Presented in 1st Latin-American Symposium on Dependable Computing – LADC 2003

Building Trust Chains between CORBA Objects

Emerson Ribeiro de Mello[1], Joni da Silva Fraga[1], Altair Olivo Santin[1], and Frank Siqueira[2]

[1] Department of Automation and Systems
[2] Department of Computer Science and Statistics
Federal University of Santa Catarina
Fax/Phone: +55(48) 331-9934
88040-900 Florianópolis, SC - BRAZIL
{emerson,santin,fraga}@das.ufsc.br, frank@inf.ufsc.br
http://www.das.ufsc.br

Abstract. This work presents an authentication and authorization model that results from the integration of the SPKI/SDSI infrastructure with CORBAsec. The paper presents the main facilities provided by the proposed model, showing the advantages of using the SPKI/SDSI infrastructure. CORBA provides to the model the advantages of interoperable distributed objects in heterogeneous environments. The idea defended in this paper is the better suitability of trust chains, as the SPKI/SDSI, with the characteristics of the world-wide network.

1 Introduction

The Internet provides a powerful communication infrastructure that has enabled the development of distributed applications with global coverage. However, together with the means provided by the Internet, new challenges appeared in the development of distributed applications, such as the aim for flexibility and scalability. CORBA (Common Object Request Broker Architecture), an architecture for open and distributed systems based on objects, appeared to solve such difficulties. CORBA middleware provides for interoperability and portability in distributed applications, as well as location transparency and platform and programming language heterogeneity.

With the wide use of distributed systems, information protection became a necessity. Distributed applications should avail of security mechanisms that guarantee properties such as integrity, confidentiality, authenticity and availability, so that these applications can operate in a correct and effective way. The CORBAsec (CORBA Security) specification appeared in the form of service objects to assist such security requirements in distributed object-oriented applications based on CORBA middleware. CORBAsec is a model capable of supplying functionality such as identification and authentication of principals, access control of method invocations, among others.

The classic authentication and authorization model employed in distributed systems defines, in a name domain, a centralized authentication process preceding the authorization, which can have its distributed controls. Such a model is suitable for environments in which the client is known beforehand; however, this model has shown to be inadequate for Internet applications, for example, when the client is likely to be unknown.

Developed in order to facilitate the conception of scalable and safe computing systems, SPKI/SDSI provides a fine access control, using a local name space and a simple authorization model based on trust chains (egalitarian model). The present work shows how to integrate the CORBA security model with the SPKI/SDSI infrastructure, providing a decentralized control for authentication and authorization policies with fine granularity. This paper is organized as follows. Section 2 describes the SPKI/SDSI public key infrastructure. Section 3 introduces the CORBA security service. In section 4, the proposed authorization model is presented, integrating the two above-mentioned technologies. Section 5 describes the implementation of the proposed model. In Section 6, related proposals found in the literature are described and compared to this proposal; and section 7 presents our concluding remarks.

2 SPKI/SDSI

The creation of both SPKI (Simple Public Key Infrastructure) and SDSI (Simple Distributed Security Infrastructure) was motivated by the limitations and by the complexity of public key infrastructures based on global name hierarchies, such as X.509. SPKI/SDSI is the union of these two formerly independent proposals. Created at the MIT by Ronald Rivest and Butler Lampson, SDSI [1] is a public key infrastructure (PKI) that has as its main characteristic the use of a local name space. Proposed by Carl Ellison and others, SPKI [2] was designed with the intent of being a simple and flexible authorization model, very well defined and of easy implementation. The union of SPKI and SDSI resulted in an authentication and authorization system for distributed applications.

2.1 SPKI/SDSI Certificates

In SPKI, a principal[3] is represented by a public key, and not by an individual name[4]. This represents a great advantage comparing to the approach based on global name hierarchies, because a public key is more stable than a name in large-scale systems (i.e. two people can have one same name).

Each SPKI principal holds a pair of keys that allows him to create, sign and publish certificates just like a X.509 certificate authority (CA). Version 2.0 of SPKI/SDSI defines two types of certificates: name certificate and authorization certificate. In SPKI/SDSI, for each public key there exists an associate local name space. A local name is a pair composed of a public key and an arbitrary identifier. The name certificate (see Table 1) is responsible for the publication of local names, allowing other principals to get the public key associated to the locally-defined name.

A name certificate can associate a name to the one public key, to other names inside the local name space of the issuer, or to other name certificates in the name space

[3] The term ‘principal’ usually refers to the individual that is represented by the public key, but it can reference entities representing this individual, such as an institution, a machine, or a process. The principal is always recognized by the security polices of the system as the authorized entity.

[4] Known as keyholder in SPKI.

Table 1. SPKI/SDSI Name Certificate

Issuer	Public key of the certificate issuer, whose signature should follow the certificate.
Name	Arbitrary local name which identifies the subject.
Subject	Public key or name composed of a public key followed by one or more identifiers.
Validity	Period during which the certificate is considered valid.

of other principals, forming certificate chains. Names propagation in SPKI is possible through trust chains, formed by certificates of linked names.

The authorization certificate (see Table 2) grants a specific authorization, given by the certificate issuer, to the subject of the certificate. The authorization granted by the issuer to the subject can be delegated by this subject, integrally or partially, for other principals if the delegation bit is active. When allowing the delegation of the certificate, the issuer assumes that the subject is trustworthy, because he will be capable of delegating rights to any principal that he wants without consulting the issuer.

Table 2. SPKI/SDSI Authorization Certificate

Issuer	Public key of the certificate issuer, whose signature should follow the certificate.
Subject	Public key or name composed of a public key followed by one or more identifiers.
Tag	Specification of the authorization that will be granted by the issuer to the subject.
Delegation bit	Binary field that indicates if the certificate can be delegated.
Validity	Period during which the certificate is considered valid.

In this way, the trust model allows the construction of authorization chains that begin with the key of the service guard and ends with keys of principals that want to access the service. A principal that wants to delegate a received right must attach the received certificate to a sequence of certificates and send this sequence to the subject, which then has the access right granted.

SPKI is a key-oriented infrastructure and as a consequence, for controlling access to services, it is necessary to resolve all the names that compose the chain in order to obtain the public key of the principal. The chain is considered valid if the first key of the chain is the public key of the resource holder and if the last key is the public key of the principal that wants to access the resource.

2.2 Access Control List (ACL)

A SPKI/SDSI ACL is composed of several entries, similarly to an authorization certificate (see Table 2), with the main difference being the lack of a certificate issuer;

therefore, an ACL is not signed. An entry in an ACL is said to be a delegation of rights given by the issuer, the owner of the service, to the subject [3]. The validity period of the entry is specified by the validity field, and in case this field is not specified the validity period is assumed to be unlimited.

3 CORBA Security Service (CORBAsec)

The CORBAsec specification [4] was designed to enforce security requirements in distributed systems. The CORBA security model is specified in the form of an object service (COSS), and provides secure exchange of information for distributed object applications.

The CORBA security specification defines a model capable of supplying mechanisms for identification and authentication of principals, controlling access to services through invocating remote methods, secure communication (including data encryption to guarantee the confidentiality as well as the integrity of messages), non-repudiation[5], auditing and administration.

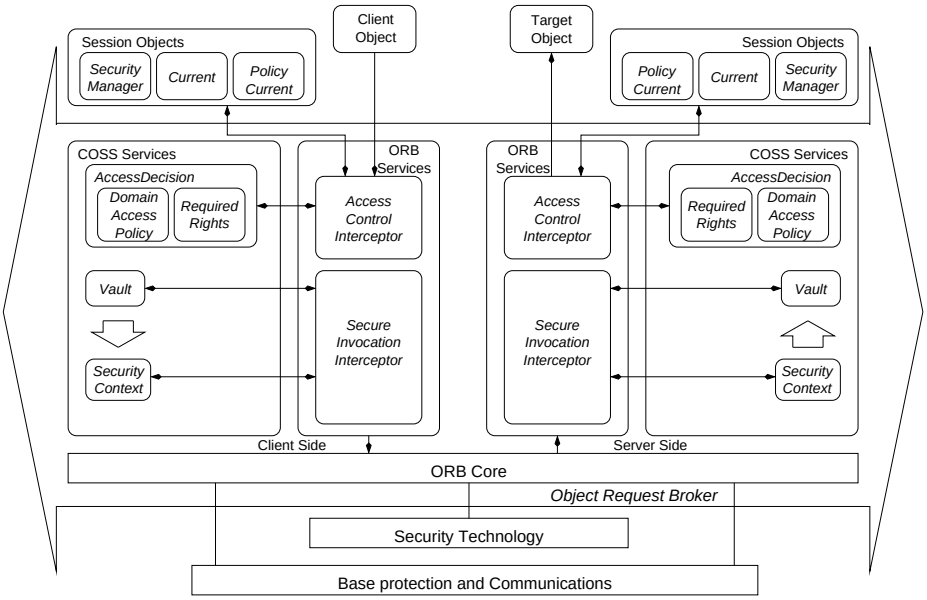

Fig. 1. CORBAsec Structure [5]

In the CORBA security model, objects are located at four levels (Figure 1): application level, which understands the application objects; middleware level, composed by service objects, ORB services and ORB core; security technology level, formed by

[5] This is optional and it is available only for CORBAsec level 2 (security-aware applications).

the underlying security services; and the basic protection level, composed of hardware platforms and local operating systems.

Notice that CORBAsec in itself does not provide any security mechanism, such as data encryption. Instead, CORBAsec provides only a standardized architecture and interfaces that can be used to provide security in a distributed object environment. Security mechanisms, such as Kerberos, SPKM, SESAME and SSL, must be provided separately. The study of these mechanisms is not the subject of this paper; a comprehensive survey on security mechanisms is found at [6].

Security in CORBAsec is provided in two levels. Security level 1, also known as ORB level security, supplies security mechanisms for security-unaware applications, guaranteeing access control and event auditing through the ORB. Security level 2, also known as application-level security, has other application and management interfaces besides supporting level 1 functionality. Level 2 is used by security-aware applications, allowing these to customize the way security is implemented.

A principal should first establish its rights before accessing remote objects. The PrincipalAuthenticator object implements the authentication service, returning credentials for the context of the principal. CORBAsec Level 2 allows the principal to change privileges in his credentials, and also allows him to choose which credential will be used for the invocation.

Access control at ORB level is accomplished in a transparent way for applications. During an invocation, or at binding time, the access control interceptor contacts the AccessDecision object that, through the access_allowed method, determines if the invocation should be allowed or not. In order to do so, the rights supplied by the client through the Credentials object are confronted with the rights required to execute the invocation (object RequiredRights).

Besides the access control interceptors that act during an invocation, CORBAsec also employs a secure invocation interceptor, which intercepts calls at a lower level in the sense of supplying integrity and confidentiality in messages transfers necessary to execute the invocation.

The way in which access control is accomplished in CORBAsec imposes to the model some granularity restrictions. The access control consists basically of the confrontation of rights supplied by the client (DomainAccessPolicy object) with the required rights (RequiredRights object).

Such fact happens because the supplied rights are related to the domain and the requested rights are related to methods, causing a granularity problem. Some solutions for such problem are proposed in [7].

4 SPKI/SDSI and CORBAsec Integration Model

In this section are described the mechanisms that supply the SPKI authorization and authentication controls in CORBAsec. In this work, SPKI/SDSI is extended with the use of Federations [8], which have the purpose of grouping principals with common interests, so that they can share SPKI/SDSI certificates.

4.1 SPKI/SDSI Federations

In [[8],[9]] a trust model that seeks the grouping of principals that possess similar interests was proposed. The main goal of SPKI/SDSI Federations is to provide means for certificate reduction and for the construction of new chains based on sharing certificates among their members.

Federations are composed of three entities: clients, servers and the certificate manager of the federation. The role of the certificate manager is to facilitate the interaction between clients and servers. The manager provides storage and recovery mechanisms, as well as mechanisms for creation of chains of authorization certificates, necessary so that clients can access servers without the need for an intermediate key.

A principal, whether he is a client or a server, when entering in a federation, supplies the name and authorization certificates (belonging to him and that can be delegated) that he considers useful for the federation. In this way, the repository kept by the certificate manager of the federation will contain all the delegable certificates of federation members. Whenever a principal wants to obtain access to a certain resource but does not have the required rights, it can ask the certificate managers of federations to which he is associated to locate members with delegable certificates that allow access to the required resource. Once found a principal with the required rights, a negotiation among principals – the one that wants to obtain rights and the other who can delegate them – is started, intending to have the required rights delegated to the first. The negotiation can comprise from a simple transfer of rights to a more complex process, such as a financial transaction; the nature of the negotiation process is directly related to the semantics of the application.

A principal can join as many federations as he is accepted. In order to join a federation, the principal may be required to supply a threshold certificate signed by k-of-n members that already belong to the federation, along with a name certificate with which he is requesting the admission. The purpose of joining several federations is to obtain access to a larger amount of resources. By having access to the delegable authorization certificates of all members of each federation joined by him, it becomes easier for a principal to be recognized by other applications spread through the worldwide network.

However, in a large-scale system such as the Internet, it might be necessary for a principal to join several federations in order to be easily recognized. The model avoids this scalability problem by establishing associations among federations, allowing searches and delegations to comprise not only one federation, but a group of associated federations. As a consequence, the amount of federations a principal must join in a large-scale system is reduced sharply.

4.2 The Role of CORBAsec in the Proposed Model

The CORBA architecture and it security service (CORBAsec) were used as the communication infrastructure, guaranteeing the interoperability and the security in the communication in distributed systems.

The proposed authorization model, which is implemented at the application level, employs five CORBAsec objects located at the ORB level: SecurityManager, PrincipalAuthenticator, Credentials, Current and AccessDecision. Other ORB level objects

responsible for establishing a secure communication channel – secure invocation interceptor, Vault and SecurityContext – are also used. However, the access control is handled at application level, employing CORBAsec specifications regarding Security Level 2.

The proposed model for integration of SPKI/SDSI with CORBAsec seeks to respect the philosophy of the SPKI/SDSI 2.0 model, where the principal that wants to access a resource is entirely responsible for locating certificate chains that supply it with the required rights to access this resource, leaving to the resource provider the task of verifying the authorization and the authenticity of the request. This verification is executed in the following way: in the attempt to access the resource, the rights supplied together with the request are confronted by the service guard (reference monitor) with the requested rights expressed in an access control list (ACL), determining whether the access to the resource will be allowed for the applicant or not.

The proposed model, therefore, uses CORBAsec Security Level 2 in the sense of maintaining the characteristics of SPKI/SDSI (i.e. keeping security controls at application level). The integration model between SPKI/SDSI and CORBAsec is also a solution for the granularity problems identified in CORBAsec, presented in section 3.

4.3 Dynamics of the Proposed Model

According to the proposed model, a client must be authenticated in order to join the CORBAsec object system. This authentication is accomplished through the PrincipalAuthenticator object, which creates the Credentials object. This object has the privilege attributes of the authenticated client, which will be stored in the SecurityManager session object. All these objects are illustrated by Figure 2.

In order to guarantee the integrity and the confidentiality of messages, SSL version 3 is employed as the underlying security technology. In this way, the authentication process in the CORBA security model is accomplished through the reading of self-signed SSL certificates, which are translated from SPKI/SDSI name certificates according to rules specified in [2]. These SSL certificates are used for mutual authentication and for establishing a secure session (i.e. a cryptographic channel) between a client and a server.

The ability to work with SPKI/SDSI objects – i.e. to generate certificate chains and to verify the authorization given by a chain – is given to the application by the SPKI/SDSI Resolve object (Figure 2), which encapsulates the SPKI/SDSI 2.0 library.

In the proposed authorization model, the confrontation of the client´s privilege attributes with the control attributes that protect the requested object occurs in a different way, therefore the authentication procedure and the credentials created by this process will not be used to guarantee the access authorization. The access authorization procedure starts with the transfer of SPKI/SDSI authorization certificates from the client to the server using CORBAsec credentials. The server uses the certificates received from the client to activate the service guard, which is responsible for confronting the requested rights with the supplied rights.

On the side of the server, the AccessDecision SPKI/SDSI object (Figure 2) is responsible for deciding if the requests send to the server object will or not to be executed. SPKI/SDSI ACLs play the role of the CORBAsec RequiredRights object, defining the

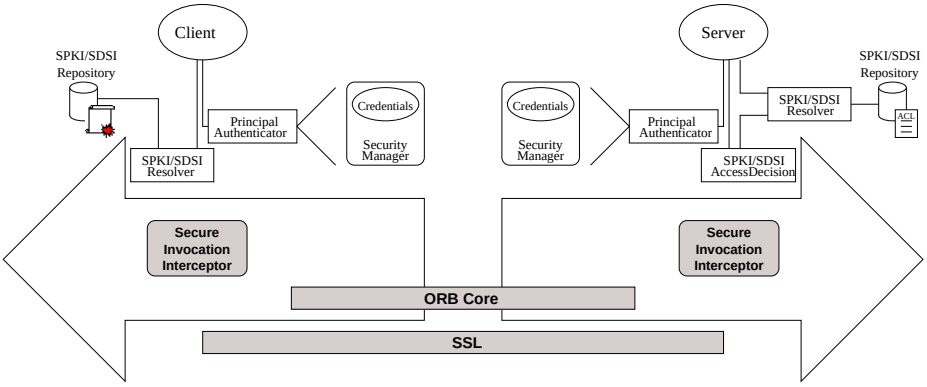

Fig. 2. Overview of the SPKI/SDSI - CORBAsec Integration Model

requested rights necessary to invocate each operation supplied by the server, an also the role of the AccessPolicy object, providing groups of principals with rights for executing operations implemented by objects belonging to a domain.

The flexibility offered by CORBAsec level 2 [4] for handling credentials is fully adapted to the use of SPKI/SDSI certificates. Security-aware applications are able to control the security options used during an invocation by choosing the method used for message protection, the privileges contained in their own credentials, the credentials that will be used in the invocation of an object, and if these credentials can only be used by the target object or if they can also be delegated. According to these statements, we have decided to make available in the form of credentials the authorization certificates belonging to the client that are necessary for obtaining of access right to distributed resources. The client's credentials are rebuilt on the server side and can be obtained through the ReceivedCredentials object (Figure 3)

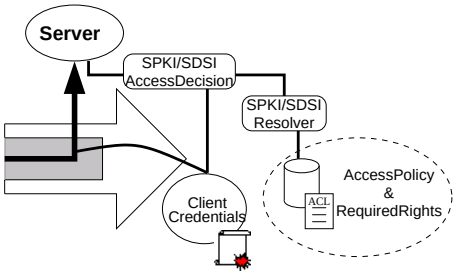

Fig. 3. Objects used for Access Control on the Server Side

4.4 Adding and Recovering Certificates

An object, upon joining the CORBAsec object system, will authenticate itself through the PrincipalAuthenticator object. The authentication is accomplished by presenting its SSL certificates, since this is the security technology used by the model. However, the credentials created during the authentication process for joining the CORBAsec object system will not be used for access control.

In CORBAsec level 2, the client has the ability to choose which credentials will be used for an invocation. For each authorization chain built by the client in the attempt to accomplish the challenges thrown to him, an object SPKI/SDSI Credentials[6] is created. These authorization chains are converted into sequences of bytes, which will be added as privilege attributes to the credentials defined by CORBAsec.

On the server side, the SPKI/SDSI AccessDecision object is called automatically for each request of a method. This object makes the confrontation of the rights supplied by the client with the necessary rights, specified in an ACL. The client's rights (certificate chain) encapsulated by the SecurityManager session object (Figure 2).

4.5 Access Control List (ACL)

SPKI/SDSI is a practical system that targets the problem of assuring that a user is authorized to execute an action and not just the problem of identifying the user. This characteristic allows larger flexibility in the sharing of resources through delegations, contrasting with authentication systems based on conventional public key infrastructures and authorization through conventional ACLs, which are built starting from the principals´ names.

In conventional public key infrastructures, if a server S wants to verify if principal U really has rights to access the service, it is necessary to consult a Certificate Authority (CA), with which the server has a trust relationship, emitting an authorization certificate that contains the permissions and the name of the principal that is requesting access (U). Besides the server, the only identification of the principal (the client's global name) is contained in the server's ACL.

In SPKI, there is no such entity as a Certificate Authority – instead, each principal is able to issue authorization certificates, so server S can issue an authorization certificate for principal U. Consequently, the use of ACL becomes unnecessary [10], since client U informs which certificates it already has upon accessing the service, and this information is enough for the server to verify the authenticity of the request and if the client has the required rights.

The proposed model uses ACLs in order to allow easier location of valid certificate chains inside SPKI/SDSI Federations. An ACL is composed by several entries, and each entry is composed by a public key and a tag. Tags could be, for example, names of methods provided by a server, allowing him to control access to his methods.

The delegation of authorization certificates is extremely important in the proposed model, since it allows that other entities become authorization issuers. The concept of

[6] This is a CORBAsec SecurityLevel2 Credentials object, but with specific functionality for controlling access using SPKI/SDSI certificates.

delegation is used by SPKI/SDSI Federations, since members supply all their authorization certificates that can be delegated and that are important for the federation upon joining it. The federation manager, being a passive entity (i.e. an entity without a public key), does not take part in the trust chain. Instead, the manager acts as a simple repository.

4.6 Mutual Authentication and Authorization in the Proposed Model

One important security principle is to guarantee that protected information will only be revealed to authorized and authentic entities. Cryptographic systems, whether symmetric or based on public keys, provide different means to guarantee these principles. Once guaranteed that the entity involved in the communication is authorized and that the request is authentic (i.e. really originated by the alleged entity), the protected information should be supplied to the requesting entity.

The establishment of secure communication demands that some control messages be exchanged among the communicating parts. The challenge/response protocol, frequently employed by public key infrastructures, is responsible for mutual authentication. This protocol gives assures the client that it is being connected to the right server, and assures the server that it is receiving requests from a valid client.

The communication between client and server in the proposed model uses the challenge/response protocol as a base for the authentication of their respective principals based on SPKI/SDSI authorization certificates and ACLs.

Consider the following scenario: a client wants to invoke a method of a server object. This method is protected by an SPKI/SDSI ACL and, if the request does not cover the necessary requirements, the client must win a challenge in order to have the invocation executed. The steps involved in this protocol between client and server are illustrated by Figure 4 and described below.

In the step 1, the client sends a request to the server together with a nonce[7] (NonceCli) without any authorization chain, because the client ignores the requested rights. Upon receiving the request, the server activates the service guard, which is responsible for confronting the requested rights, contained in the ACL[8] , with the supplied rights. The server generates a challenge, signed by its private key, to the client (step 2). The challenge is composed of the ACL that protects the resource, the nonce sent by the client (NonceCli) and a second nonce (NonceServ) generated by the server.

The client verifies the authenticity of the challenge using the public key of the server[9] and, if the challenge is authentic, activates the necessary devices to generate the certificate chain of that guarantees the access to the resource provided by the server.

Once generated a valid authorization chain, the client sends the response to the server (step 3). The response is composed of the original request, the NonceServ (both signed with the client's private key), followed by the authorization chain.

[7] A random value sent by a server or application requesting user authorization.

[8] Sending the ACL to the client does not imply on security problems, because the ACL entries are composed of public keys, which do not identify their holders.

[9] The server´s public keys are known beforehand by the client, possibly through a name certificate.

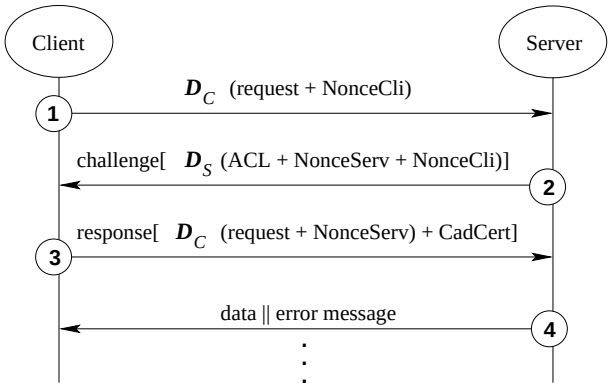

Fig. 4. Access Control Protocol

The server, by holding the client's public key (that can be obtained getting the last key of the certificate chain received from the client), verifies the authenticity of the response. Guaranteed the authenticity, the server activates the service guard in order to verify if the chain really grants the required authorization. In the step 4, the request is granted, returning the wanted information, or refused if the provided chain is not accepted.

It is important to point out that in the protocol described above, the mutual authentication is totally based on SPKI/SDSI authorization certificates. This is a characteristic of the SPKI/SDSI infrastructure.

5 Prototype Implementation

A prototype implementation of the proposed model using SPKI/SDSI certificates is presented and discussed in this section. The architecture of the prototype (Figure 5) employs tools widely used on the Internet, environment in which this work is inserted.

5.1 Architecture

The prototype avails of support for SPKI/SDSI provided by the JSDSI2.0 library, implemented by [11]. Client and server codes are interpreted by a Java Virtual Machine (JVM). The use of Java is currently widespread in distributed applications, mainly because it provides means for the development of platform-independent applications based on Internet application protocols.

CORBA and CORBAsec make possible the interoperability between distributed applications, which are composed of objects running on heterogeneous systems. In the prototype, ORBacus [12], which is an Object Request Broker (ORB) compatible with CORBA version 2.3, was used together with ORBAsec SL2 [13], which despite not being a complete implementation of the CORBAsec level 2 specification, satisfies the needs of the prototype.

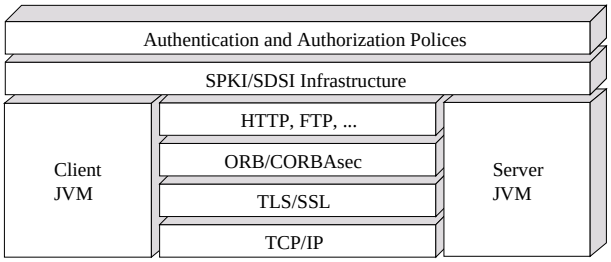

Fig. 5. Prototype Architecture

The protection of messages sent through the network is guaranteed by TLS/SSL (Transport Layer Security / Secure Sockets Layer). By using this layer, the prototype can guarantee the confidentiality and the integrity of the communication through the network among components running on different machines. The prototype integrates to the ORB the iSaSiLk library [14], which is a fully functional implementation of SSL version 3. Other implementations of SSL, however, can also be used.

5.2 Search for Certificate Chains

The first step of the client when receiving a challenge is to look for a certificate chain linking the public key of the server to his public key in the local repository. In case such chain does not exist, the second step is to search for chains linking him to principals listed on the ACL sent by the server.

The search for a chain is performed using the following parameters: PubKeyOfThePrincipalOfTheACL, PubKeyOfTheClient, TAG. At first, the search is performed on the local repository of the client. This search is repeated N times, where N is the number of public keys contained in the ACL.

The third step occurs inside the context of SPKI/SDSI Federations to which the client belongs, whenever the chain of necessary authorization satisfying the challenge does not exist in the local repository. The search is then performed by the certificate managers of each federation. The mechanisms used for searching and generating the certificate chains used by managers are out of the scope of this text; a description of these mechanisms can be found at [8].

5.3 Certificate Repository

SPKI/SDSI certificate repositories can be implemented in several ways. SPKI/SDSI objects are composed of S-expressions [15] that can be stored in ASCII format.

In [16] a standardized way to transform SPKI/SDSI objects represented as S- expressions into XML documents (eXtensible Markup Language) was proposed. Local repositories were built in the prototype based on this proposed, i.e. the local repositories store XML documents converted from SPKI/SDSI objects coded as S-expressions. SPKI/SDSI objects are transformed into XML documents so that they can be stored, but

once recovered from the repository they are converted back into S-expressions. This solution allows certificates to be easily exchanged among applications (CORBA objects) involved in the communication and is still compatible with any application that works with SPKI/SDSI objects.

6 Related Work

In [17] is described a complete implementation of the current version of SPKI/SDSI. The implementation was verified building a HTTP server on top of it. The access control protocol adopted defines challenges thrown by the server to the client. Some variations of this protocol have been presented seeking to reduce the amount and the size of messages exchanged between the client and the server.

Clarke also describes a form to protect ACLs against non-authenticated principals, with the protection being done through another ACL. In this way, rights held by each principal on a resource will remain secret for non-authenticated users. Nevertheless, a deeper analysis verifies that is possible that non-authenticated users get the list of principals that have access to the resource, because the key of the principal should be contained in two ACLs. The same is not true for the access rights, because they can be different in each ACL.

The approach described above has the inconvenience of countless message exchanges between the client and the server, because each method request generates two challenges and two responses. The security gain obtained by this approach is not very significant, because SPKI/SDSI public keys already conceal the entities that they represent, imposing to the invader the difficulty of locating the target to be attacked.

In [18] is presented a way of implementing authorization in CORBA distributed applications with SPKI certificates. The resulting implementation is compared with CORBA´s access control, showing the advantages of the proposal. Lampinen employed SPKI version 1.0, which still had not been merged with SDSI – this took place in version 2.0. The client, when joining the CORBA object system, makes available all the authorization certificates belonging to him in a single Credentials object. When receiving an invocation, the server obtains all the authorization certificates belonging to the client and searches for a chain that provides the necessary authorization.

Such approach, besides diverging from the principles used to conceive SPKI/SDSI, also presents scalability problems, because the client can have a large amount of certificates, which is likely to reduce the performance of the application.

In [19] is described a proposal of distributed security model for the Jini technology. The author states that the Jini architecture does not provide any security mechanism in addition to the standard mechanisms provided by Java, such as protection of the client's Java Virtual Machine (JVM) against malicious code executed by proxies.

Eronen and Nikander [19] proposes a solution for the security problems that are not targeted by the Jini architecture, being able to authenticate clients and services and to verify authorization at the level of method calls. However, requires the TAG field of SPKI certificates to be modified, preventing interoperability with SPKI/SDSI certificates provided by other applications.

The model proposed in this paper focused on the current version of SPKI/SDSI. In our proposal the principal, which is the client of the communication, is the only responsible for locating the authorization chain necessary for the access. The server has only the task of verifying whether the chain is valid or not. The proposed model intends to provide a more flexible model for authorization for distributed objects by integrating CORBAsec level 2 with SPKI/SDSI. The granularity problems presented by the former (see section 3) are solved by the integration with SPKI/SDSI.

Access control is implemented in a decentralized way with the necessary granularity. All the characteristics of the current version of SPKI/SDSI are respected, as well as the CORBAsec security level 2 specification. The facilities provided to principals, clients and servers do not impose constraints on their implementations, limiting the effort necessary for using the security service. The authentication protocol guarantees that the entities involved in the communication are authentic and protects the system from replay attacks.

7 Conclusion

This paper presents the proposal of a flexible and scalable authentication and authorization model for large-scale systems such as the Internet. The proposal employs SPKI/SDSI certificates as an alternative for access control in the CORBA security model. CORBAsec is employed as a vehicle for the introduction of the concept of trust chains in order to develop interoperable distributed objects in large-scale distributed systems.

CORBAsec security level 2 provides the necessary facilities for developing this proposal. Such freedom for the implementation of security in applications was extremely important in the model. The use of SPKI/SDSI certificates has shown to be the right choice because it limits the scalability and flexibility problems present in large-scale distributed systems. The implementation of the repositories in XML simplifies the search and generation of certificate chains.

This work is part of a larger project that aims to develop an authentication and authorization support for distributing documents through the Internet.

References

1. Rivest, R.L., Lampson, B.: SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession (1996) http://citeseer.nj.nec.com/rivest96sdsi.html.
2. Ellison, C.M., Frantz, B., Lampson, B., Rivest, R., Thomas, B.M., Ylonen, T.: SPKI Certificate Theory. Internet Engineering Task Force RFC 2693. (1999)
3. Li, N.: Local Names in SPKI/SDSI. In: Proceedings of The 13th Computer Security Foundations Workshop, IEEE Computer Society Press (2000) 2–15
4. OMG: Object Management Group - Security Service v1.7. OMG Document 01-03-08 (2001)
5. Westphall, C.M.: Um esquema de autorização para a segurança em sistemas distribuídos de larga escala. PhD thesis, Federal University of Santa Catarina - Brazil (2000)
6. Gollmann, D.: Computer Security. John Wiley & Sons (1999)
7. Hartman, B., Flinn, D.J., Beznosov, K.: Enterprise Security with EJB and CORBA. OMG Press. John Wiley & Sons (2001)

8. Santin, A., Fraga, J., Mello, E., Siqueira, F.: Um modelo de autorização e autenticação baseado em redes de confiança para sistemas distribuídos de larga escala. In: Anais SSI 2002, São José dos Campos, SP - Brazil, ITA, SSI (2002) 101–110
9. Santin, A., Fraga, J., Mello, E., Siqueira, F.: Teias de Federações como extensões ao modelo de autenticação e autorização SDSI/SPKI. In: Anais XXI Simpósio Brasileiro de Redes de Computadores, Natal, RN - Brazil, SBRC (2003) 553 – 568
10. Nikander, P., Viljanen, L.: Storing and Retrieving Internet Certificates. In: Proceedings of the Third Nordic Workshop on Secure IT Systems. (1998)
11. Morcos, A.: A Java implementation of Simple Distributed Security Infrastructure. Master's thesis, MIT (1998)
12. IONA Technologies Inc.: ORBacus User Guide. (2001) Version 3.3.4.
13. Adiron, LLC: ORBAsec SL2 User Guide. (2000) Version 2.1.4.
14. Institute for Applied Information Processing and Communications (IAIK): iSaSiLk 3 - Reference Manual. (2000) Version 3.
15. Rivest, R.L.: SEXP (S-expressions). Internet Engineering Task Force - Internet Draft (1997) http://theory.lcs.mit.edu/ rivest/sexp.html.
16. Orri, X., Mas, J.M.: SPKI-XML Certificate Structure. Internet Engineering Task Force - Internet Draft. (2001) http://xml.coverpages.org/ni2001-12-18-a.html. Visited in 04/02/2003.
17. Clarke, D.E.: SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI. PhD thesis, MIT (2001)
18. Lampinen, T.: Using SPKI certificates for authorization in CORBA based distributed object-oriented systems. In: Proceedings of the 4th Nordic Workshop on Secure IT Systems (NordSec '99), Kista, Sweden (1999) 61–81
19. Eronen, P., Nikander, P.: Decentralized Jini Security. In: Network and Distributed System Security Symposium - (NDSS 2001), San Diego, California (2001)