Chapter 21

A HYBRID PKI MODEL: APPLICATION TO SECURE MEDIATION

Joachim Biskup and Yücel Karabulut

Abstract For distributed computing systems, specification and enforcement of permissions can be based on a public key infrastructure which deals with public keys for asymmetric cryptography. We review previous approaches and classify them as based on trusted authorities with licencing and dealing with free properties (characterizing attributes including identities), e.g. X.509, or based on owners with delegation dealing with bound properties (including capabilities), e.g., SPKI/SDSI. These approaches are extended and integrated into a hybrid model which uses protocols to convert free properties into bound properties. Furthermore, we unify licencing and delegation by introducing administrative properties. Secure mediation is taken as an example for a wide range of potential applications.

Keywords: Distributed system, permission, public key infrastructure, trusted authority, licencing, free property, owner, delegation, bound property, capability, administrative property, confidentiality, secure mediation

1. Introduction

The proper administration of IT-systems requires to specify which clients are allowed to access which services, and to effectively and efficiently enforce such specifications. In advanced *distributed systems* [16], some basic assumptions of traditional access control scheme are not longer valid. In particular, a client may not be registered in advance by an identifying name at the site of a server, and thus he may be unknown to a server at the time of a request. In order to overcome these and related difficulties a diversity of proposals has arised. While all proposals exploit cryptography, some of them use symmetric cryptographic mechanisms, like Kerberos [11], and others rely on asymmetric cryptography, like X.509 [9] and SPKI/SDSI [5, 6].

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35697-6_26

E. Gudes et al. (eds.), *Research Directions in Data and Applications Security*

In this paper and its full version[1] we deal with approaches to specify and to enforce *permissions* of clients on remote servers which are based on *asymmetric cryptography*, see for instance [3, 4, 13]. More specifically, we aim at presenting the following contributions:

- We design a hybrid model for a *public key infrastructure*, PKI for short, to be used for specifying and enforcing permissions in distributed computing systems.
- The hybrid model integrates and unifies previous approaches.
- The design meets the requirements for secure mediation [1], but it appears to be also worthwhile for a broad spectrum of applications.

The design of our approach of *secure mediation* is shortly outlined as follows. A client proves his eligibility to see a piece of information by a collection of so-called personal authorization attributes. Independently of a specific request and a specific source, a personal authorization attribute has been assigned to the client by some trusted authority who certified such an assignment within a credential[2] A source always receives a mediated request to deliver some information together with a set of credentials stemming from the pertinent client. Then the source decides on the permission of the request by evaluating the credentials and the contained personal authorization attributes with respect to its *confidentiality* policy. In case of an allowance, the returned data is encrypted with the public keys found in the credentials on which the permission decision has been based. Thus the returned data can only be decrypted by that client who has proven his eligibility by showing an appropriate collection of personal authorization attributes.

We argue that similar scenarios can be found in distributed systems not only for mediation but for many other applications as well. The following points are crucial: a) A client is represented by (one of) his public key(s) and characterized by the assigned attributes. b) A trusted authority assigns attributes to public keys. c) A server follows a security policy that is expressed in terms of attributes.

The hybrid model for a PKI presented in this paper constitutes a far more detailed elaboration of the crucial points. The model is adaptable for security policies concerned with confidentiality and integrity if suitable challenge-response mechanisms are implemented. Figure 1 visualizes some of the details of the model to be explained in the following sections.

[1] `http://ls6-www.cs.uni-dortmund.de/issi/publications/2002.html.en`.

[2] Here we still use the term credential as in [1]. In the present paper, we would prefer to call the document a certificate in the sense of Section 2.4. However, the document is also used like a credential in the sense of Section 2.5. This dual use is further discussed in Section 5.

2. Real World and Virtual Views

In a distributed system, a specific entity cannot directly see the other entities. In some sense, the (real) world of the other entities is hidden behind the interface to the communication lines. Surely, the entity can send and receive messages to and from this hidden (real) world, and based on these messages the entity can produce a virtual view which is actually visible to the entity. As a consequence, security policies and permission decisions are solely grounded on the locally available visible view on the global (real) world. This exposition is elaborated as follows.

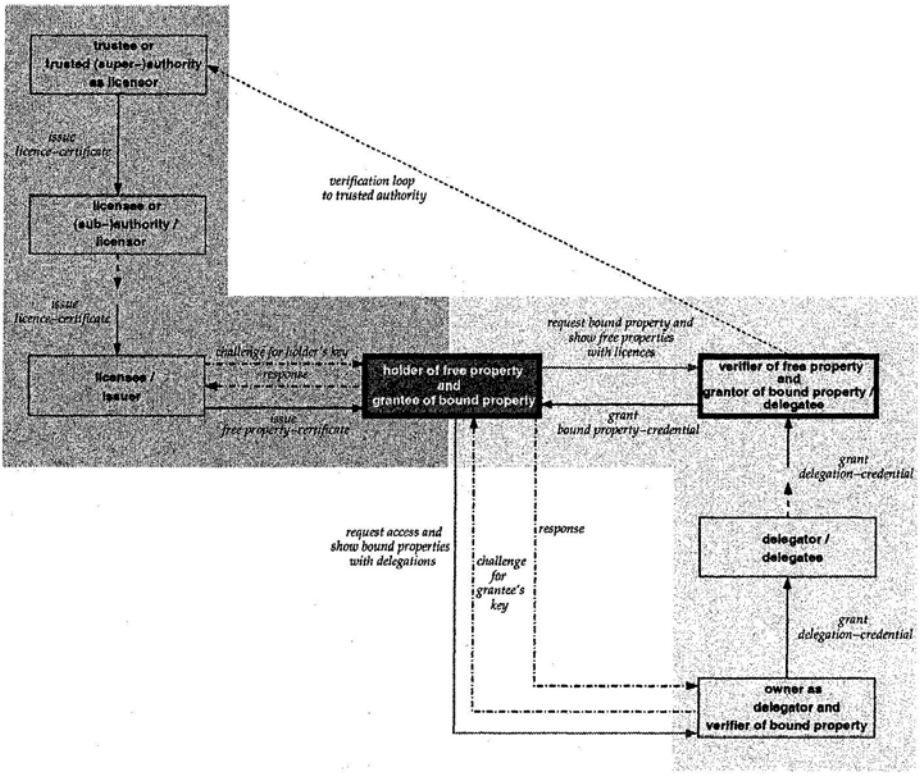

Figure 1. Outline of an instance of the hybrid model for a PKI.

2.1 Property Assignments and Documents

An identifiable *entity* in the (real) world might be an *individual*, a *computer* or something similar that can act in the distributed system. An entity may *enjoy* various *properties* which might be relevant for security policies and permission decisions. In most cases, such properties are *assigned* to an entity by another entity.

In general, neither the entities themselves nor their properties are visible to other entities. Thus we need a notifiable representation of such circumstances. In a public key infrastructure, PKI, entities are represented by keys for asymmetric cryptography. More precisely, an entity is uniquely represented and distinguishable from other entities by one or more pairs of *secret* and *public keys*: while the entity must keep the secret part strictly hidden, the entity uses the matching public part as a visible surrogate for itself. From the perspective of the visible virtual views, these surrogates are called *principals*. For the sake of conciseness, here we assume that each entity has exactly one key pair for digital signatures and one key pair for ciphers. Hence, here a principal is specified by a *public key for verification*, which is good for proving the integrity and authenticity of messages from the represented entity, and by a *public key for encryption*[3], which is good for sending confidential messages to the represented entity.

Then, a property assignment to an entity in the (real) world is *presumably captured by* a *digital document* in the visible virtual world. Such a document is called a *certificate* or a *credential* depending on the details explained below. In all cases, the document has at least the following fields: a) a *subject* field which contains the principal which visibly represents the entity under consideration; b) a *content* field which textually describes the assigned property (where, depending on the the concrete format, we can also allow compound properties); c) a field for a *responsible agent*: this field contains the principal which visibly represents the entity that is responsible for the property assignment and that has generated and digitally signed the document; d) a *signature* field which contains a digital signature for the document: the signature is valid iff it can be verified with the responsible agent's public key for verification, i.e. if it has been generated with the matching secret key for signing[4].

Depending on the specific format for digital documents, usually additional fields are needed, for example: e) a *type* field which indicates the meaning of the document and provides further technical hints on how to process the document; f) a *validity*[5] field which indicates the validity period of the document.

The relationships of *presumably_captured_by* are ideal claims that do not necessarily hold. A specific entity has to evaluate his individual trust about such an ideal claim. The treatment of issues regarding trust evaluation is started in Section 2.3.

[3] In practice, for efficiency reasons, usually a hybrid encryption scheme is exploited.

[4] In this paper, for the sake of simplicity, we assume that keys do not expire.

[5] Accordingly, again for the sake of simplicity, we also assume that digital documents are always usable and are never revocated.

2.2 Properties

In the following we classify properties of entities (and thus contents of certificates and credentials) according to two aspects. The first aspect deals with properties which characterize entities with security policies in mind. We distinguish two kinds of such *characterizing properties*:

- A *free property* is intended to express some feature of an entity by itself (e.g. personal data, a technical detail, a skill, an ability, ...). Other entities may possibly base their security policies and permission decisions on shown free properties, but in general these entities have not expressed any obligation whether or how to do so. In particular, enjoying a free property usually does not entail a guarantee to get the permission for a specific service.
- A *bound property* is intended to express some relationship between a client entity and another entity which might act as a server (e.g. a ticket, a capability, ...). Typically, such a server has declared in advance that it will recognize a shown bound property as the permission to use some of its services. In particular, enjoying a bound property entails some kind of promise to get a specific service as expressed in the relationship.

We are mainly interested in the analysis of how an entity can exploit its free properties in order to acquire bound properties, or in other words, how a client entity can convince a server entity to grant him the permission to use some services. This analysis is given in Sections 3 and 4.

The second aspect deals with the administration of characterizing properties. More specifically, the assignment of characterizing properties to entities is regulated by corresponding administrative properties which must be hold by the entity that is responsible for such an assignment. We distinguish two kinds of *administrative properties*: a) The *administration status* expresses whether an entity can make assignments in its own right or only on behalf of another entity. In the former case, the assigning entity has the status of an *origin* (for the administered characterizing property), and in the latter case, it is considered as a *dependant* (for the administered characterizing property). The relationship between an origin and its direct or indirect dependants has to be suitably expressed, again by appropriate administrative properties. b) The *administration function* expresses the following roles. In the role of a *distributor* an entity can responsibly assign the corresponding characterizing property to a qualifying entity. In the role of an *intermediate* an entity can establish new dependants.

Though administrative properties look quite similar for free properties and for bound properties, there are also some differences explained in Section 2.4 and Section 2.5 below. Moreover, usually different terms are used. For free properties, an origin is called a *trusted authority* , or *trustee* for short, and a dependant a *licensee*; for bound properties, an origin is seen as an *owner* (of a service) and a dependant as a *delegatee*. Accordingly, a distributor is an *issuer*

or a *grantor*, respectively, and an intermediate is a *licensor* or a *delegator*, respectively.

We emphasize that all (potentially hidden) assignments of properties to entities, whether they refer to characterizing or administrative properties, have to be represented in the visible virtual world of certificates and credentials.

2.3 Evaluating Trust about Ideal Claims

The very purpose of the *administrative properties*, the corresponding certificates and credentials and the gathering of them into appropriate chains or dags is to provide a reliable foundation for trust evaluations, as explained in the following.

Conceptually, permission decisions are intended to be based on *characterizing properties* of entities appearing as clients. However, since property assignments occur in the hidden (real) world, actually the permission decisions have to be based on available and visible digital documents the contents of which mean the respective characterizing properties. Consider any such document as a so-called "*main document*" from the point of view of an entity entitled to take a permission decision. Then, the question arises whether the literal meaning of the content is indeed valid in the hidden (real) world, i.e., whether the digital document *captures* a "real" property assignment. This question is answered using further "supporting documents" the contents of which mean appropriate administrative properties. However, for each of these supporting documents the same question arises: is the literal meaning of the content indeed valid in the hidden (real) world? Thus we are running into a recursion.

In order to be helpful, the "main document" and its "supporting documents" should form a *directed acyclic graph (dag)* with respect to their relationships concerning *support*. As a special case, we just get a *chain*. For example, Figure 1 shows two chains, one for a free property-certificate as the "main document", and another one for a bound property-credential as the "main document".

In any case, the ultimate trust about all ideal claims pertinent to the documents relies on the *nonsupported* documents referring to *origins*. Rather than using explicit documents for origins, the evaluating entity often just decides on its own discretion that it wants to treat the responsible agent of a "supporting document" as denoting an origin. This situation is shown in Figure 1, where no explicit documents occur for origins, i.e. the trustee and the owner, respectively. A main difference between administrative properties for free properties and for bound properties stems from different treatments of and assumptions on origins and how origins determine their dependants.

2.4 Model of Trusted Authorities and Licencing

Following and extending the basic approach of X.509 [9], free properties and the corresponding certificates are handled by trusted authorities using licencing. The upper part of Figure 1 visualizes an instance of the general situation. In the simplest case (where in Figure 1 all entities on the left side are identified), an entity acts and may be considered as a "trusted authority" or trustee for a free property.

In *acting*, such an entity assigns a free property to another entity and, accordingly, *issues* a suitable certificate about this assignment.

In *being considered* as a "trusted authority", such an entity is later evaluated by a further entity (verifier). This further entity, seeing the issued certificate, may decide to treat the issuing entity as an origin for the free property. Thus, after having verified the signature of the certificate, the further entity concludes his trust evaluation whether or not the certificate captures the corresponding property assignment. The crucial point is that in general the issuer and the holder of the certificate are different from the entity which inspects the certificate.

In more advanced cases (as shown on the left side of the upper part of Figure 1), additionally *licencing* is used. Then, basically, an entity engaged in licencing does not assign free properties on its own right. Rather it has to be *explicitly licenced* to do so. More specifically, some other entity, acting as a *licensor* and trusting the *licencee*, has expressed by a licence-certificate that the licensee should be entitled to assign a specific free property, i.e. to issue corresponding certificates.

Additionally, licencing can be transitively organized: a licensor can express his trust in a licensee to act as a licensor in turn, again by a suitable licence-certificate. It might also be the case that a licensee needs to be trusted by several licensors.

2.5 Model of Owners and Delegations

Following and extending the basic approach of SPKI/SDSI [5, 6], bound properties and the corresponding credentials are handled by owners of services using delegation. The lower part of Figure 1 visualizes an instance of the general situation. In the simplest case (where in Figure 1 all entities on the right side are identified), an entity acts as an owner of its services (resources) offered to other entities and is explicitly addressed by these other entities.

In *acting*, such an entity assigns a bound property to another entity and, accordingly, *grants* a suitable credential about this assignment.

In *being addressed* as an owner, such an entity is afterwards contacted by some further entity that requests the offered service. The request comes along with showing a credential in order to get the permission to access the service.

The addressed owner inspects the credential whether he himself has granted it, i.e., he checks the signature with his public key for verification. In the positive case, the owner interpretes the bound property meant by the content of the credential according to his security policy. If the bound property is interpreted as a traditional *capability*, then the owner immediately allows access to the requested service provided the capability is good for the service. Thus, for a capability the essential permission decision has been taken before at the time of granting the credential. If the bound property is interpreted as a more general *bound authorization attribute*, then the owner might base his final permission decision not only on the shown bound authorization attribute granted some time before, but also on additional factors which are not directly encoded in the credential. It is important to note that, so far and neglecting misuse of stolen credentials, the owner needs no trust evaluation except that he is willing to accept his own signatures. The crucial point here is that the grantor of the credential is identical with the entity which afterwards inspects the credential.

In more advanced cases (as shown on the right side of the lower part of Figure 1), additionally *delegation* is used. Then, basically, an entity engaged in delegation does not assign bound properties on its own right and for its own services. Rather it is acting on behalf and in explicit delegation of a different owner. More specifically, some other entity, acting as a *delegator* and trusting the *delegatee*, has expressed by a delegation-credential that the delegatee should be entitled to assign a specific bound property, i.e. to grant corresponding credentials.

Additionally, delegation can be transitively organized: a delegator can express his trust in a delegatee to act as a delegator in turn, again by a suitable delegation-credential. It might also be that a delegatee needs to be trusted by several delegators.

In this model of owners and delegation, if any bound property-credential is used as a "main document" then it is shown to the owner of the services to which the bound property refers. Moreover, any dag of supporting delegation-credentials contains just one origin, namely the owner himself.

3. Converting Free Properties into Bound Properties

Why is a grantor, whether the owner of a service himself or any of his delegatees, willing to assign a bound property to an entity and to grant a corresponding credential, i.e., to express a possibly conditional permission to access a service? The general answer is that the grantor follows a *security policy* that maps free properties on bound properties. More precisely, the policy specifies *which set of free properties an entity has to enjoy in order to get a bound property assigned.*

Rephrased more technically in terms of the visible world of digital documents, this means the following: the security policy specifies *which free poperty-certificates as "main document" together with which "supporting licence-certificates" are accepted in order to get which bound property-credential granted.*

The middle part of Figure 1 visualizes the situation. The entity on the right is the grantor following a security policy. The entity in the center requests a promise for a permission, i.e. a bound property. The grantor a) verifies the submitted free property-certificates with the supporting licences, b) extracts the contents of the free property-certificates and interpretes them as free properties, c) applies his security policy on the extracted free properties, and d) finally, if all checks have been successfully completed, grants a bound property-credential where the subject (grantee) is the same as in the submitted free property-certificates.

4. Full Hybrid Model

For succinctness, we summarize previous considerations without explaining additional details and special cases. An *instance* of the full hybrid PKI model consists of overlapping components of three kinds: a) trusted authorities (also called trustees) and licencees for and a holder of a free property (see Section 2.4), b) an owner and delegatees for and a grantee of a bound property (see Section 2.5), and c) a holder of free properties and a grantor of a bound property (see Section 3).

Components of the first two kinds can form so-called loops. A verifier for a free property in a component of a first kind *has to close* the verification loops to the trusted authorities in order to found his trust. More precisely, closing the loop means here that the verifier *decides* which entities he wants to trust and to accept as origins. A granteee of a bound properties *automatically closes* a loop when addressing the pertinent owner. A component of the third kind is used as a link between a component of a first kind and a component of the second type. The link identifies the verifier of the former component with the grantor of latter component, and the holder of the first component with the granteee of the latter component. In some special cases a linked component may appear to be degenerated. This also allows to subsume traditional access right management under the hybrid model. The links are the important feature that makes the hybrid model highly flexible and adaptable for many applications. Figure 1 shows a simple example in which two loops for chains are linked.

5. Secure Mediation

We are using the hybrid PKI model to implement an extended version of our design of secure mediation [1]. The basic design, as sketched in the intro-

duction, only exploited the PKI model of trusted authorities and licencing: We assumed that trusted authorities and their licencees for a special kind of free properties, called personal authorization attributes, were already in operation. So far, there were no explicit bound properties since mediators and sources executed permitted requests immediately.

The motivation to extend the basic design and to elaborate the hybrid PKI model has mainly originated from a conceptual challenge and a corresponding implementation task. Both the challenge and the task are related to our view that all entities are implemented as autonomously cooperating software agents which base their communications on KQML [7] and their data exchanges on CORBA [12].

The conceptual challenge concerns how remote and autonomous entities can agree on a common understanding of "personal authorization attributes". On the one hand, trusted authorities assign personal authorization attributes as *free properties*, in principle without knowing their later usage. And on the other hand, sources and mediators independently define their security policies in terms of personal authorization attributes, and thus implicitly treat them like *bound properties* (in a degenerate case of the model of owners without explicitly granting capabilites with the corresponding promise of a service). Also clients wish to be assisted by a mediator to assemble appropriate properties within suitable digital documents in order to receive the information services they want. Thus mediators should participate in the common understanding of personal authorization attributes, too.

In general, for any specific case we need an additional explicit *mediation process* that helps to interpret a free property as a bound property for some service and to arrange the corresponding conversion, as explained in Section 3. Then a mediating agent is acting both as verifier of free properties and as a delegatee and grantor of a bound property on behalf of a source which is the owner of an information service, as visualized by Figure 1. In the basic design of secure mediation, the sources themselves still took the burden of this task.

The implementation task concerns the software agents for secure mediation. For any singular mediation request, the specific entites directly involved perform a fixed role as client, mediator or source, respectively, and those entities which possibly indirectly contribute perform a fixed role as (trustee or licensee/licensor or issuer) or (owner or delegatee/delegator or grantor), respectively. In general, however, all entities should be able to act in any of these roles during their lifetime. Thus for our prototype implementation, we need a *core functionality* to be made available for all agents. In particular, all agents should be enabled to deal with free and bound properties and to convert the former into the latter.

Accordingly, we provided an *agent PKI framework* offering basic PKI services, such as XML-encoding and issuing of certificates/credentials. The agent

functional core includes functional modules to process and to evaluate the content (e.g. an OQL query) of the KQML-performatives. The communication interface of the agent core includes now new KQML-performatives to handle the external actions and reactions among agents involved in the hybrid PKI model.

6. Related Work and Conclusions

Most of the works [1, 2, 8, 10, 14, 15, 17] investigating the application of certificate/credential-based access control treat both PKI models discussed in Section 2.4 and 2.5 as competing approaches and base their work on a single PKI model. Even some of these works abstract from any particular PKI model. Previous approaches for defining and applying a PKI are classified as based either on trusted authorities with licencing or on owners with delegations. In this paper, we identified the similarities and the differences of these approaches. We argue that many applications require to use and to link both kinds of PKI. Accordingly, we outlined a *hybrid PKI model* which unifies and extends the previous approaches. The *core functionality* of the PKI hybrid model, as needed for an extended version of our approach to *secure mediation*, has been implemented using XML-encoded documents.

References

[1] C. Altenschmidt, J. Biskup, U. Flegel and Y. Karabulut. Secure mediation: Requirements, design and architecture. *Computer Security* (to appear).

[2] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *Proceedings of the Seventh ACM Conference on Computer and Communications Security*, pages 134–143, Athens, Greece, 2000.

[3] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, Cambridge, Massachusetts, 2000.

[4] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030-1044, 1985.

[5] D. Clarke, J. Elien, C. Ellison, M. Fredette, A. Morcos and R. Rivest. Certificate chain discovery in SPKI/SDSI. *Computer Security*, 9(4):285-322, 2001.

[6] C. Ellison. SPKI/SDSI certificates. `http://world.std.com/~cme/html/spki.html`, 2002.

[7] T. Finin, Y. Labrou and J. Mayfield. KQML as an agent communication language. In J. Bradshaw (ed.), *Software Agents*. MIT Press, Cambridge, Massachusetts, 1997.

[8] A. Herzberg and Y. Mass. Relying party credentials framework. In D. Naccache (ed.), *Topics in Cryptology – CT-RSA 2001, The Cryptographer's Track at RSA Conference (LNCS 2020)*, pages 328–343, San Francisco, California, 2001.

[9] ITU-T Recommendation X.509: The directory – Public key and attribute certificate frameworks, 2000.

[10] I. Lehti and P. Nikander. Certifying trust. In *Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography*, pages 83–98, San Diego, California, 1998.

[11] S. Miller, B. Neuman, J. Schiller and J. Saltzer. Section E.2.1: Kerberos authentification and authorization system. *M.I.T. Project Athena.* Technical Report, Cambridge, Massachusetts, 1987.

[12] Object Management Group. The CORBA Security Specification. `www.omg.org/cgi-bin/doc?formal/2002-03-11`, 2002.

[13] P. Samarati and S. de Capitani di Vimercati. Access control: Policies, models and mechanisms. In R. Focardi and R. Gorrieri (eds.), *Foundations of Security Analysis and Design (LNCS 2171)*, pages 137–196. Springer, Berlin, Germany, 2000.

[14] K. Seamons, W. Winsborough and M. Winslett. Internet credential acceptance policies. In *Proceedings of the Workshop on Logic Programming for Internet Applications*, Leuven, Belgium, 1997.

[15] K. Seamons, M. Winslett, T. Yu, B. Smith, E. Child and J. Jacobsen. Protecting privacy during on-line trust negotiation. In *Proceedings of the Second Workshop on Privacy Enhancing Technologies*, San Francisco, California, 2002.

[16] A. Tanenbaum and M. van Steen. *Distributed Systems.* Prentice-Hall, Upper Saddle River, New Jersey, 2002

[17] W. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson and A. Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of the 8th USENIX Security Symposium*, Washington D.C., 1999.