# Extending the SDSI / SPKI model through federation webs*

Altair Olivo Santin[1,2], Joni da Silva Fraga[1], Carlos Maziero[2]

[1] Federal University of Santa Catarina – DAS/CTC/UFSC
C.P. 476, CEP 88040-900 – Florianópolis – Brazil
{santin, fraga}@das.ufsc.br
[2] Pontifical Catholic University of Paraná – PPGIA/CCET/PUCPR
R. Imaculada Conceição 1155, CEP 80215-901 – Curitiba – Brazil
{santin, maziero}@ppgia.pucpr.br

**Abstract.** Classic security systems use a trust model centered in the authentication procedure, which depends on a naming service. Even when using a Public Key Infrastructure as X.509, such systems are not easily scalable and can become single failure points or performance bottlenecks. Newer systems, with trust paradigm focused on the client and based on authorization chains, as SDSI/SPKI, are much more scalable. However, they offer some difficulty on locating the chain linking the client to a given server. This paper defines extensions to the SDSI/SPKI authorization and authentication model, which allow the client to build new chains in order to link it to a server when the corresponding path does not exist.

## 1. Introduction

In the classic view of authentication and authorization in distributed systems, the naming service centralizes the authentication procedure, restricting its action to the local naming domain. On the other hand, authorization mechanisms are generally implemented in a distributed way. This model, usually adopted in corporate networks, grows in complexity when applied to the whole Internet. 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. The X.509 PKI, 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 can also be

* This project has been partially supported by the Brazilian Research Council – CNPq, under the grant 552175/2001-3.

difficult to use due to its complex and inflexible scheme. In other words, trust models based on a centralized entity (names/authentication service), besides representing critical points regarding faults and vulnerability, may impose restrictions to performance and system's scalability on large-scale environments [1].

Internet applications must be developed taking into account authentication and authorization models in which the trust relationships can be established on a flexible, scalable, and distributed way. The *Pretty Good Privacy* (PGP) mechanism, employed to cipher and authenticate computer files and e-mail [2], adopts a structure for key and certificate management based on a *web of trust*. When compared 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 World Wide Web features. However, choosing such pondered based models leads to difficulties for making trust decisions, as multiple signatures can be demanded in a single certificate for assuring credibility.

On egalitarian trust models – which have the main purpose of adapting authentication and authorization models to the distributed worldwide network environment – i.e. the Internet – the trust management concept has been proposed mainly as a focused paradigm for authorization [3]. The trust management unifies the concepts of security policies, identification, access controls, and authorization.

Two different approaches are found in the technical literature that can follow this concept. In the first one, the trust management is set using a language for authorization and credentials description, and an engine that defines the compliance checker module. *PolicyMaker* and *KeyNote* [4] are systems that use this approach. The concept of trust management can also be implemented using a standardized information structure that allows the description of both credentials stating 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 has been motivated by the complexity of the X.509 global naming scheme. SDSI [5] is a security infrastructure which main purpose is to make the building of secure distributed systems easier. SPKI [6] is the final result of concentrated efforts on a project of a simple and well-defined authorization model. As they have complementary purposes, SPKI and SDSI proposals are combined together, resulting in a unique base for authentication and authorization in distributed applications.

The main difficulty in SDSI/SPKI is to find an authorization chain that certifies a given principal (client) is granted permission to access an object or a service in the distributed system. Several architectures and algorithms have been proposed to help a client to search a certificate chain. However, none of these proposals offers alternatives to the client when a search for a certificate chain is unsuccessful (i.e. no certificate chain is found between the client and the server).

This work presents a new approach for using trust chains for authentication and authorization in large scale distributed systems. The SDSI/SPKI trust model is extended through the *federations* notion, in order to simplify certificate management, as well as to establish new trust relationships in large scale systems. Federations define domains in which there exist trust relationships among principals, providing mechanisms that allow principals to compose global trust relationships.

Therefore, in the absence of a given authorization chain, principals can locate certificates in the federation web and then negotiate the concession of privileges in order to establish a new authorization chain.

This paper is structured as follows: section 2 shortly summarizes SDSI/SPKI; section 3 introduces the proposed extensions to the SDSI/SPKI model; section 4 explains how new authorization chains can be established; section 5 details the prototype implementation; section 6 summarizes related works, and finally section 7 outlines some conclusions.

## 2. Overview of SDSI/SPKI

The SDSI/SPKI defines an egalitarian trust model: principals are public keys that can sign and issue certificates – similarly to a Certification Authority (CA) on an X.509 PKI environment. In the current version of SDSI/SPKI, two different types of certificates are defined: one for names and the other for authorization.

A name certificate links names to public keys, or even to other names. The names described on a name certificate are meaningful only within the naming space of the certificate issuer. The concatenation of the public key of the certificate issuer with a local name represents a SDSI/SPKI unique global identifier. A certificate is always signed with the private key of the issuer. The SDSI/SPKI names and naming chains are used only to ease searching the real principal identifiers: the public keys. When names need to be resolved, the naming chain must be examined in order to reach the corresponding public key. The procedure of resolving the naming chain to reach the real certificate name is called “naming chain reduction”.

The SDSI/SPKI authorization certificates link authorizations to a name, to a special group of principals – called *threshold subjects* – or to a public key. Through these certificates, the issuer can delegate access permissions to other principals (other public keys) in the system.

The SDSI/SPKI authorization certificates can be used for two different purposes. If the delegation bit is off (delegation not allowed), the received privileges cannot be delegated (forwarded). In such case, the subject (principal) should keep the authorization certificate considering it as “private”, i.e. only that principal can use it. If the delegation bit is on (delegation allowed), the subject is holding a “public” authorization certificate, enabling it to delegate (grant) access privileges, which means, keeping them for private use, passing them on to a third party – either as a whole or partially – or both [7].

For the access control procedures, the granted rights through consecutive delegations (authorization chains) must be “reduced / summarized” to only one certificate containing the intersection of all the privileges granted to that subject, in a procedure called “authorization chain reduction”.

Fig. 1 shows the authorization flow on the SDSI/SPKI trust model. Through the delegation of privileges from the application server, authorization chains are built, ensuring trust paths between the server and the clients. In the Fig. 1, clients A and B, after receiving the certificates, will have authorization chains allowing them to

access the server. The authorization chains are usually built arbitrarily. The privilege owner must keep the corresponding certificate and present it to the server when accessing the protected resource. Based on this, one can state that the trust model adopted by SDSI/SPKI is focused on the client.

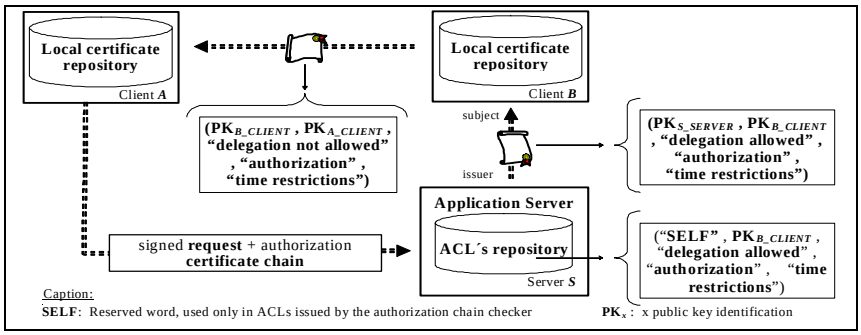


**Fig. 1.** SDSI/SPKI authorization flow (trust model focused on the client)

## 3. A Trust Model Proposal Based on SDSI/SPKI Extensions

This section presents the proposal of an extension to the SDSI/SPKI trust model, which allows building new authorization chains. The proposed trust model is based on the concept of a *federation*, which emphasizes the grouping of principals with common interests. The purpose of a federation is to assist its members on reducing principal names and on building new authorization chains, through the federation's *Certificate Manager* (CM).

By joining a federation, principals get access to the federation facilities and new trust relationships among these principals can be established. In this sense, the SDSI/SPKI trust model is extended by adding a Certificate Manager. The CM offers a certificate search alternative, either for name reduction or for creating new authorization chains.

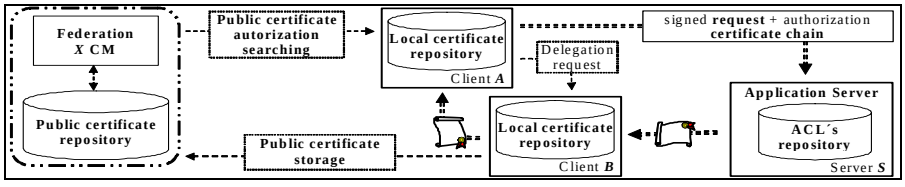


**Fig. 2.** SDSI/SPKI extended trust model

Fig. 2 shows the CM integrated to the SDSI/SPKI classic model. It ensures that client B stores its public certificates in the federation certificate repository. Through a search on the CM repository, client A – which has no access to the server S – can

identify a principal (client B) in the federation holding such privilege. Client A can then negotiate with client B in order to receive this privilege.

The presence of a client at distinct federations allows this client to easily access the public authorization certificates held by members of those 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*. The certificate managers can be associated to each other, linking those that can better represent the needs of their members. Such associations are done through trust relationships constituting *federation webs* (Fig. 3). This approach frees clients and servers from joining a considerable number of federations to achieve global scope.

Fig. 3 illustrates how the entities constituting a federation web are organized. Client authorization certificates (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 in the CMs of the federations they belong. The certificates available through CMs are used in the search of potential issuers of delegable permissions.

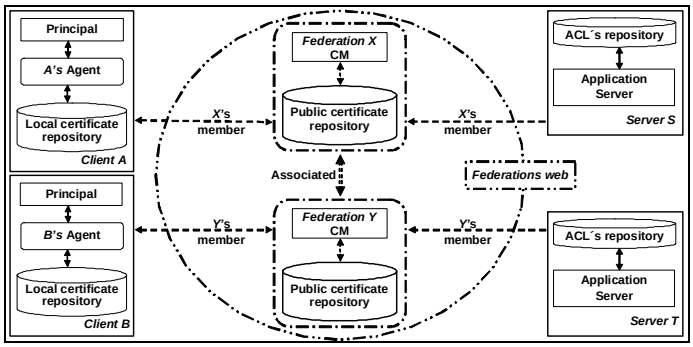


**Fig. 3.** Federation web overview

One can notice that there is no centralization or hierarchical arrangement in the proposal. The federation webs are arbitrarily formed, and do not play any active role in the authorization chains. In other words, they just carry out support roles in the authorization procedures.

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

### 3.1 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

own federation. The public keys of its members constitute a SDSI group. As the CM does not actively participate on any authorization chain, therefore it is not seen as a principal – it is basically a repository of public certificates.

In order to any ordinary principal participate in a federation, an endorsement in the form of a *threshold certificate* is demanded from it [8]. 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. When 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 make ease principal identification (section 3.3). To every new member joining the federation, a name certificate stating SDSI group inclusion is issued, for membership proving purposes.

The creation of associations among federations (federation webs) is also interpreted as membership inclusion in the SDSI groups of each federation involved. In this case, the new member (the other federation) is recognized as a group defined and administered within another naming space, according to the definition of SDSI groups [5].

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

### 3.2 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 (Fig. 3). This agent is a program 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; 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. 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.

### 3.3 Authentication, Authorization, and Auditing in the Model

In the SDSI/SPKI principals' authentication, the identification is not performed using names, but public keys, and the authentication is done through digital signatures. In order to check the digital signature on the destination, the principal's public key must arrive there securely. As there is no public key distribution entity 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. When this verification is successful, the guardian uses the last key in the authorization chain (the client's key) to check the digital signature on the request. Having this check been successful, then client's authenticity is verified.

Every authorization certificate carries the public key of the principal signing that certificate in 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 authorization chain. When all the mentioned procedures are successfully done, 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 search 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 entire mentioned authentication and authorization scheme described in this section is in compliance with the SDSI/SPKI specifications.

## 4. Creation of New Authorization Chains on the Proposed Model

There are several related experiences regarding procedures for searching SDSI/SPKI certificates [9,10,11,12]. However, in all these approaches, if a certificate chain is not found, the search is finished reporting an exception (fault), and the client is unable to perform the desired access. This work, through the federation notion, proposes a schema that enables a client to locate a certificate holding the needed authorization in a federation web. Later on, the client can negotiate with the privilege holder such grant to build an authorization chain that makes possible the desired access.

The scenario detailed bellow will consider the situation depicted in 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). In Figure 4, the messages exchanges are depicted for the case in which the chain between the client B and the server S does not exist.

The client B, member of federation Y, starts by requesting an access to server S (message *m1* in Fig. 4). Server S replies by sending a challenge message back to B. In this message (*m2*), server S informs the ACL protecting the requested object and asks from client B to prove its authorization for the requested access. In this case, SDSI/SPKI ACL data is effective to accelerate the searching process.

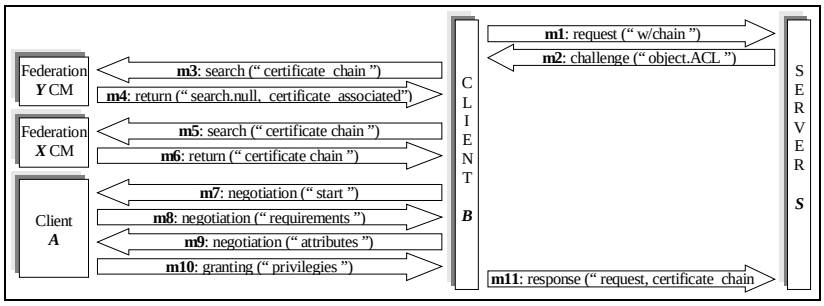


**Fig. 4.** Messages exchanges in the authorization chain compounding

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 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 (*m3*).

The attributes considered in the search are the required permissions and the public key of server S. In the case considered in Fig. 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, the certificates belonging to members of the associated federations (Federation X, in Fig. 3), so that it can keep on searching (*m4* message). Having the associated certificates, the client extends its search on the other federation's CMs (belonging to the federations web). Message *m5* corresponds to the queries on federation X in the considered example. In the message *m6*, client B receives as return from the X's CM a chain – the authorization certificate with the access permission between client A and server S. Then client B sends to the rights holder (A) the delegation right request (message *m7*). The delegation of permissions can be carried out in a simple way – because both the client and the rights holder are sharing the same federation, for example. However, depending on the application semantics, more complex negotiations may be demanded. The Fig. 4 represents this situation: the requested rights holder notifies client B about a set of requirements for the permission concession (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 and sends it on message *m10*. By this last message, the chain compounding process is concluded and client B can answer the challenge proposed by server S in the response message *m11*.

### 4.1 Example: Internet Commerce Application

In this section is depicted an example scenario to overview the usage of federation webs, which synthesizes the proposed schema. This scenario is built upon a Web-based e-commerce application, and involves 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 simplify the presentation, let's consider a credit card institution (CC) and a banking institution (BK), having some business agreement allowing each other easy financial transactions. Based on this agreement, the CC representative grants to the BK representative the right to "allow purchase" – in this case, the bank representative can allow purchases if payments are to be charged to credit cards issued by the credit card institution. The BK representative, whenever receives an authorization certificate with the delegation flag on, stores it on the CM of the federation FB.

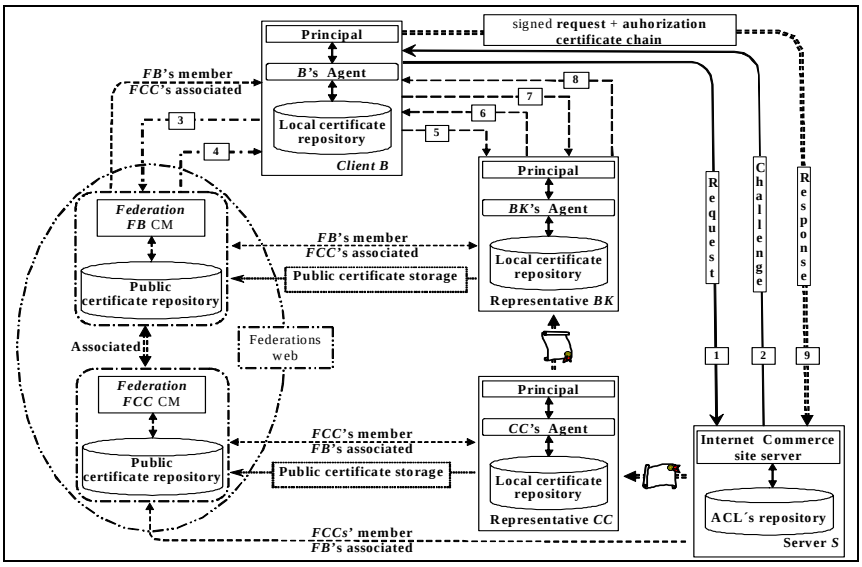


**Fig. 5.** Scenario for purchases in the Internet using of the federation web

Table 1 describes the messages (numbered in Fig. 5) exchanged between entities in order to implement the purchase transactions in the e-commerce site.

**Table 1.** Messages exchanged in Fig. 5 scenario

| Step | Actions |
|---|---|
| 1 | Client B navigates through the web pages offered by the e-commerce server S. After selecting some items to purchase, client B proceeds to checkout. |
| 2 | Server S sends back to the client a message containing the purchase bill and a challenge: the "allow purchase" privilege holder is the CC representative. |

| Step | Actions |
|---|---|
| 3 | Client B queries its local repository and finds no chains linking it to the CC representative. Then, client B sends the chain query to the CM of federation FB. |
| 4 | CM of federation FB makes a search in their repository and finds the required chain. It sends back to client B the chain between the server S and the BK representative. |
| 5 | Client B requests to BK representative the delegation of privilege “allow purchase”. |
| 6 | BK representative notifies client B that delegating the requested privilege requires paying the bill using one of the payment options offered by BK. |
| 7 | B client pays the bill using one of the options offered by BK. |
| 8 | BK representative delegates the “allow purchase” privilege to client B. |
| 9 | Client B sends the authorization chain to server S, along with the request (in a response message) and the server concludes the purchase transaction. |

In order to monitor the “allow purchase” privilege delegations, the CC representative receives a copy of all paid purchase bills from the e-commerce site.

In the scenario described here, no authorization chains existed linking the CC representative to client B. However, the mechanisms 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 e-commerce site. Of course, if the chain holding the requested authorization was not found in the CM of federation FB, the search would continue on the associated federations until an appropriate chain was found.

For the scenario described in Fig. 5, it should be noticed that the ACL of the server 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 strict 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, client information is stored only by its banking institution.

## 5. Architecture Implementation Aspects

The SDSI/SPKI infrastructure and the policies applied in the model (previously described in sections 3 and 4) are totally independent from the adopted technology. In this sense, the tools used in the prototype (Fig. 6) have been highly influenced by the model usage in the Internet – environment assumed as the context of this work.

The motivation on adopting CORBA as middleware is to take advantage of the services provided by that platform, mainly in aspects related to object lookup (name resolution) and secure remote access invocation. SSL (Secure Socket Layer) was adopted for remote communications. In order to establish a secure communication channel between a client and a server (holding SSL integrity and confidentially properties), mutual authentication for the principals (client and server) is required.

However, SPKI uses keys as principals, instead of names. To solve this, a function was developed to translate SDSI/SPKI into SSL name certificates.

The SDSI/SPKI integration with the distributed object middleware (shown in Fig. 7) was done using CORBASec at application level (CORBASec Level 2) [13].

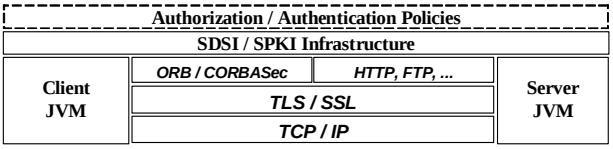


**Fig. 6.** Prototype model architecture

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 at the ORB level has been kept. The following session objects were maintained: *PrincipalAuthenticator*, *SecurityManager* and *Credentials* (Fig. 7).

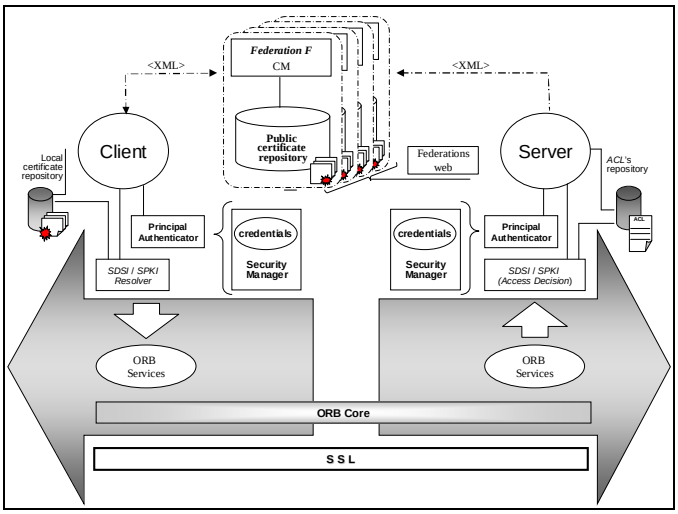


**Fig. 7.** CORBA-SPKI integration prototype overview

Fig. 7 shows other implementation details. The CM public certificate repository is implemented using Apache Xindice (which stores XML native data) [14]. The CM is implemented as an extension module of the Apache server [15]. All messages exchanged between members and CM is written in XML. The SDSI/SPKI certificates, originally coded as S-expressions, are translated into XML in our prototype for portability and standardization reasons [16]. The *SDSI/SPKI resolver* object shown in Fig. 7 is a partial implementation of the client's agent, covering chain searching 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.

## 6. Related Work

In [9], the DNS service was 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.

The work [10] views the network built by the propagation of SDSI/SPKI authorization certificates as an oriented graph. It also assumes that, in typical corporate environments, such graph is hourglass-shaped. This is 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 DFS 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 analyses some improvements in the DFS forward algorithm.

One can notice that the 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.

According to [17], 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. As the main purpose of the 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.

The chain search algorithms suggested by [12] and aspects considered there are deeper refinements of RFC 2693 recommendations. It also presents an implementation of the SPKI current version, quite rich in content, although no architectural propositions for distributed systems have been developed.

## 7. Conclusion

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

managers is to help in the construction of authorization chains, through a repository searching facility, for locating privileges needed by access. 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 constitutes a critical point regarding to faults, vulnerability or performance.

Adopting the federation web model 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 that 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 dynamic, even considering that in some cases the number of messages exchanged to create a new chain can be expressive.

## References


1. Horst, F. W., Lischka, M.: Modular Authorization. Proceedings of ACM SACMAT (2001)
2. Garfinkel, S.: PGP:Pretty Good Privacy. O'Reilly & Associates, Inc (1995)
3. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized Trust Management. Proceedings of the 17th IEEE Symposium on Security and Privacy (1996)
4. Blaze, M., Feigenbaum, J., Lacy, J.: The KeyNote Trust Management System, Version 2. IETF-RFC2704 (1999)
5. Lampson, B., Rivest, R. L.: A Simple Distributed Security Infrastructure (1996). URL http://theory.lcs.mit.edu/~cis/sdsi.html, Last access on Jun, 2003.
6. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI Certificate Theory. IETF-RFC2693 (1999)
7. Gasser, M., Mcdermott, E.: An Architecture for Practical Delegation in a Distributed System. Proceedings of the IEEE Symposium on Security and Privacy (1990)
8. Aura, T.: On the Structure of Delegation Networks. Proceedings of IEEE CSFW (1998)
9. Nikander, P., Viljanen, L.: Storing and Retrieving Internet Certificates. Proceedings of 3th Nordic Workshop on Secure IT Systems (1998)
10. Aura, T.: Fast Access Control Decisions from Delegation Certificate Databases. Proceedings of 3th Australian Conference on Information Security and Privacy (1998)
11. Ajmani, S.: A trusted Execution Platform for Multiparty Computation. Master thesis, Dep. of Electrical Engineering and Computer Science, MIT (2000)
12. Clarke, D. E.: SPKI/SDSI HTTP Server Certificate Chain Discovery in SPKI/SDSI. Master dissertation, Dep. Electrical Engineering and Computer Science of MIT (2001)
13. OMG – Object Management Group: Security Service Specification, v1.8 (2002). URL http://www.omg.org/cgi-bin/doc?formal/02-03-11.pdf. Last access on Jun, 2003.
14. Staken, K.: Xindice Developers Guide 0.7.1 (2002). URL http://xml.apache.org/xindice/guide-developer.html. Last access on Jun, 2003.
15. Thau, R.: Design Considerations for the Apache API (2002). URL http://modules.apache.org/reference. Last access on Jun, 2003.
16. Terreros, X. S. L., Ribes, J-M. M.: SPKI-XML Certificate Structure (2002). URL http://www.oasis-open.org/cover/xml-spki.html. Last access on Jun, 2003.
17. Li, N.: Local Names in SPKI/SDSI. Proceedings of the IEEE CSFW (2000).