Delegation in Distributed Systems: Challenges and Open Issues

Óscar Cánovas[†], Antonio F. Gómez[‡]
[†]*Department of Computer Engineering*
[‡]*Department of Information and Communications Engineering*
University of Murcia
30071 Murcia (Spain)
ocanovas@ditec.um.es, skarmeta@dif.um.es

Abstract

New certificate-oriented access control systems are based on delegation of privileges. In these scenarios, resource guards have an ACL which delegates to some authorization or naming authorities the right to manage the access to the controlled resources. These authorities can issue certificates delegating these permissions to other subordinates authorities, or to specific users. In this way, the generated structure reflects the way the authorization is managed. In this paper we present a survey of different issues related to certificate-based delegation, such as management structures, authority and ownership, anonymity, certificate distribution, and revocation.

Keywords: Delegation, authorization certificates, certificate management, anonymity

1 Introduction

Several authorization proposals have been introduced in the past. The most well-known scheme is the discretionary access control (DAC), where permissions are specified as access control lists (ACLs) within each resource guard. However, organizations change over time, and require a better model of updating access control policies. New DAC mechanisms aim for a more distributed model for management and enforcement of privileges. This decentralized management requires a mechanism for delegating the privileges from the higher levels of an organization to its lower levels. This task can be accomplished making use of digital certificates [5, 7, 8]. In a general sense, a certificate is a record stating some information about the entity the certificate was issued to, and this information may be a role membership statement, or an authorization. Authorization certificates bind a capability to a key, and this capability can be used to determine what the entities are allowed to do.

The main idea behind delegation is that resource guards delegate the authorization-related tasks to specific authorities. These authorities can issue certificates delegating these permissions to other subordinates authorities, or to specific users. In this way, the structure generated reflects the authorization process. Then, the user together with his access request to an object provides a subset of these certificates that proves his permission for that access. Finally, the guard validates the certificates and also check whether they comply with its delegation-based policy.

In this paper we present the different opportunities which can be offered by delegation, especially from a management point of view. Moreover, we try to identify the existing challenges related to this access control model, and we make reference to some related works in order to analyze their contributions. We have structured this analysis according to the topics of management, support for delegation chains, differences between authority and ownership, anonymity, certificate distribution, and revocation. The work is not based on any particular specification for authorization or attribute certificates. We provide our opinion about the open issues to be addressed, and we also mention our contributions to this field.

This paper is organized as follows. Section 2 introduces the management structures created by delegation. Section 3 presents the main differences between authority and ownership. Sec-

tion 4 explains the certificate reduction. Section 5 outlines the problems related to certificate distribution and retrieval. Section 6 contains some ideas about revocation and delegation. Finally, Section 7 makes some concluding remarks.

2 Delegation and management structures

2.1 Privilege management

Authorization certificates provide a mechanism for establishing organizational structures which can be dynamically changed. The certificate structure can reflect the structure of an organization, but in contrast to classic access control lists (ACL) the control of the permissions contained in these certificates is widely distributed [2]. Changes over the authorization policy do not have to be propagated to all the ACLs controlling the resources, and managing the certificates is a simple task since it is distributed among different entities controlling a small subset of permissions. Here, we show an example about how delegation can simplify ACLs, and therefore the resource guards. In §1 we present two ACLs for two different resource guards. The ACL of *guard1* grants two permissions P^1 and P^2 to the public keys K_A and K_B. Other permissions are assigned to K_D and K_E by the ACL of *guard2*.

$$\begin{aligned} ACL(guard1) &= (K_A, P^1), (K_B, P^2) \\ ACL(guard2) &= (K_D, P^3), (K_E, P^2) \end{aligned} \tag{1}$$

Now, a new public key K_C must be authorized by both *guard1* and *guard2* to perform P^2. Using this approach, both ACLs must be modified to include (K_C, P^2). Although this can be considered as simple, things get tricky if we think that this update could involve several distributed ACLs. Consistency, network bandwidth, availability and denial of service attacks are some issues involved in ACL-based solutions. We can redefine the access control policy shown by §1 using delegation. Delegation can be expressed by means of specific labels (*prop*) stating that a particular key can issue new certificates for certain privileges. In §2 the same conditions stated by §1 are expressed using a delegation label and three authorization authorities K_{auth1}, K_{auth2} and K_{auth3}.

$$\begin{aligned} ACL(guard1) &= (K_{auth1}, P^1, prop), (K_{auth2}, P^2, prop) \\ ACL(guard2) &= (K_{auth3}, P^3, prop), (K_{auth2}, P^2, prop) \end{aligned} \tag{2}$$

In order to authorize the public keys to access the resources, the authorities must issue certificates stating the permissions being granted. In §3 we include the certificates necessaries to emulate the authorization policy of §1. We decided omitting validity dates for simplicity.

$$\begin{aligned} &authorization(K_{auth1}, K_A, P^1) \\ &authorization(K_{auth2}, K_B, P^2) \\ &authorization(K_{auth2}, K_C, P^2) \\ &authorization(K_{auth2}, K_E, P^2) \\ &authorization(K_{auth3}, K_D, P^3) \end{aligned} \tag{3}$$

In this way, providing to K_C the permission to perform P^2 only involves the generation of a new authorization certificate (K_{auth2}, K_C, P^2), and it does not require to update any existing ACL. Furthermore, this delegation scheme can be extended in order to truly create management hierarchies reflecting the organizational structure. That is, K_{auth2} could also delegate a subset of permissions $P^{2\prime}$ to K_B by means of an authorization $(K_{auth2}, K_B, P^{2\prime}, prop)$. It makes sense if we think that K_{auth2} could be the public key of a department manager, and K_B the one belonging to a section manager.

2.2 Delegation chains

As we mentioned above, rights can be redelegated to another keys, and these keys can redelegate them to a third one and so on. Therefore, delegation certificates constitute a chain where permissions flow from authorities to subjects (as is commented in [2], in fact delegation does not create chains but graphs).

However, managing certificate chains can become a complex task. Authorization decisions based on long chains are not trivial since, as we will see in section 5, certificate distribution and retrieval can be computationally expensive. Moreover, from an attacker's point of view, delegation chains can reveal too much information about organization structure (authorities, resources, propagation conditions). Consequently, in some scenarios the information contained in these certificates can be considered as sensitive, thus requiring the provision of mechanisms limiting the disclosure.

A technique called *certificate reduction* [2] can overcome some of those drawbacks. If we observe the certificate chain in §4, we can infer the certificate reduction presented in §5.

$$authorization(K_{auth1}, K_i, P^1, prop) \quad (4)$$
$$authorization(K_i, K_j, P^2)$$

$$authorization(K_{auth1}, K_j, (P^1 \cap P^2)) \quad (5)$$

The new certificate is not stating a new permission, it is only a simplified version of the original chain, and it is issued for saving time during the certificate retrieval and verification. Its validity interval will be the intersection of the validity dates contained in the certificate chain.

However, it is worth noting that reduction is not always possible, and sometimes it cannot be performed without losing some features of the original certificates. Certificate reduction must be limited if the certificates forming the chain must be verified using on-line methods (like OCSP). Otherwise, validation of intermediate certificates will not be performed according to the on-line criteria.

2.3 Delegation control

For the sake of simplicity, in the examples that we have shown, delegation control has been performed using a boolean-based approach. There are several alternatives in order to control propagation. In [7], a boolean-based approach is preferred against other proposals based on limiting the delegation depth. They argue that it is unable to predict the proper depth of delegation, and that there is no control on the proliferation of permissions on the width of the tree. Nevertheless, SPKI offers another way to control the propagation by using threshold certificates. For example, K_A would like to propagate a permission P to K_B but retaining the control over the further propagation of P from K_B. By using a boolean control method, K_A cannot enforce this requirement. However, K_A can issue a certificate to the subject $(2 - of - 2)(K_B)(K_A)$, denying K_B the propagation of P without the intervention of K_A. This proposal, not only transforms K_A in a *central* authority, but also does not impede a confabulation of principals which obtained a certificate of the form $(2 - of - 2)(K_{any})(K_A)$ for the same permission.

In [4], a fine-grained mechanism for constrained delegation is presented. Limitation is based on regular expressions establishing the organizational subtree that can be included in the delegation chain. Only the certificates issued for those entities included in the subtree will be considered as valid. Our opinion is that this proposal is a valuable step toward a better control of delegation, although can be inefficient in some scenarios where the organization structure is very dynamic. The main drawback is that establishing a new leaf node in the subtree can involve the modification of all the delegation certificates from the root to the node in order to reflect the new propagation policy.

3 Authority and ownership

One of the topics that has raised a good deal of controversy is: *may a delegator also exercise the permissions being managed by himself?*.

There is no general agreement about this issue and some authors believe that a delegator can also issue a new certificate for his own temporary keys in order to delegate himself the rights he cannot exercise. However, our belief is that a security administrator may or may not be part of the scope of his administration, and appropriate mechanisms should be provided in order to limit his authority.

Some authors make a clear distinction between having a permission and being able to manage a permission [13]. In general, the term *authority* makes reference to the creation and delegation or permissions, and the term *privilege* is used to cover both authority and permission. However, specification of independent policies for managing and using permissions is an open research field.

On the other hand, it is worth noting that the act of signing an authorization certificate does not invalidate any existing certificates. In this way, the issuer does not lose any of his permissions. *Transfer* is far difficult to implement than delegation since it involves that the revocation of previous privileges and the issuance of new ones must be executed as an atomic operation. Moreover, since authorization certificates only support policies where access rights grow monotonically, it is impossible to verify that an entity does not have a particular permission since negative statements are not allowed.

4 Anonymity

Section 2.2 introduced the problems derived from the disclosure of sensitive information contained in the certificates forming a chain. In fact, the certificate structure shows the relationships among the keys, and the keys might be easily mapped to real users when names are used.

In [3] two proposals are presented for preventing the tracking of keys: temporary keys and reduction. In this section we are going to present the basis of these proposals, we provide some refinements, and we explain their limitations.

4.1 Temporary keys

In order to impede tracking the keys, users can redelegate their rights to self-generated temporary keys, which will be used every time the user is requesting to perform an operation on a resource. In this way, the original public key, probably related to an identity certificate, can be hidden. In §6 we show a certificate chain where the user K_U delegates a subset of permissions to a self-generated temporary key K_T.

$$\begin{gathered} authorization(K_{auth1}, K_U, P^1, propagate) \\ authorization(K_U, K_T, P^{1\prime}) \\ where\ \ P^{1\prime} \subseteq P^1 \end{gathered} \tag{6}$$

It is worth noting that the certification chain will be valid only if K_U has the privilege to further delegate P^1 (or a subset). In some application environments, such as e-commerce systems, redelegation is not allowed since acquiring access rights can involve some kind of charge.

Furthermore, the support for temporary keys is hard to implement when techniques for constrained delegation are being used. As we mentioned in section 2, constraints are based on the specification of valid previously-known subtrees. Temporary keys are dynamically generated, and their values cannot be predicted.

We propose a solution to this problem in §7. As we can see, delegation is allowed to members of group G, which is defined by the entity K_M. In order to enable a temporary key, K_M must consider K_T as a member of G.

$$\begin{gathered} authorization(K_{auth1}, K_U, P^1, prop(K_M\$G)) \\ authorization(K_U, K_T, P^{1\prime}) \\ where\ \ K_T \in K_M\$G \end{gathered} \tag{7}$$

Although this solution is valid, avoiding further modification on the delegation constraints and certificates, it involves the registration of every temporary key generated by the users. Therefore, a strong authentication of members of G and a mechanism for unique traceable identifiers must be provided. Consequently, using temporary keys for preventing the track of public keys must be subjected to strong authentication of those keys.

4.2 Reduction and trusted reducers

In the previous section we presented temporary keys as a mechanism to hide the activity of the users' private keys. However, sole use of temporary keys does not hide the intermediate keys in a chain of certificates. In §6 and §7, K_U is still included in the chains. However, as we presented in §5, the reduced certificate contains only the first key in the chain (which verifies the certificate), and the last one.

Here we present a scheme which does not requires the participation of the root key to perform the reduction. We introduce the concept of *trusted reducers* as specific entities which are authorized to generate certificate reductions on behalf of a set of root authorities for a particular set of privileges. Trusted reducers can be set up to manage small sets of permissions, and can release the root keys from the task of reducing long chains of certificates.

Trusted reducers can be introduced as valid authorities using two approaches. In §8 we show an ACL-based approach and in §9 we present an authorization-based alternative.

$$ACL(guard1) = (K_{root}, P^1, prop), (K_{redcr}, P^{1\prime}, prop) \tag{8}$$

$$\begin{gathered} ACL(guard1) = (K_{root}, P^1, prop) \\ authorization(K_{root}, K_{redcr}, P^{1\prime}, prop) \\ where\ \ P^{1\prime} \subseteq P^1 \end{gathered} \tag{9}$$

The ACL-based approach requires adding the public keys of the trusted reducers to the ACLs involved. When the number of controllers and reducers is high, this option is not suitable. On the other hand, the authorization-based approach presented in §9 makes use of authorization certificates to introduce the reducers as trusted entities. The certificate issued by K_{auth} to K_{redcr} authorizes the reducer to generate new reductions for permissions contained in $P^{1\prime}$. In contrast with the ACL-based approach, the reduced certificate generated by the trusted reducer is not enough to gain access to the resources controlled by the guards since it does

not constitutes a direct authorization from the public key contained in the ACL. Consequently, the delegation certificate for the trusted reducer must be also presented. Certificate reduction by means of trusted reducers is a mechanism offered by our distributed credential management system [6].

5 Certificate retrieval

Once the certificates have been generated, some of them will be made available for the rest of users, and a subset will be protected if they contain sensitive information. Therefore, retrieving the certificates necessaries to check whether an access request must be granted is not a trivial task. First, since certificates are widely distributed among different issuers, repositories and users, we need to discover the location of these entities (generally named suppliers). Second, since some certificates can include sensitive information, it is needed to provide access control methods to protect them [14]. As we will see, there are several alternatives to query the suppliers (user-directed, guard-directed and distributed among the suppliers).

We can find in the literature several proposals for certificate retrieval [1, 9, 11]. In this section we are going to identify the main issues to be solved.

5.1 The *hidden membership* problem

In §10 we present what we called the *hidden membership* problem, that is, the task of determining whether a particular public key is a member of an authorized group or role. The ACL of $guard1$ specifies that only members of group $staff$ can exercise the permission P, which is the one being requested by K_U.

$$ACL(guard1) = (K_{root}\$"staff", P) \quad (10)$$
$$K_{root}\$"staff" = \{K_{level1}\$"secA", K_{level1}\$"secB"\}$$
$$K_{level1}\$"secA" = \{K_{level2}\$"dept1", K_{level2}\$"dept2"\}$$
$$K_{level2}\$"dept2" = \{K_T, K_U, K_V\}$$

By means of this example, we can check that K_U is indeed a member of group $staff$ since it is a member of group $dept2$ defined by K_{level2}. However, determining membership can involve an exhaustive analysis of the whole tree representing the relationship among the existing groups. The problem can even become more complex if the relationship does not constitutes a tree but a cyclic graph.

5.2 The *hidden permission* problem

The *hidden permission* problem is similar to the one commented above. In §11 we present an ACL where $guard1$ delegates the authority about P to an entity K_{root}. In this example, K_U is requesting the operation P^4, which is a subset of P. Groups are defined in the same way presented in the previous example.

$$ACL(guard1) = (K_{root}, P, prop) \quad (11)$$
$$auth(K_{root}, K_{level1}, P^1, prop) \ where \ P^1 \subseteq P$$
$$auth(K_{root}, K_{level1}\$"secA", P^2) \ where \ P^2 \subseteq P$$
$$auth(K_{level1}, K_{level2}\$"dept1", P^3) \ where \ P^4 \subseteq P^3 \subseteq P^1$$
$$auth(K_{level1}, K_{level2}\$"dept2", P^4) \ where \ P^4 \subseteq P^1$$

By means of this example, we can check that K_U is authorized since it is a member of group $dept2$ defined by K_{level2}, and that group has been authorized to exercise the permission P^4. As we commented in the previous section, discovering this path can involve the analysis of several certification chains. In fact, in this example we might find an alternative authorization chain if P^4 was a subset of P^2. That is, K_U will be authorized since it is a member of $dept2$ which is a subgroup of $sectionA$. In conclusion, an efficient certificate chain discovery method should manage both group membership and an algebra for reasoning about privileges.

5.3 Approaches for certificate retrieval

Discovery of delegation chains can be performed using different approaches. Traditionally, the requestor was responsible for obtaining the needed certificates from public repositories or smartcards. Nowadays, a client can get benefit in several ways from using a server to acquire certificates as input to the validation process [11]. In this context, the client is relying on the server to interact with repositories to acquire the data that the client would otherwise have to acquire using repository access protocols.

As we showed in the previous section, discovery can involve several authorities or repositories, which can be located in different servers,

and can be accessed more than once for the same query. This query can be directed by the guard, or can be performed using a distributed-approach where several suppliers are both requestors and suppliers, an architecture which has been proposed by several authors [15]. According to §11, we might consider that the certificates issued by K_{root} are stored in a server (the supplier), which is different from servers containing the certificates issued by K_{level1} or K_{level2}. In this way, a certificate chain discovery request sent by $guard1$ to the supplier of K_{root} can be partially forwarded to the other suppliers in order to obtain some elements of the chain.

In our opinion, much research efforts must be focused on optimizing this type of distributed resolution. One of the main problems is how to provide some control for redundancy of queries, since the delegation graph can contain repeated references to the privileges or groups defined by a particular supplier, and in absence of some kind of coordination data, some queries can be redundant. Additional issues that must be addressed are the disclosure of sensitive information and the support for certificate caching and management of revocations.

6 Revocation of delegation certificates

Authorization certificates can be revoked because the privilege given in the certificate does not hold any longer. Revocation is often considered regarding the simplest form of revocation, which makes a certificate invalid from the time the revocation is performed for all times in the future (a classification of revocation schemes is provided in [10]). In order to support revocation to have prospective and retrospective effects, it is necessary to distinguish between the time a certificate is revoked and the time for which the associated privilege is created.

In [12], the authors propose some mechanisms in order to reasoning about revocation according to propagation and dominance. Those mechanisms make use of certificates represented as:

$$authorization(K_{auth}, K_U, P[I], time-stamp, id) \quad (12)$$

The time-stamp makes reference to the time the privilege is created, and I is the interval for which the privilege P holds. Time-stamps are used in order to avoid that subsequent certificates could be considered as valid once the issuer has lost the authority, which might be accomplished by forging the time interval I. On the other hand, revocations are represented as:

$$revocation(K_{auth}, id, [I], time-stamp) \quad (13)$$

They contain the id of the certificates which are subject of the revocation, and an interval $[I]$ called the *disabling interval*. The main intention behind the disabling interval is the ability to revoke privileges that have been granted in the past.

For example, a disabling interval with a *not-before* date equal to the date contained in the time-stamp is used to revoke a particular certificate, but it will not affect any other existing certificates. This can be useful when a section chief is replaced by other person, since we want to maintain the existing authorization certificates, despite that person will not be able to manage further privileges. However, if we think that this person has abused his authority we need to delete not only his authority but also all those privileges that were delegated by him. This can be done using a disabling interval with a *not-before* date placed in the past.

In our opinion, these mechanisms have several drawbacks which can limit their deployment in a typical distributed system. First, it requires the use of trusted time stamps provided by a centralized service. Delegation by means of authorization certificates is an inherently distributed system which can be in conflict with this type of services. In fact, most of the authorization certificates are supposed to be generated in an off-line manner. On the other hand, revocation affects the certificates being identified by *id*. When the same privilege has been stated by different certificates, revoking one certificate does not disable the privilege itself. We believe that revocations should make reference to the privilege being revoked, not to a serial number. Thus, not only the time interval I must be included in the revocation but also the privilege P (or a subset of P).

7 Conclusions

In this paper we have presented the different opportunities provided by a decentralized approach for access management. We have identified the existing challenges related to this model,

especially about management, support for delegation chains, differences between authority and ownership, anonymity, certificate distribution, and revocation. We have also outlined some interesting future research activities to be done, and we have also introduced our contributions to this field.

References

[1] T. Aura. Fast access control decisions from delegation certificate databases. In *Proceedings of 3rd Australasian Conference on Information Security and Privacy ACISP'98*, volume 1428 of *Lecture Notes in Computer Science*, pages 284–295. Springer, July 1998.

[2] T. Aura. On the structure of delegation networks. In *Proc. 11th IEEE Computer Security Foundations Workshop*, pages 14–26, MA USA, June 1998. IEEE Computer Society Press.

[3] T. Aura and C. Ellison. Privacy and Accountability in Certificate Systems. Technical report, Helsinki University of Technology, April 2000. Research Report A61.

[4] O. Bandmann, M. Dam, and B. Sadighi. Constrained Delegations. In *Proceedings of 2002 IEEE Symposium on Security and Privacy*, 2002. To be published.

[5] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. *The KeyNote Trust Management System Version 2*, September 1999. Request For Comments (RFC) 2704.

[6] O. Cánovas and A. F. Gómez. A Distributed Credential Management System for SPKI-Based Delegation Systems. In *Proceedings of 1st Annual PKI Research Workshop*, pages 65–76, 2002.

[7] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI certificate theory*, September 1999. Request For Comments (RFC) 2693.

[8] S. Farrel and R. Housley. *An Internet Attribute Certificate Profile for Authorization*. Internet Engineering Task Force, April 2002. Request for Comments (RFC) 3281.

[9] C. A. Gunter and T. Jim. Policy-directed certificate retrieval. In *Proceedings of Software - Practice and Experience*, pages 1609–1640, 2000.

[10] A. Hagstrom, S. Jajodia, F. Parisi, and D. Wijesekera. Revocation: a classification. In *Proceedings of the 14th IEEE Computer Security Foundation Workshop*. IEEE Press, 2001.

[11] D. Pinkas and R. Housley. *Delegated Path Validation and Delegated Path Discovery Protocol Requirements*. Internet Engineering Task Force, May 2002. draft-ietf-pkix-dpv-dpd-req-05.txt.

[12] B. Sadighi and M. Sergot. Revocation Schemes for Delegated Authorities. In *Proceedings of IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002. To be published.

[13] B. Sadighi, M. Sergot, and O. Bandmann. Using Authority Certificates to Create Management Structures. In *Proceeding of Security Protocols, 9th International Workshop*, April 2001.

[14] K. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Proceedings of Network and Distributed System Security Symposium*, April 2001.

[15] J. Vollbrecht, P. Calhoun, S. Farrell, L Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. *AAA Authorization Framework*. Request For Comments (RFC) 2904, August 2000.