Delegation-based access control for intelligent network services

Tuomas Aura, Petteri Koponen, Juhana Räsänen
Helsinki University of Technology FIN-02015 HUT, Finland
{Tuomas.Aura,Petteri.Koponen,Juhana.Rasanen}@hut.fi

Abstract

Delegation with public-key certificates appears to be a natural technique for access control between intelligent network (IN) service providers. It supports strongly the IN business model and fits well to an object-oriented design. In the Calypso project, we are implementing access control to Java-based IN services with SPKI delegation certificates.

1 Introduction

The Calypso IN service architecture [5] is based on flexible distribution of service and network control functions among service clients, servers and network nodes. This is achieved by providing a Java-based, thin service platform, on top of which a dynamically configurable set of services can be executed. In the Calypso architecture, a service is implemented as a Java package. A service may use other services by calling methods of the classes that belong to them. The service platform manages most of the access control between the services.

In this paper, we argue that delegation with key-oriented certificates is an excellent choice for access control between IN service providers (SPs). The concept of delegation directly supports the business model for IN service production. It allows free evolution of relations between the producers of the services. Key-oriented certificates enable flexible and completely distributed access control without central authorities. Moreover, when services are implemented as modules of an object-oriented programming language, these modules provide a natural level granularity for authorizations.

2 Calypso architecture

The Calypso architecture is designed for ATM-based access networks. In the ATM networks, a workstation running the Calypso service platform controls an ATM switch. We adopt here the broad definition of a service by Magedanz and Popescu-Zeletin [6]: the Calypso services include both the network services (*bearer services* such as audio, video, and data transmission and signalling services) and the services offered by the IN platform to the end users (*value-added services*). Thus, in the Calypso architecture, also the traditionally static network services can be configured according to the needs.

2.1 Business model

Calypso is based on a business model where the network operator who owns the infrastructure offers network resources to service providers. These resources are implemented as low-level of Calypso services. Thus, the SPs purchase from the network operator access rights to certain Calypso services. For example, if one or more services of an SP

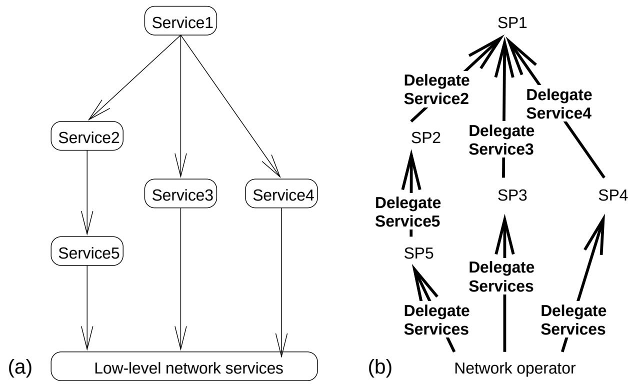

Figure 1: Service composition and delegation of access rights between SPs.

need to create ATM connections, the SP buys the right to use the connection service. After this, access rights to the connection service are granted to the correct services of the SP.

Service providers can either market the right to use their services to end users or they can sell them to other service providers for reselling or for use as building blocks of more sophisticated services. Because all the services are implemented as Calypso services, the service provision mechanism of network operators and service providers is the same. Consequently, the same access control mechanism can be used for services at all levels.

2.2 Implementation issues

The service platform will be implemented with Java. However, some of the services may include native libraries that are implemented with C language. For example, in the current implementation, the ATM switch service includes native parts for communicating with device drivers. These services are considered trusted and they are usually provided by the network operator or equipment manufacturer.

The Calypso architecture should enable the execution of *untrusted* services, i.e. Java classes coming from the service providers, at all networks nodes. This is because some services benefit from the possibility of directly controlling the network node or from being located near the end users (in the nearest network node, for example); see the [5] for details. Hence, the network nodes must have an access control system that facilitates execution of code from mutually distrusting SPs and contracting of services between them.

3 Service structure

A typical intelligent network service is built on top of other IN services. The layered structure of services frees designers of high-level services of low-level considerations, thus making it possible to implement a wide variety of specialized services at reasonable cost. For example, a video conferencing service can make use of point-to-multipoint services if they are available but it cannot implement the multicast routing itself.

The composition of a service can be depicted as a tree. In a *service call tree*, the service is placed at the root of the tree, the lower-level building blocks utilized by the service are drawn as its children, and their building blocks as their children, etc. (Fig. 1 (a)).

A service may access building blocks produced and marketed by other service providers. These relationships between service producers are a key to successful IN operation. The architecture should encourage innovative content and value-added service development by independent producers. It is essential that contracting building blocks from other service providers is easy and safe for both parties. Thus, a mechanism for access control and secure distribution of access rights between SPs must be integrated into the IN architecture.

Traditional telecom models of access control, however, cannot provide the flexibility needed in distributed IN services. The IN access control mechanisms should facilitate free formation of relations between service producers and merchants distributing access rights to the services. The entities involved in the value chain should be able to specialize as service developers, content providers, service brokers or consumer-oriented service providers. In particular, competition from the fast-advancing Internet services sets high requirements for the ease of purchasing and selling IN services.

Delegation, in a natural way, allows complex relations between entities without forcing too much pre-define structure on the service market. Fig. 1 (b) shows how access rights to services are delegated by the producers to the clients whose services need them.

4 Delegation-based access control

In the Calypso project, we have chosen to build access control on delegation. Certificates signed with public-key cryptography are used to delegate access rights to services between service providers. In practice, the certificates follow the SPKI draft standard [3].

4.1 Delegation certificates

Cryptographic signatures are written with a private key and verified with a public key. The owner of the key keeps the private part of the key secret. The public part can be distributed freely and it is used for verifying the signatures. Certificates are signed documents with which the signer conveys its beliefs or decisions about some other entity to those who accept the signer as an authority in the matters stated in the certificate. For example, a certificate signed by the owner of a file can grant file access rights to another entity. This type of certificates that delegate authority from one entity to another are called *delegation certificates*.

We take a *key-oriented* point of view where a cryptographic signature key represents the entity holding the private part of the key. The access rights are delegated directly from key to key without need to explicitly mention the names of the involved entities in the certificates.

With key-oriented certificates, redelegation is simple. If Alice's key authorizes Bob's key to use a service and Bob's key authorizes Charlie's key to use it, this is considered equivalent to Alice's key delegating directly to Charlie's key. Certificates forming a chain can be recognized and reduced in this way into a single certificate.

After the access rights have been delegated from key to key, the last key in the chain must be able to use the service. Normally in delegation systems, it simply signs a service request with its signature key and sends the chain of certificates to the server along with the request. The server checks that the request is signed by the key to which the last certificate of the chain has been issued, that the issuer of the first certificate is the owner of the service requested, and that the certificate chain properly reduces into a single certificate. If the check succeeds, access is allowed.

The Calypso system differs slightly from this standard scenario. In Calypso, the last key in the chain writes one more certificate to delegate the access rights to the service code that will use them. In practice, the certificate is issued to a cryptographic hash value of the code module. The service platform acts as a trusted entity that checks the

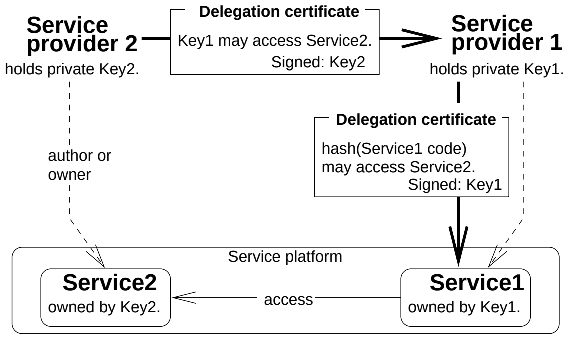

Figure 2: Delegation to another service provider

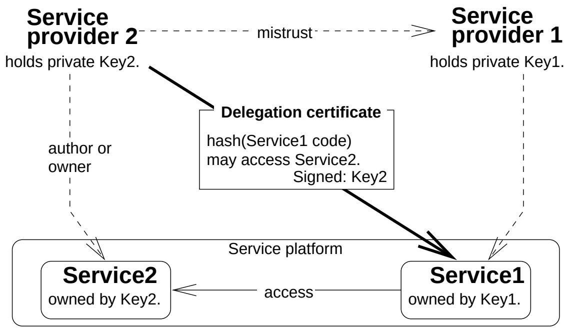

Figure 3: Delegation directly to code

certificates before allowing the code to access its building blocks or their proxies on the same platform.

SPKI certificates support both types of delegation: to public keys and to object hashes. In our system, the chains of delegation always end at code hash values.

In the certificate chain, the rights can be restricted at every step by delegating only a subset of the rights. It should be noted that although the authority appears to flow through the certificate chain from the beginning to the end, the certificates can be created and updated in any order. Also, expired certificates can be replaced by certificates from other issuers so that a complete chain is formed again.

4.2 Delegation in Calypso

Figures 2-4 show three ways of implementing the delegation from SP2 to SP1 in Fig. 1.

The most common scenario for delegation from one service provider to another is presented in Fig. 2. SP2 owning Service2 delegates the right to use it to SP1. This further delegates the right to a code module implementing Service1. Both certificates are passed to the service platform with the code of Service1. The platform verifies the certificates and notes that there is a complete chain of delegation from the key Key2 of SP2 to the hash of Service1 code. Consequently, it will allow access from Service1 to Service2.

Fig. 3 has a different scenario where SP2 does not trust code written by SP1 and wants to review it before granting access. After the review, it delegates directly to Service1 code. However, since the certificate to the code must be renewed every time the code is updated, it is usually preferable to follow the model of 2 and to give this task to the service provider responsible for the code.

The delegation between SP2 and SP1 does not need to be direct. Fig. 4 shows how

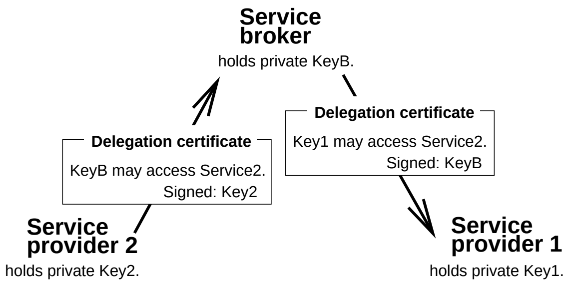

Figure 4: Delegation through a service broker

the right to access a service may be passed through a service broker.

Between IN service providers, the delegated authorizations are always rights to access a service. Since services in the Calypso architecture are encapsulated into Java classes and packages, this is also the granularity level of the access control. In some cases, the certificates may contain parameters that limit the scope of the delegated service.

This means that every time the delegated service is restricted in some way, a new *handle* class must be created to act as a proxy between the full-rights service and the client. If the restriction is of a new type, the handle class must be implemented by the service provider and distributed to the networks nodes. On the other hand, handle classes for standard restrictions, such as limiting access to a subdirectory of a document tree, can be created automatically at the service nodes. In practice, the mechanism is controlled by a modified Java class loader and it resembles the idea of name space management in [9].

At the moment, we do not have any secure accounting mechanisms for service usage. In the future, some secure mechanisms must be added for accounting and enforcing quantitative limits on the delegated rights. This requires some more experience with applications and the preferences of service providers. We are investigating the use of the SPKI on-line testing feature and accounting servers.

5 Comparison to related work

Key-oriented access control is a relatively new concept [3, 2] and there are few theoretical treatments of it [1]. Key-oriented systems do not need any central trusted entity. This their main advantage compared to traditional, identity-oriented systems where certificates and access control list (ACL) entries list entity names instead of their public keys.

In fact, the system presented in this paper does not have any ACLs. Each service has exactly one key associated with it: the key of the service author or owner. This key distributes the access rights by signing delegation certificates. The decisions to delegate depend on the business relations between the entities possessing the private keys.

Moreover, we avoid using identity certificates completely. The certificates are always issued to public keys or code hashes. Certificates written to object hashes are included in SPKI and similar ideas have been used earlier at least in the Taos operating system [11]. In the future, we may add support for SDSI-style linked names [7].

In most capability-based access control systems there is some control for redelegation of access rights [8]. Although it is possible to forbid redelegation in SPKI certificates, we have avoided using this feature. Instead, the access rights can be limited in each step of delegation to reflect the level of trust in the client. The purpose is to allow free and completely distributed formation of trust relations between entities.

Although the high-level services in Calypso are implemented in Java, the Java (JDK

1.2) security model [4] does not support the business relations between IN service providers as well as delegation. In particular, we make no use of X.509 certificates or stack inspection [10]. The delegated rights are rights to use a service which always implies indirect access to lower-level service building blocks. This way, if all services have acquired access rights to their direct building blocks, service calls will always succeed. In the standard Java security model, the top level service would usually have to possess rights to all building blocks even several layers below it.

The main reason why we have been able to design a simple and elegant access control system based solely on delegation is that the players in the system are IN service providers. Unlike typical applet users, they are able to make policy decisions when signing certificates and creating handle classes with limited rights. However, the service producers in our system do not actually need to think about access control policies. Instead, they sell and buy the certificates with business relations in mind.

6 Conclusion

In this paper, we discuss techniques for access control between service providers in an intelligent network. The main observation is that delegation with key-oriented certificates is naturally suited for the purpose. It fits well together with the business model for IN service and with object-oriented implementation techniques. We emphasize that the access control mechanisms should be designed to support business relationships between service producers.

References

[1] Tuomas Aura. On the structure of delegation networks. In *Proc. 11th IEEE Computer Security Foundations Workshop*, Rockport, MA, June 1998. IEEE Computer Society Press.

[2] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proc. 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, CA, May 1996. IEEE Computer Society Press.

[3] Carl M. Ellison, Bill Franz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylönen. Simple public key certificate. Internet draft, IETF SPKI Working Group, March 1998.

[4] Li Gong. Java security architecture (JDK1.2) draft document revision 0.8. Technical report, Sun Microsystems, March 1998.

[5] Petteri Koponen, Juhana Räsänen, and Olli Martikainen. Calypso service architecture for broadband networks. In *Proc. IFIP TC6 WG6.7 International Conference on Intelligent Networks and Intelligence in Networks*. Chapman & Hall, September 1997.

[6] Thomas Magedanz and Radu Popescu-Zeletin. *Intelligent Networks – Basic Technology, Standards and Evolution*. International Thomson Publishing, 1996.

[7] Ronald L. Rivest and Butler Lampson. SDSI — A simple distributed security infrastucture. Technical report, April 1996.

[8] Lawrence Snyder. Formal models of capability-based protection systems. *IEEE Transactions on Computers*, C-30(3):172–181, March 1981.

[9] Dan S. Wallach, Dirk Balfanz, Drew Dean, and Edward W. Felten. Extensible security architectures for Java. In *Proc. 16th ACM Symposium on Operating System Principles*, Saint-Malo, France, October 1997. ACM.

[10] Dan S. Wallach and Edward W. Felten. Understanding Java stack inspection. In *Proc. 1998 IEEE Symposium on Security and Privacy*, Oakland, California, May 1998.

[11] Edward P. Wobber, Martín Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos operating system. *ACM Transactions on Computer Systems*, 12(1):3–32, February 1994.