Authorization Methods for E-Commerce Applications

Rolf Oppliger
Swiss Federal Strategy Unit for Information Technology (FSUIT)
Monbijoustrasse 74, CH-3003 Berne, Switzerland
oppliger@computer.org

Abstract

In the recent past, a lot of work has been done in establishing public key infrastructures (PKIs) for electronic commerce (e-commerce) applications. Unfortunately, most of these PKIs can only be used to authenticate the participants of e-commerce applications; they can't be used to properly authorize the participants and to control access to system resources accordingly. Consequently, these PKIs address only half of the problem with regard to e-commerce applications, and some complementary technologies are required to address the authorization problem, as well. In this paper, we elaborate on such technologies and corresponding authorization methods for e-commerce applications. In particular, we address certificate-based authorization, the use of attribute and SDSI/SPKI certificates, as well as the use of databases. We conclude with the insight that there is no single best authorization method, and that different e-commerce applications may require different authorization methods.

1 Introduction

It is commonly agreed that the use of public key cryptography for encryption and digital signatures requires the existence of public key infrastructures (PKIs) [1]. In short, a PKI consists of one or several certification authorities (CAs) that issue and revoke certificates for users or other CAs. In the recent past, a lot of work has been done in establishing PKIs for electronic commerce (e-commerce) applications. Examples include government-sponsored programs to create PKIs, as well as PKIs that are established for specific applications, such as S/MIME (secure MIME) for secure electronic messaging or SET (secure electronic transaction) for secure credit card payments over the Internet. Also, the Internet Engineering Task Force (IETF) has tasked two working groups (WGs) to work in this particular field:

- On the one hand, the Public Key Infrastructure X.509 (PKIX) WG has been tasked to profile and actually build a PKI for the Internet community based on the ITU-T recommendation X.509 [2];
- On the other hand, the Simple Public Key Infrastructure (SPKI) WG has been tasked with designing and producing a certificate structure and operating procedure to meet the needs of the same community for trust management in as easy, simple, and extensible a way as possible (not necessarily based on the ITU-T recommendation X.509).

The reason that has motivated the IETF to task two WGs is due to the possibility that the task of actually building an X.509-based PKI for the Internet community might be too big. Note that the Privacy Enhanced Mail (PEM) WG failed to build an X.509-based PKI for secure electronic messaging a couple of years ago [3]. Nevertheless, it is only fair to mention that this failure was also due to product immaturity at this time, and that the situation has changed fundamentally in the meantime. As of this writing, there are several working systems in operation that effectively make use of X.509-based PKIs.

Having a closer look at the two approaches being followed by the IETF WGs mentioned above, one can easily recognize that the main difference between them is that the PKIX WG assumes the existence of a global namespace, whereas the SPKI WG does not make this assumption and starts from local namespaces, such as proposed by Ron Rivest and Butler Lampson in their Simple Distributed Security Infrastructure (SDSI). The resulting SDSI/SPKI certificates are conceptually similar to attribute certificates (at least with regard to authorization).

Unfortunately, contemporary PKIs can only be used to authenticate the participants of e-commerce applications (and to establish session keys based on the public key certificates); they can't be used to properly authorize the participants and to control access to system resources accordingly. Consequently, these PKIs address only half of the problem with regard to e-commerce applications, and some complementary technologies are required to address the authorization problem, as well. According to a position paper presented by Joan Feigenbaum at the 3rd USENIX Workshop on Electronic Commerce in 1998, a PKI that enables applications to decide who signed a request isn't immediately useful; rather, one needs an infrastructure that allows the verifier of a digitally signed

document to decide whether the signer has the authority to do what he wants to do [4]. This point is further explored in [5]. Consequently, a PKI should not be used primarily to enable authentication; rather, it should be used to enable authorization.

Following this line of argumentation, we elaborate on complementary technologies and corresponding authorization methods for e-commerce applications in this paper. In particular, we briefly address certificate-based authorization in Section 2, and discuss the use of attribute certificates, SDSI/SPKI certificates, and databases in Sections 3 to 5. Finally, we conclude with the insight that there is no single best authorization method, and that different e-commerce applications may require different authorization methods in Section 6.

2 Certificate-based Authorization

Given the current format of an X.509v3 public key certificate [2], arbitrary information can be encoded into the corresponding standard and extension fields. On the server side, this information can then be used to make more or less intelligent access control decisions. For example, consider the case that you want to make a Web server accessible only to the users that are affiliated with a specific organization or organizational unit. In this case, it is possible to set up an SSL- or TLS-enabled Web server (note that the Transport Layer Security (TLS) protocol is just a slightly enhanced version of Netscape's Secure Sockets Layer (SSL) protocol and as such it is being standardized by the Internet Engineering Task Force (IETF) working group (WG) of the same name) that is configured to require client authentication and to check the certificates provided by the clients that want to access the server accordingly (also note that the certificates are provided as part of the corresponding SSL or TLS handshake protocol messages):

- If the „O“ or „OU“ field of the certificate provided by the client included the name of the organization or organizational unit, access would be granted;
- Otherwise, if neither of the above-mentioned fields included this information, access would be denied.

Obviously, this is a very simple example and it is possible to make more intelligent access control decisions. In fact, the decisions may take into account any information that is found in a client X.509v3 public key certificate. This information is made available to the Web server and can be accessed through corresponding environment variables.

Obviously, certificate-based authorization is appropriate if the authorization-relevant information is relatively constant in time and high granularity of access control decisions is not required.

3 The Use of Attribute Certificates

More recently, the singular use of the term „certificate“ as originally proposed by Loren M. Kohnfelder in [6] has been challenged with the use and proliferation of attribute certificates in the Internet community. In a more general sense, the term „certificate“ refers to a digitally signed testimony to whom it may concern, stating some fact or granting some form of privilege. One possibility for a certificate is to bind a public key to a (globally or locally unique) name. However, this is just a possibility for a certificate to state a fact, and there are many other facts a certificate may state, as well. For example, a certificate may grant some attributes to its owner. This is actually the aim of an attribute certificate (AC). Similar to public key certificates, ACs bind characteristics of an entity, called attributes, to that entity by the digital signature of an Attribute Authority (AA) on a particular AC. Consequently, the major difference between a public key certificate and an AC is that the former includes a public key (the key that is certified), whereas the latter includes a more general attribute (the attribute that is certified). As such, the AC can be used for various purposes. For example, the AC may include group membership, role, clearance, or any other authorization or access control-related information associated with its owner. In conjunction with authentication services, ACs may provide the means to securely transport authorization information to decentralized applications. In fact, ACs are well suited to control access to system resources, and to implement role-based authorization and access controls [7]. In this case, ACs are conceptually similar to privilege attribute certificates (PACs) as used in the SESAME (Secure European System for Applications in a Multivendor Environment) project, the Open Group's Distributed Computing Environment (DCE) [8], and Microsoft Windows 2000.

In general, AC-issuing AAs are assumed to be certified by CAs, so that a single point of trust (namely a trusted public key of a root CA) can be used to validate the certificates of peer CAs, AAs, and users. Also, ACs are distributed in the same way as public key certificates. So if an organization already has a directory service or certificate repository that can be used to distribute public key certificates and certificate revocation lists (CRLs) or other certificate status information, this service may also be used to distribute the ACs. Note that - similar to public key certificates - ACs can be used in either the „push“ or „pull“ model:

- In the push model, the AC is pushed from the client to the server;
- In the pull model, the AC is pulled by the server from a network service (either the AC issuer or a directory service that is fed by the AC issuer).

An AC infrastructure should support both models, since some applications work best when a client pushes the AC to the server, whereas for other applications it is more convenient for the client simply to authenticate to the server and for the server to request the client's AC from a network service. Note that this is somehow contradictory to Proposition 2 of [9], where it is claimed that „the signer can (and should) supply all evidence the acceptor needs, including recently information.“ While this proposition holds in most situations, there are other situations that require a server to handle specific tasks on the client's behalf (e.g., thin clients, or - more generally - devices with only few computing power).

More recently, the Internet Engineering Task Force (IETF) TLS WG has started to work on AC-based authorization as a possible extension to the TLS protocol. There are several requirements which ACs are supposed to meet. Among the more important requirements are the following ones:

- AC validity periods are typically measured in hours, as opposed to months (or years) for public key certificates;
- ACs may be valid only for a set of durations (e.g., from 8am to 1pm and from 2pm to 6pm);
- Delegation mechanisms and even chains of delegation should be supported;
- ACs should support the encryption of some, or all, attributes (e.g. passwords);
- ACs should support audit and billing mechanisms in one form or another.

In addition, it is sometimes required that ACs also support anonymity in the sense that certain ACs should be usable even when they don't contain a name for their owners. In this case, the use of ACs is very closely related to the use of capabilities in operating systems and corresponding access control models. Note, however, that the use of anonymous ACs is contraversial.

According to the specifications that are currently under development, an AC may consist of the following fields:

- *Version*: This field indicates the version of the AC format in use (currently version 1);
- *Subject*: This field identifies the principal with which the attributes are being associated. Identification can be either by name or by reference to an X.509 public key certificate (such a reference comprises a combination of an X.509 issuer name and a corresponding certificate serial number);
- *Issuer*: This field identifies the AA that issued the AC;
- *Signature*: This field indicates the digital signature algorithm used to sign the AC;
- *Serial Number*: This field contains a unique serial number for the AC. The number is assigned by the issuing AA and used in a CRL to identify the AC;
- *Validity*: This field may contain a set of possibly overlapping time periods during which the AC is assumed to be valid;
- *Attributes*: This field contains information concerning the owner of the AC (the owner is the principal that is referred to in the subject field). The information may be supplied by the subject, the AA, or a third party, depending on the particular attribute type in use;
- *Issuer Unique Identifier*: This field contains an optional bit string used to make the issuing AA name unambiguous in the case that the same name was reassigned to different principals through time;
- *Extensions*: This field allows for the addition of new fields to the AC. It basically works the same way as the extensions field of an X.509 certificate.

Note that attribute certificates constitute a general-purpose mechanism that potentially has many uses, and that distribution of authorization information is just one use. Also note that the above-mentioned fields represent just a proposal for a standardized AC format and that it is possible and very likely that other (competing) formats will be proposed and submitted for standardization in the future. For example, the World Wide Web Consortium (W3C) Digital Signature (DSig) Working Group has proposed a standard format for making digitally signed, machine-redable ssertions about a particular information resource. In fact, it is the goal of the DSig project to provide a mechanism to make a statement of the follwoing form: *Signer* believes *statement* about *information resource*. Obviously, ACs also represent information resources and can be digitally signed according to the DSig syntax and semantics. Refer to the corresponding Web pages hosted at `http://www.w3.org` for further information about the DSig project. In this paper, we don't address all possible formats of ACs, but rather focus on their functionality. From this point of view, it doesn't really matter whether ACs are implemented according to the formats proposed by the ISO, IETF, or W3C.

4 The Use of SDSI/SPKI Certificates

The third authorization method for e-commerce applications involves the use of SDSI/SPKI certificates as briefly mentioned in Section 1. In short, the SDSI uses S-expressions as the standard format for certificates. In short, an S-expression is recursively defined as being either an octet-string (a finite sequence of eight-bit octets), or a finite list of simpler S-expressions.

Contrary to the philosophy of X.500 and X.509, an SDSI principal refers to its public key (and not to a string

that may be associated with the principal that holds the correspondig private key), and the main feature of the SDSI is its extensive use of local name spaces. In short, a local name space is defined relative to a particular key, which can later be dereferenced to a key or another SDSI name. A SDSI name, in turn, is a sequence of arbitrary length consisting of a public key followed by zero or more identifiers. An example of a SDSI name is (K_{Rolf} Isabelle Sister). It begins with the key K_{Rolf} that refers to my public key. The identifier Isabelle following the principal K_{Rolf} is understood to be equivalent to another SDSI name $K_{Isabelle}$ in the name space of K_{Rolf}. Subsequently, the identifier Sister is defined in the name space defined by the key that is bound to Isabelle. In this example, Sister can be dereferenced to Caroline who's actually my sister-in-law (or the sister of my wife Isabelle respectively). Obviously, this scheme can be generalized and SDSI's local name spaces can be linked to form arbitrary chains. Further information about the SDSI versions 1.0 and 2.0 can be found on the Web by following the URL `http://theory.lcs.mit.edu/~cis/sdsi.html`.

The participants of the IETF SPKI WG realized that SDSI met their requirements and merged the two approaches into a collaborative effort. As such, the IETF SPKI WG has done some work and come up with a number of Internet Drafts. For the purpose of our discussion, SDSI/SPKI certificates are conceptually similar to attribute certificates, and the two classes of certificates are not further distinguished in this paper. Note, however, that industry is pushing attribute certificates more strongly than it is supporting SDSI/SPKI certificates.

5 The Use of Databases

The fourth authorization method for e-commerce applications involves the use of databases. The architecture of a corresponding distributed certificate management system (DCMS) to support group-based access was originally proposed in [10,11]. In short, the aims of the DCMS are two-fold:

- On the one hand, the DCMS is to provide a high degree of delegation and decentralization with regard to the provision of CA (and AA) services. The underlying assumption is that delegated and decentralized services can be provided more effectively and efficiently than their centralized counterparts, and that they are also less subjected to overheated political discussions (since everybody can contribute to the provision of the service);
- On the other hand, the DCMS is also to support the notion of group membership to provide group-based access controls.

The DCMS architecture consists of three main components:

- The DCMS core;
- One or several decentralized DCMS frontends;
- The DCMS database that is a distributed database maintained by the DCMS core. Data that are collected at the DCMS frontends are periodically uploaded to the DCMS core, processed by the DCMS core, and redistributed to the DCMS frontends.

The DCMS core is the component that actually represents the CA. It holds the private key that is used to digitally sign and issue public key certificates. As such, the DCMS core is assumed to run in a phyically secure environment and operated by DCMS administrators with corresponding privileges. In general, the DCMS core is operated offline. It is put online only to communicate with the DCMS frontends for a relatively short period of time (namely for database synchronization). Contrary to the DCMS core, the DCMS frontends are provided by Web servers that are operated by corresponding system administrators. The DCMS frontends can either run as normal Web servers or SSL/TLS-enabled server-only authenticated Web servers (which is notably the preferred configuration). The same DCMS frontend can even run on a fully operable SSL/TLS-enabled Web server (including client authentication), which then allows certain users, called DCMS agents, to perform specific actions within the database. In short, a DCMS agent is a user who has been granted special privileges with regard to the verification of user identities or the confirmation of their appropriate group memberships. Other terms are used elsewhere for essentially the same function. For example, the term local registration agent (LRA) is used in ANSI X9 standards, local registration authority (also acronymed as LRA) is used in [1], organizational registration agent (ORA) is used in certain U.S. government specifications, and plain registration agent (RA) is used elsewhere. It is up to the DCMS administrators to nominate users as DCMS agents. In either case, communications between the DCMS core and its frontends must be secured with a cryptographic security protocol, such as the IPsec suite of protocols, the SSL or TLS protocols, or the Secure Shell (SSH) software [3]. Note that the DCMS topology is a star, simplifying the tasks of key management (for the cryptographically secured communication between the DCMS core and its frontends) and database synchronization considerably.

The DCMS architecture is centered around the notion of a group. In essence, a certificate may be granted group membership in which case the certificate owner obtains privileges related to this group. Note, however, that the privileges that are granted to a certificate are not encoded into corresponding data, but stored offline within the

DCMS database. The database entries are then used to link public key certificates to corresponding group memberships and privilege information. This has the advantage of having public key certificates in a relatively constant and permanent form, whereas all transient group membership information is stored and dynamically maintained in the database.

With regard to a given group A, a certificate can be in one of the following states:

- The certificate can be in the *applied for* A state. In this state, the certificate has been applied for but is not (yet) a member of group A. Eventually, membership will be granted or revoked at some later point in time. Synonymously, one can say that the certificate is in the pending state for A;
- The certificate can be in the *member of* A state. In this state, the certificate has privileges related to group A. Synonymously, one can say that membership of A is granted or issued to the certificate, or that the certificate has the issued state for A respectively;
- The certificate can be in the *revoked out of* A state. In this state, membership to group A (and related privileges) has (have) been revoked, eventually only temporarily. This state can only be reached after a certificate has had the issued or pending state for A;
- Finally, if a certificate does not belong to any of the above-mentioned states, it is in the *unknown* state for A. No explicit state is given in this case. Consequently, the certificate has no privileges related to group A (the same is true for the pending and revoked states). A certificate in unknown state is also said to be external (external from A's point of view). All other states mentioned above indicate that a certificate is internal, meaning that it has a well-defined state (either issued, revoked, or pending). An external certificate can become internal with regard to A either by application from a user, or by an explicit import operation performed by a legitimate user (who will be called an agent for group A).

As mentioned above, each group is managed by one or several users with special privileges and these users are called DCMS agents (or agents in short). A DCMS agent is a strongly authenticated (e.g., through the use of SSL/TLS client authentication) user who has the privilege on the DCMS frontends to modify the state of certificates from his groups. His groups are actually all groups he's a legitimate agent for. Obviously, a user can be an agent for several groups (all of them are called „his groups“), and a group can be run by one or several agents. Consequently, there is an n:m-relationship between groups and agents (n,m >= 1).

Per definition, DCMS administrators are agents for all groups. Also, there is a special group (let's use the dot sign to refer to this group). Granting a certificate access to the „.“ group means that the identity of the certificate requester has been verified according to a certain policy or certificate practice statement (CPS). Similar to any other group, the legitimate agents of the „.“ group are nominated by DCMS administrators. In the special case of the „.“ group, the DCMS agents are also called validators, meaning that the agents of this group are authorized to verify the identity of the corresponding certificate requesters, and to validate the certificates accordingly. Certificate validation may require some policy-driven procedure, such as a phone call to the certificate requester or having him appear in person and present some official document, such as a photo ID (the procedure is specified in the CPS). The important point to note is that there may be several validators around. Each validator is authorized to validate a certificate, no matter if it belongs to any of his groups. Actually, a validator does not even need to be a DCMS agent for any other group than „.“. Also note that each certificate can be subject to several validations. As soon as V1 validates a certificate, it gets the „V1,T1“ state (T1 being a timestamp for the first validation performed by V1). Any validated certificate can be revalidated at any time, possibly even several times. For example, if another validator V2 validates the certificate at some later point in time, the certificate gets the „{V1,T1;V2,T2}“ state (T2 being the second timestamp for the second validation performed by V2). Consequently, a list of all validations for a given certificate can be requested and, for example, used by a DCMS agent to determine whether or not to grant membership to one of his groups. Note that a validation can't be revoked by the validator. Validation revocation must always be done by the DCMS core (this is an important feature for the scheme to work).

The DCMS architecture as described in this section has been prototyped and is being used by the Swiss government to control access to intranet resources. The implementation has been called PECAN, an acronym derived from PErl Certification Authority Network [11]. A DCMS frontend is accessible from the Internet at URL `https://ca.admin.ch`.

6 Conclusions and Outlook

In the recent past, a lot of work has been done in establishing public key infrastructures (PKIs) for e-commerce applications. However, contemporary PKIs can only be used to authenticate the participants of e-commerce applications; they can't be used to properly authorize the participants and to control access to system resources accordingly. Consequently, these PKIs address only half of the problem with regard to e-commerce applications, and

some complementary technologies are required to address the authorization problem, as well.

In this paper, we have elaborated on such technologies and corresponding authorization methods for e-commerce applications. In particular, we have addressed certificate-based authorization, the use of attribute certificates, the use of SDSI/SPKI certificates, and the use of databases. Each of the four authorization methods has its own advantages and disadvantages. For example, certificate-based authorization is simple and straightforward. As such, it is well suited for applications that don't require sophisticated access control decisions. Both the use of ACs and SDSI/SPKI certificates as well as the use of databases can be used for applications that require intelligent and more sophisticated access control decisions. The more static the authorization information is, the more advantageous is the use of ACs and SDSI/SPKI certificates. Contrary to that, the use of databases is better suited for situations in which the authorization information is highly transient and changes dynamically. Consequently, we conclude with the insight that there is no single best authorization method, and that different e-commerce applications may require different authorization methods.

The question which method is best suited in a given situation primarily depends on the temporal characteristics of the corresponding authorization information:

- If the authorization information is transient and changes very dynamically (let's say all couple of minutes or hours), the use of databases is certainly appropriate;
- If, however, the authorization information changes less frequently (let's say each couple of hours or days), the use of attribute or SDSI/SPKI certificates is certainly appropriate;
- Finally, if the information is permanent and not subject to frequent changes, the use of public key certificates (with encoded authorization information) is certainly the best way to go.

Finally, it is important to note that certificate revocation (for all types of certificates) raises some new and very challenging issues. It turns out that the need to deal with certificate revocation reintroduces some online components (that security engineers have been able to reduce due to the use of public key cryptography). Technologies to address the certificate revocation problem are beyond the scope or this paper. They are further addressed in Chapter 8 of [12].

References

[1] W. Ford, and M.S. Baum, *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures & Encryption*, Prentice Hall PTR, Upper Saddle River, NJ, 1997

[2] ITU-T (former CCITT) Recommendation X.509: *The Directory - Authentication Framework*, 1988

[3] R. Oppliger, *Internet and Intranet SecuritySecurity.* Artech House, Norwood, MA, 1998

[4] J. Feigenbaum, *Towards an Infrastructure for Authorization*, Position Paper presented at the 3rd USENIX Workshop on Electronic Commerce, August 31 - September 3, 1998, Boston, Massachusetts, MA (USA)

[5] M. Blaze, J. Feigenbaum, and J. Lacy, *Decentralized Trust Management*, Proceedings of IEEE Conference on Security and Privacy, 1996, pp. 164 - 173

[6] L.M. Kohnfelder, *Towards a Practical Public-key Cryptosystem*, MIT bachelor's thesis, May 1978

[7] R. Oppliger, G. Pernul, and C. Strauss, *Using Attribute Certificates to Implement Role-based Authorization and Access Controls*, in submission

[8] R. Oppliger, *Authentication Systems for Secure Networks*, Artech House Publishers, Norwood, MA, 1996

[9] R.L. Rivest, *Can We Eliminate Certificate Revocation Lists?* MIT Laboratory for Computing Science, Cambridge, MA

[10] A. Greulich, R. Oppliger, and P. Trachsel, *Der Einsatz eines verteilten Zertifikat-Managementsystems in der Schweizerischen Bundesverwaltung*, German Informatics Society (GI) Working Conference „Verlässliche IT-Systeme“ (VIS '99), Essen (Germany), September 22 - 24, 1999

[11] A. Greulich, and R. Oppliger, *A Distributed Certificate Management System (DCMS) Supporting Group-based Access Controls*, in submission

[12] R. Oppliger, *Security Technologies for the World Wide Web*, Artech House Publishers, Norwood, MA, to appear in December 1999.