the required rights for accessing server *S* (message *m3*). The attributes considered in the search are the required permissions and the public key of server *S*.

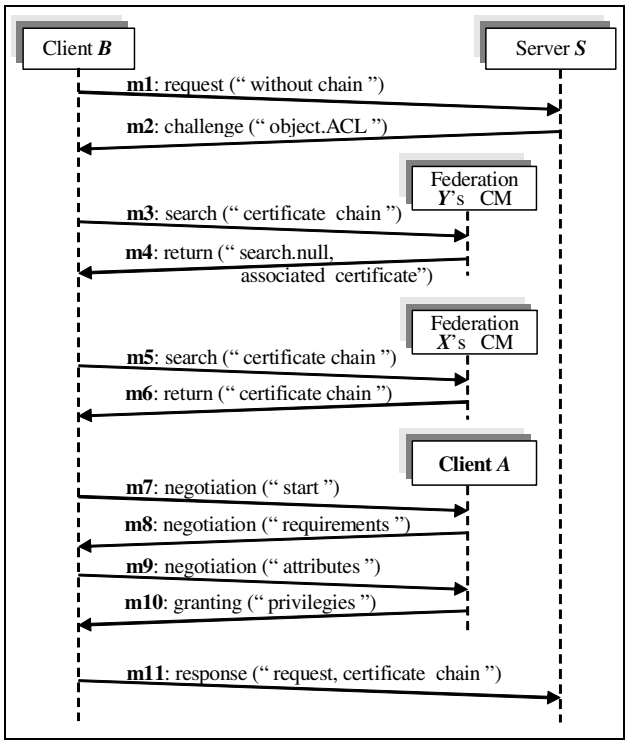

Figure 4 – Messages Exchanged to Compound the Authorization Chain

In the case considered in Figure 4, the search does not result in any authorization chain. In this situation, the CM of federation *Y* returns to client *B*, as a result of the search, member certificates [1] of the associated federations – i.e. client *B* receives the member certificate of federation *X* – so that it can contact the associated federations (message *m4*).

Having the certificates of associated federations, client *B* contacts the CMs of these federations. Message *m5* corresponds to the queries on federation *X* in the considered example. In message *m6*, client *B* receives as return from the CM of federation *X* a chain – the authorization certificate with the access permission granted by server *S* to client *A* (C_{SA}). Then client *B* sends to the permission holder – i.e. client *A* – the request for delegation of access rights (message *m7*). The grant of permissions can be carried out in a simple way, because both the client and the rights holder belong to the same federation, for example. However, depending on the application semantics, more complex negotiations may be demanded. Figure 4 represents this situation: the requested rights holder notifies client *B* about a set of requirements for granting the permission (message *m8*). The client gathers the demanded requirements and sends them to client *A* (message *m9*). Once the application requirements are satisfied, the rights holder issues a certificate granting permissions to client *B* (C_{AB}) and sends it on message *m10*. With this last message, the chain compounding process is concluded, and client *B* can now answer the challenge proposed by server *S* sending the response message *m11*.

3.5. Case Study: Internet Commerce Application

In this section is depicted a scenario to illustrate the usage of federation webs, which synthesizes the proposed schema. This scenario is built upon a Web-based sales application, which illustrates access privileges location and negotiation. One should notice that the proposed schema is quite general and can be applied to distinct situations.

In order to understand the example, first consider a credit card operator (*CC*) and a banking institution which have a business agreement that allows electronic financial transactions to be performed. Based on this agreement, the credit card institution grants to the bank the right to allow purchases if payments are to be charged to credit cards issued by the credit card operator. The bank, whenever receives an SDSI/SPKI authorization certificate with the delegation flag on, stores it on the CM of the federation it belongs (Bank federation).

[1] A SDSI/SPKI name certificates that state SDSI groups inclusion (the federation is implemented as a SDSI group)

Based on the message exchanged presented in the Figure 4, the steps on Table 1 summarize the actions showed in Figure 5 for the purchase transactions example implemented by an Internet-based application. In order to monitor the "allow purchase" privilege delegations, the credit card operator receives a copy of all paid purchase bills from the internet site server *S*.

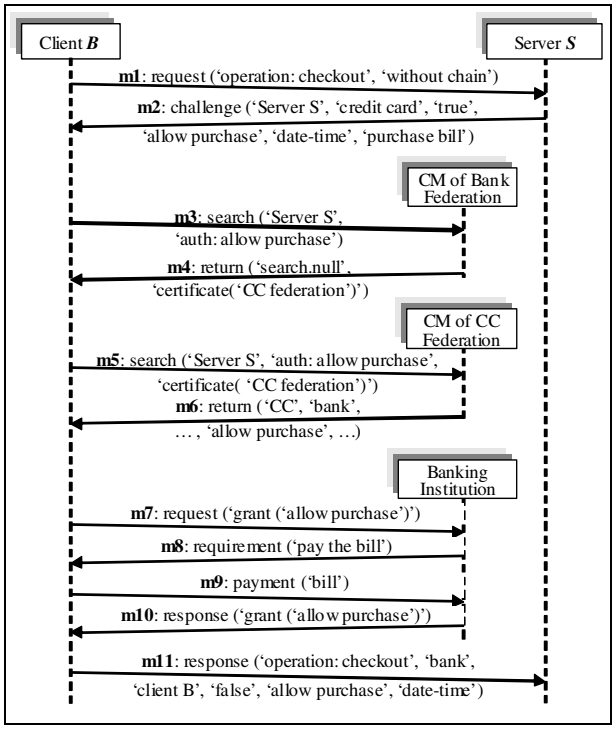

Figure 5 – Messages Exchanged by the Web-based Sales Application during a Purchase

In the scenario described above, no authorization chains exist linking the credit card operator to client *B*. However, the scheme proposed by the federation web model allowed to dynamically and automatically creating the requested authorization chain, in order to complete the purchase operation on the site server *S*. Of course, if the chain holding the requested authorization was not found in the *CC* federation's CM, the search would continue on the associated federations until an appropriate chain was found. It should also be noticed that the server ACL does not have an entry for client *B* allowing it to access the services. Therefore, it is no longer required to register the clients on the server ACL to allow their access to the services. Consequently, all clients' private information is stored only in those institutions with which they have direct relationships. In the example above, the client can pay for the purchase not only if it is a credit card customer – but also being only an ordinary bank customer. By doing so, no credit card numbers or other client-related information is transmitted through the network. Also, the entire client's information is stored only by its banking institution.

Message	Action description
m1	Client *B* navigates through the web pages offered by internet site server *S*. After selecting some items to purchase, client *B* proceeds to checkout.
m2	Server *S* sends back to the client a message containing the purchase bill and a challenge: the principal holding the privilege "allow purchase" is *CC* - requiring from the client the authorization chain issued by *CC*.
m3	Client *B* queries its local repository and finds no chains linking it to C*C*. Then, client *B* sends a chain search message to the Bank federation's CM containing the public key of server *S* and the requested privilege "allow purchase".
m4	CM of bank federation searches in its public certificates repository for the required chain. It sends back to client *B* a "chain not found" message along with the member certificate of associated federation (federation *CC*).
m5	Client *B* sends a query chain search message to *CC* federation's CM (associated federation), which contains the public key of server *S* and the requested privilege "allow purchase".
m6	*CC* federation's CM performs a search on its public certificates repository and finds the required chain. It sends back to client *B* the chain between the server *S* and the banking institution.
m7	Client *B* requests to the banking institution the "allow purchase" privilege delegation.
m8	The bank notifies client *B* that delegating the requested privilege requires paying the purchase bill using one of the payment options.
m9	Client *B* pays the bill using one of the options offered by the bank.
m10	The bank delegates the "allow purchase" privilege to client *B*.
m11	Client *B* sends the authorization chain to server *S*, along with the request in a response message and the server can concludes the purchase transaction.

Table 1 – Description of Messages from Figure 5

4. Implementation and Results

This section presents the prototype implementation of the model described in the previous section. Performance measurements obtained with this prototype are also presented.

4.1. Prototype Implementation

The SDSI/SPKI infrastructure and the policies applied in the model previously described are totally independent from the technology adopted to build it. In this sense, the technologies employed to build the prototype, which are shown in Figure 6, have been highly influenced by the model usage in the Internet – environment assumed as the context of this work.

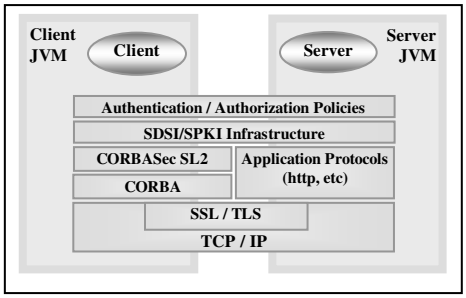

Figure 6 – Technologies Employed to Build the Prototype Implementation of the Mode

The motivation for adopting CORBA as middleware is to take advantage of the services provided by this platform, mainly in aspects related to object lookup (name resolution) and secure remote access invocation. SSL (Secure Socket Layer) was adopted for remote communication. In order to establish a secure channel between a client and a server (holding SSL integrity and confidentially properties), mutual authentication for the principals (client and server) is required. However, since SPKI uses keys as principals instead of names, an operation to translate SDSI/SPKI name certificates into SSL name certificates had to be developed.

The SDSI/SPKI integration with the distributed object middleware was done using CORBASec at application level (CORBASec Security Level 2) [15] implemented by Adiron´s ORBAsec [16] on top of Ionas´s ORBAcus [17] using IAIK´s SSL module [18]. Figure 7 shows such integration.

Security Level 2 is not helpful in structuring security functions at application level. However, in order to make use of the CORBA security model, a minimum set of objects originally present at the ORB level had to be maintained at the application level. These objects, which are show in Figure 7, are: *PrincipalAuthenticator*, *SecurityManager* and *Credentials*.

Figure 7 shows other implementation details. The CM public certificate repository is implemented using Apache Xindice (which stores XML native data) [19]. The CM is implemented as an extension module of the Apache server [20]. All messages exchanged between members and the CM are written in XML. The SDSI/SPKI certificates, originally coded as S-expressions, are translated into XML in our prototype for portability and standardization reasons (XML translations are based on [21]). The SDSI/SPKI *resolver* object shown in Figure 7 is a partial implementation of the client's agent, covering chain searching, local name consistency and digital signature management. Finally, the reference monitor (guardian) is implemented by the SDSI/SPKI *Access Decision* object. The client and server integration onto the prototype environment was greatly facilitated by using plug-ins and applets in the application deployment.

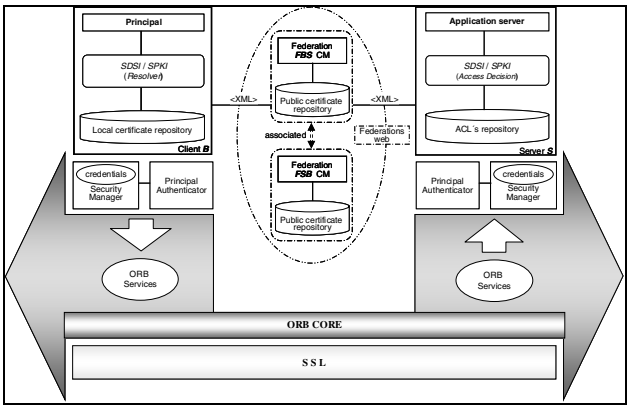

Figure 7 – CORBA-SPKI Integration

4.1 Performance Measurements

Response times of the architecture shown in Figure 7 were measured in order to evaluate the performance of the prototype when the client searches for the authorization certificate chain in three different cases:

- Search for certificates in the client's local repository
- Search for certificates in the client's federation repository, with both located at the same LAN
- Search for certificates in associated federations distributed through the Internet

The application used for performing these measurements behaves like the echo function of the ICMP protocol – i.e., it just copies the input message to the output flow. All the authentication and authorization

mechanisms proposed in this paper were implemented, except the negotiation mechanisms for granting access privileges, which are context-dependent.

Analyzing the average response times obtained with the measurements, for local access (same host) and through the local area network (Figure 8), and for access through the Internet (Figure 9), with message size varying from 256 bytes to 1MB, the following tendencies have been observed:

- For message sizes up to 1KB, the increase in the response time is not significant when the client searches in its repository or in the federation's repository located in the same LAN. Over 1KB, the response time behaves similarly to the measurements obtained over the Internet, where the increase is proportional to the message size, i.e. if the message size doubles, the response time is also multiplied by two.
- The results also show that for local and LAN access, the model adds an average overhead of about 10 times the response time of access using SSL (ORBacus + ORBAsec). In this situation, the overhead is caused basically by the challenge/response protocol and by the search for the authorization chain. On the other hand, the average increase in the response time on the Internet is approximately four times the response time without any cryptography (ORBacus) for messages larger than 256 bytes and tends to decrease until it stabilizes in approximately twice the response time for messages over 8KB.

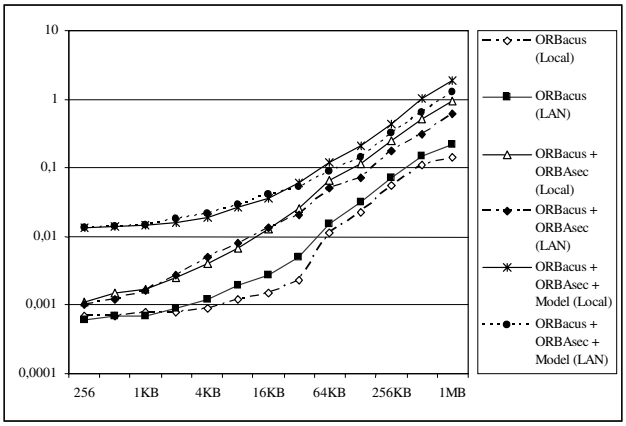

Figure 8 – Average Response Times (in seconds) for Local host and LAN (log graph)

The results shown in Figure 8 were obtained on a test bed composed of PCs with 1.4 GHz Pentium IV processors and 256MB of memory, running Linux operating system, and connected through a Fast Ethernet LAN. Tests over the Internet (Figure 9) employed also a machine linked to the network through a 128Kbps (for upload) ADSL connection in which a federation repository was located.

In a nutshell, one can consider that for local access and when the interacting entities are on the same LAN, the prototype of the model inserts a significant overhead if compared to a secure call only (i.e. a remote method invocation in CORBA using SSL as underlying security service). This happens because in addition to the overhead caused by the use of a secure connection, the prototype implementation exchanges messages in order to implement the challenge/response protocol and to search for the authorization chain. However, due to the short processing time intervals incurred in the overall operation, as can be seen on Figure 8, one can consider that the response time is perfectly acceptable when it is compared to other similar applications.

The overhead caused by the prototype running on the Internet, shown in Figure 9, has an acceptable processing time due to the advantages offered by the proposed architecture if compared to similar mechanisms implemented using classic PKIs.

One can also notice that the results of this experiment can vary significantly according to the semantic of the applications built using the proposed schema.

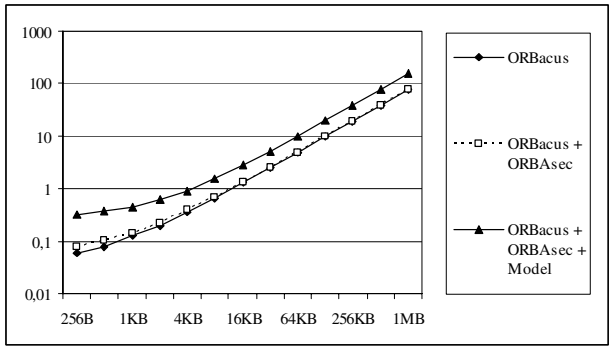

Figure 9 – Average Response Times (in seconds) for Internet Access (log graph)

5. Conclusions

This paper proposed architectural extensions to the SDSI/SPKI authorization and authentication model, allowing the client to build new authorization chains in order to link it to a server when the corresponding path does not exist. This proposal is centered on the notion of federations and on entities called Certificate Managers. The role of certificate managers is to assist in the

construction of authorization chains by locating principals with privileges needed by other principals. As the certificate manager does not participate in the authorization chains, the proposed model can be considered fully decentralized. Thus, the manager does not centralize nor turns hierarchical the relationship between clients and servers, neither it can be considered a critical point regarding faults, vulnerability or performance.

The proposed scheme provides Internet applications with larger flexibility regarding country-specific legal aspects than that offered by the X.509 infrastructure. It is assumed that a client generally negotiates the concession of privileges with a principal belonging to its domain (from the same geographical location or country, for example) and this principal can be inserted in remote domains. Consequently, there is a strong relationship between the client and local principals. In addition, there can be relationship among the local principals and principals from remote domains, so that contexts compatible with the universe of each principal can be defined on an arbitrary way.

The federation web model proposed in this paper frees the server from user account management. It also frees the client from the traditional account creation procedures in order to have access to a server – even in a global context.

The proposed model presents a support to certificate management which allows the creation of new authorization chains. This facility is not observed in any other proposal presented in the technical literature. The proposed scheme is quite flexible and automatic, even considering that in some cases the number of messages exchanged to create a new chain can be expressive.

The prototype implementation of the proposed model shows its effectiveness in current systems integration. In addition, the performance measurements obtained with this prototype can be evaluated positively based on the experimental results presented in this paper.

References

[1] ITU - International Telecommunication Union (1993). Recommendation X.509 – Information technology – Open Systems Interconnection – The Directory Authentication Framework.

[2] HORST, F. W., LISCHKA, M. (2001). Modular Authorization. In: Proceedings of the Sixth ACM Symposium on Access control models and technologies.

[3] GARFINKEL, S. (1995). PGP:Pretty Good Privacy. O'Reilly & Associates, Inc.

[4] BLAZE, M., FEIGENBAUM, J., LACY, J. (1996). Decentralized Trust Management. In: Proceedings of the 17th IEEE Symposium on Security and Privacy.

[5] BLAZE, M., FEIGENBAUM, J., LACY, J. (1999). The KeyNote Trust Management System, Version 2. IETF *RFC2704*.

[6] LAMPSON, B., RIVEST, R. L. (1996). A Simple Distributed Security Infrastructure. [online] available in http://theory.lcs.mit.edu/~cis/sdsi.html, Last access on January, 2003.

[7] ELLISON, C., FRANTZ, B., LAMPSON, B., RIVEST, R., THOMAS, B., YLONEN, T. (1999). SPKI Certificate Theory. IETF RFC2693.

[8] GASSER, M., McDERMOTT, E. (1990). An Architecture for Practical Delegation in a Distributed System. In: proceedings of the IEEE Symposium on Security and Privacy.

[9] NIKANDER, P., VILJANEN, L. (1998). Storing and Retrieving Internet Certificates. In: 3th Nordic Workshop on Secure IT Systems.

[10] AURA, T. (1998). Fast Access Control Decisions from Delegation Certificate Databases. In: proceedings of 3th Australian Conference on Information Security and Privacy.

[11] AJMANI, S. (2000). A trusted Execution Platform for Multiparty Computation. Master thesis. Department of Electrical Engineering and Computer Science of MIT.

[12] CLARKE, D. E. (2001). SPKI/SDSI HTTP Server Certificate Chain Discovery in SPKI/SDSI. Master dissertation. Department of Electrical Engineering and Computer Science of MIT.

[13] LI, N. (2000). Local Names in SPKI/SDSI. In: proceedings of the IEEE Computer Security Foundations Workshop.

[14] AURA, T. (1998). On the Structure of Delegation Networks. In: proceedings of 11th IEEE Computer Security Foundations Workshop.

[15] OMG – Object Management Group (2002). Security Service Specification, v1.8. [online] available in http://www.omg.org/cgi-bin/doc?formal/02-03-11.pdf. Last access on January, 2003.

[16] ADIRON, LLC (2000). ORBAsec SL2 User Guide. Version 2.1.4.

[17] IONA Technologies Inc. (2001). ORBacus User Guide. Version 3.3.4.

[18] IAIK (2000). iSaSiLk 3 - Reference Manual. Institute for Applied Information Processing and Communications (IAIK). Version 3.

[19] STAKEN, K. (2002). Xindice Developers Guide 0.7.1. [online] available in http://xml.apache.org/xindice/guide-developer.html. Last access on January, 2003.

[20] THAU, R. (2002). Design Considerations for the Apache API. [online] available in http://modules.apache.org/reference. Last access on January, 2003.

[21] TERREROS, Xavier O. S., RIBES, J-M. Mas (2002). SPKI-XML Certificate Structure. [online] available in http://www.oasis-open.org/cover/xml-spki.html. Last access on January, 2003.