A Distributed Certificate Management System (DCMS) Supporting Group-based Access Controls

Rolf Oppliger Andreas Greulich Peter Trachsel
Swiss Federal Strategy Unit for Information Technology FSUIT
Monbijoustrasse 74, CH-3003 Berne, Switzerland
{rolf.oppliger,andreas.greulich,peter.trachsel}@isb.admin.ch

Abstract

Mainly for scalability reasons, many cryptographic security protocols make use of public key cryptography and require the existence of a corresponding public key infrastructure (PKI). A PKI, in turn, consists of one or several certification authorities (CAs) that issue and revoke certificates for users and other CAs. Contrary to its conceptual simplicity, the establishment and operational maintenance of a CA or PKI has turned out to be difficult in practice. As a viable alternative, this paper proposes an architecture for a distributed certificate management system (DCMS) that can also be used to provide support for group-based access controls. The architecture has been prototyped and is being used by the Swiss Federal Strategy Unit for Information Technology (FSUIT) to protect access to intranet resources.

1 Introduction

Network security is a hot topic today. Access control services are usually provided through the use of firewalls, whereas communication security services are usually provided through the use of cryptographic security protocols that work at the Internet, transport, and/or application layer of the TCP/IP communications protocol suite [1,2]. Mainly for scalability reasons, many of the cryptographic security protocols make use of public key cryptography and require the existence of a corresponding public key infrastructure (PKI). A PKI, in turn, consists of one or several certification authorities (CAs) that issue and revoke certificates for users or other CAs. The CAs may be organized in many ways, including, for example, a hierarchy or a decentralized web of trust (making heavy use of cross-certificates).

Since the term "certificate" was first used by Loren M. Kohnfelder to refer to a digitally signed record holding a name and a public key [3], it has been assumed that the only purpose of a certificate is to bind a public key to a globally unique name. In fact, this assumption has led to the design of several PKIs on top of existing naming schemes and directory services, such as provided by the ITU-T recommendations of the X.500 series. However, the singular use of the term "certificate" has recently been challenged with the use and proliferation of attribute certificates within the Internet community [4,5]. Unfortunately, standardization is far away from coming up with a commonly agreed and widely deployed format for attribute certificates, and the use of attribute certificates on a large scale and across applications is still not possible today (this will probably change when attribute certificates become widely deployed on the Internet).

Due to the singular use of the term "certificate" in the security community, there is still considerable confusion on how to build a PKI. For example, the Internet Engineering Task Force (IETF) has tasked a Public Key Infrastructure X.509 (PKIX) Working Group (WG) to design and build a PKI for the Internet community based on the ITU-T recommendation X.509 [6], whereas the Simple PKI (SPKI) WG has been tasked with 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. The fear that motivated the IETF to task two WGs is actually due to the possibility that the task of building an X.509-based PKI for the Internet community is too big. Note that the Privacy Enhanced Mail (PEM) WG failed to build and establish an X.509-based PKI for secure electronic mail for the Internet community a couple of years ago [7]. Having a closer look at the two approaches being followed by the WGs, one actually recognizes that the main difference between them is the fact that the IETF PKIX WG assumes the existence of a global namespace, whereas the IETF SPKI WG does not make this assumption and starts from linked local namespaces, such as the ones proposed by Ron Rivest and Butler Lampson in their Simple Distributed Security Infrastructure (SDSI). Furthermore, the SDSI/SPKI effort addresses authorization in addition to authentication (since a PKI that only addresses authentication isn't very useful for e-commerce applications). The results of the IETF PKIX and SPKI WGs and are overviewed and discussed in Chapter 8 of [8]. They are not further addressed in this paper.

One may reasonably dispute whether an X.509-based PKI is actually the infrastructure required for the Internet community and the e-commerce applications that are supposed to take place within the Internet. Anyway, from a

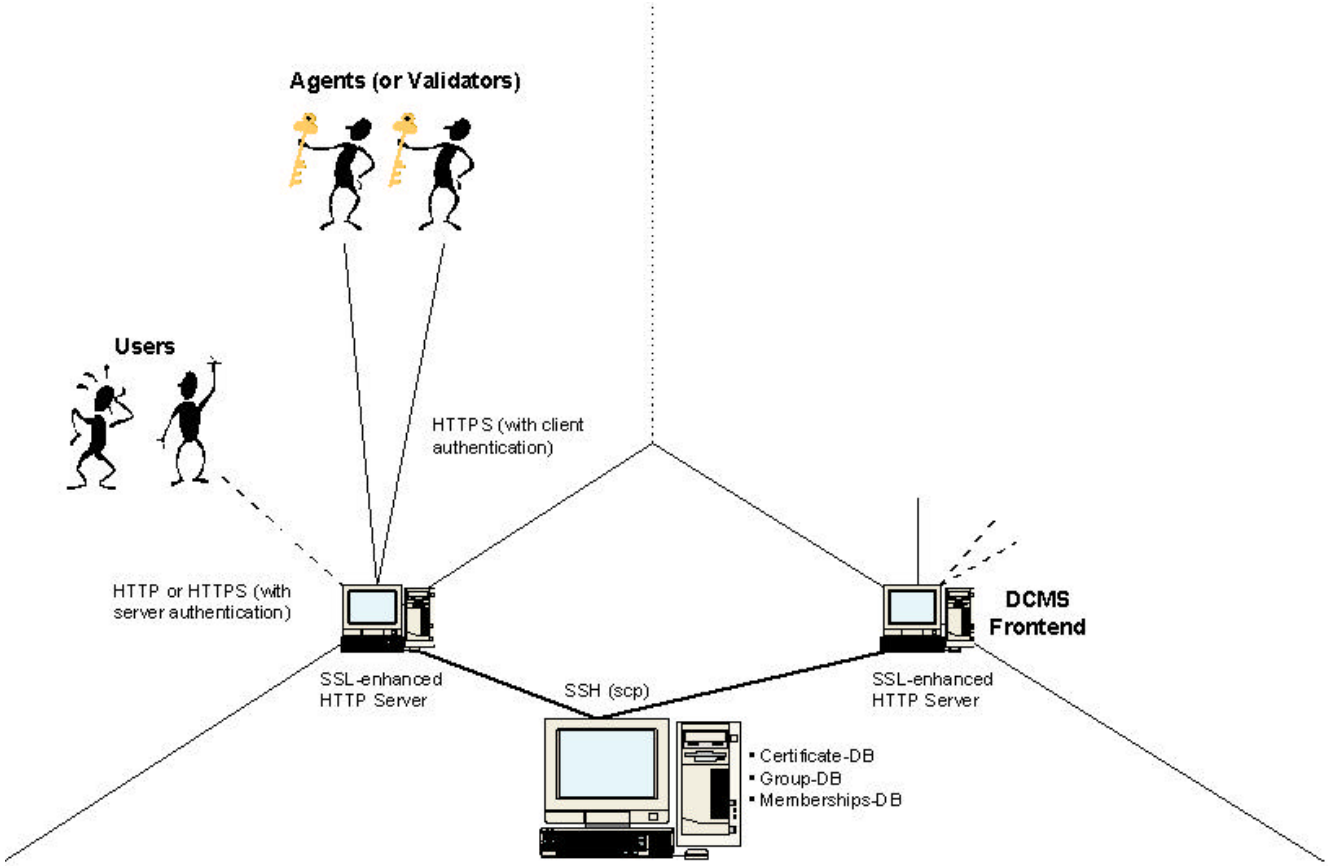

Figure 1: The architecture of the distributed certificate management system (DCMS)

practical (and corporate) point of view, the situation is fundamentally different and simpler. Within a corporate environment, there typically exists a namespace in which each employee is not only identified with a unique name, but is also assigned a unique employee number. Consequently, there exists a namespace (within the corporate environment) that can be used to have certificates bind public keys to unique names. Consequently, ITU-T X.509 provides a useful ground for building a PKI for a corporate environment. In fact, many organizations follow this approach and establish X.509-based PKIs to secure their intranet and extranet connections.

Assuming the existence of an internal (locally unique) namespace within a corporate environment (which can either be a company or an organization), this paper proposes an architecture for a distributed certificate management system (DCMS). In addition to manage X.509-based public key certificates, the DCMS can also be used to provide support for group-based access controls. As such, the DCMS addresses authentication and authorization from an architectural point of view. In short, the aims of the DCMS are two-fold:

- On the one hand, the DCMS architecture is to provide a high degree of delegation and decentralization with regard to the provision of its certification services. The underlying assumption is that a delegated and decentralized certification service can be provided more effectively and efficiently than it centralized counterpart (since existing organizational units, such as the human resources department, can be used).
- On the other hand, the DCMS architecture is to introduce the notion of a group membership to provide support for group-based access controls. This is similar to the use of attribute certificates to facilitate access controls [4,5].

The DCMS architecture has originally been described in [9]. This paper goes some steps further in refining the architecture, elaborating on an existing prototype implementation called PECAN (an acronym derived from "PErl Certification Authority Network"), and pointing out some areas for further study.

The rest of this paper is organized as follows: The DCMS architecture is introduced and overviewed in Section 2. The corresponding prototype implementation is described in Section 3, and areas of further study are addressed in Section 4. Finally, conclusions are drawn in Section 5. The paper refers to ongoing work within the Swiss Federal Strategy Unit for Information Technology (FSUIT). Consequently, future papers will address the experiences

that are made with the use and deployment of PECAN within the intranet of the Swiss governmental bodies.

2 DCMS Architecture

The DCMS architecture is illustrated in Figure 1. It basically consists of three components:

- A DCMS core;
- One or several decentralized DCMS frontends;
- A DCMS database that is maintained by the DCMS core. Data that are collected at the DCMS frontends are periodically synchronized, processed by the DCMS core, and replicated and redistributed to the frontends.

The DCMS core is a standalone application that is operated by DCMS administrators with corresponding privileges and access rights. In fact, DCMS administration requires some sort of shell access to the underlying operating system (e.g., UNIX or Windows NT). Contrary to the DCMS core, the DCMS frontends are provided by "normal" HTTP or Web servers that are operated by the corresponding system administrators. The DCMS frontends can run either as normal Web servers or SSL-enabled server-only authenticated Web servers (which is actually the preferred configuration). The same DCMS frontend can also run on a fully operable SSL-enabled Web server (including, for example, client authentication), which then allows certain users, called DCMS agents, to perform specific actions within the database. In essence, a DCMS agent is a user who has been granted special privileges with regard to the verification of user identities or the confirmation of corresponding group memberships. It is up to the DCMS administrators to nominate users as DCMS agents. In either case, communications between the DCMS core and the corresponding frontends must be secured with a cryptographic security protocol, such as the Internet Security Protocol (IPSP), the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol, or the Secure Shell (SSH) [1]. 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 (to support group-based access controls). In essence, a group is an attribute granted to a certificate to provide its owner with some specific privileges. Unlike other proposals, such as attribute certificates, the privileges that are granted to the certificate owners are not directly encoded into the corresponding certificate data structures, but are stored off-line within a database. The database entries are then used to link certificates to corresponding group membership information. This has the advantage of having certificates in a permanent form, whereas all transient group membership information are stored and maintained dynamically in the database (where it can be managed more easily). The list of available groups is determined by the DCMS administrators and can be changed at will.

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

- The certificate can be in the "member of A" state. In this state, the certificate has access 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.
- The certificate can be in the "revoked out of A" state. In this state, membership to group A (and its related privileges) has been revoked.
- 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. Syononymously, one can say that the certificate is in the "pending" state for A.
- Finally, if a certificate does not belong to any of the states mentioned above, it is in the "unknown" state for this particular group A. In this case, no explicit state has been given yet. Consequently, the certificate has no access 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 group A's perspective). All other states mentioned above indicate that a certificate is "internal", meaning that it has a well-defined state. 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 agent of the group.

As mentioned above, each group is managed by one or several users who have special privileges (the so-called DCMS agents). In short, a DCMS agent is a strongly authenticated (e.g., through the use of SSL client authentication) user who has the privilege on the DCMS frontends to modify the states of the certificates for "his groups." These are all the 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 more agents. Consequently, there exists 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 called ".". Granting a certificate access to the "." group actually means that the identity of the certificate requester has been verified according to a specific policy or certification practice statement (CPS). Similar to any other group, the legitimate agents of the "." group are nominated by the DCMS administrators. In the case of the "." group, 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. Validation of a certificate implies some policy-driven procedure, such as placing a phone call to the appli-

cant or appearing in person and showing a photo ID. The important point to note is that there may be several validators around. Each validator is authorized to validate any certificate, no matter if it belongs to one of his groups. Actually, a validator does not even need to be an agent of any other group. Also note that each certificate can be subject to several validations. As soon as a validator V1 validates an anonymous certificate, it gets the "V1T1" state (T1 referring to a timestamp for the validation process). But any validated certificate can be revalidated at any time, possibly several times. So, if another validator V2 revalidates the certificate at some later point in time, the validation state gets the "{V1T1,V2T2}" state (T2 referring to the timestamp for the new validation). Consequently, a list of all validators that validated a certificate can be requested and, for example, used by an agent in order to determine whether or not to grant membership to one of his groups. Note, however, that the monotony property of the validation process (meaning that a validation can't be revoked at some later point in time) is important for the scheme to work. In spite of the fact that certificates must be able to be revoked, there is no need to revoke validations (validation revocation doesn't make sense from a logical point of view).

3 Prototype Implementation

The DCMS architecture described so far has been prototyped and is being used at the Swiss Federal Strategy Unit for Information Technology (FSUIT) in a software called PECAN (an acronym derived from "PErl Certification Authority Network"). The name of the software has been chosen due to the fact that it's main parts are implemented with the Perl scripting language.

In the subsections that follow, we address the content of the databases that are used by PECAN, the database synchronization process, access control lists (ACLs) extraction, and the corresponding Perl scripts.

3.1 Databases

PECAN uses a Perl-based database management system (DBMS) called Sprite, but any other SQL-based DBMS could be used instead. The PECAN software currently comprises three main databases (a fourth database that includes the rules that specify how to use the certificates to control access to intranet resources is not addressed in this paper):

- The `Certificate-DB` stores all certificate information, including the original request (e.g., in Netscape's SPKAC format), fields filled in by the user, state of the internal certificate (pending, issued, or revoked), timestamps, and the certificate itself as soon as it is issued. Also, each new certificate gets a unique identifier (ID), called the CERTID, which is a letter followed by a 6-digit number (the letter is unique for each DCMS or PECAN frontend). Consequently, the format of a `Certificate-DB` entry is as follows:

CERTID	Unique certificate ID
TEL	Telephone number of the requester
CN	Common name (CN) of format "surname, firstname :SEQ=\d:" (SEQ referring to a sequence number)
CC	Country code
ST	State/province information
OO	Organization
OU	Organization unit (suborganization)
EMAIL	Electronic mail address
TIMECREAT	Timestamp of certificate request
TIMEMOD	Timestamp of last modification
STATUS	Certificate status ($PENDING, $ISSUED, or $REVOKED)
REQ	Original certificate request
REQFORMAT	Format of REQ (e.g., SPKAC or PKCS7)
CERT	Certificate (as soon as it is issued)
CERTFORMAT	Format (e.g., X509)
EXPIRES	Expiration date

  Note that the certificate's common name (CN) is usually sent by the browser and constructed by a JavaScript code segment. It contains the name entered by the user, and a sequence number given by the user (usually set to 1). Sequence numbers allow reissuing certificates, for example, for certificate renewal or second browsers. Users with the same CN except its sequence number are considered to be the same, whereas certificates with different sequence numbers are considered to be different. The part of the CN field up to the first ":" is called the "owner of the certificate" or "name of the owner of the certificate", whereas the full CN is called the "name of the certificate". Consequently, certificates with CN "Greulich, Andreas :SEQ=1:" and "Greulich, Andreas :SEQ=2:" have the same owner, but are still considered different certificates (the first and second certificate). In either case, the name of the owner is "Greulich, Andreas."

- The `Group-DB` stores group-related information. More precisely, each entry in the `Group-DB` contains a unique group ID for the group, the name of an agent, and eventually a group description (only one entry may contain a group description). As such, the `Group-DB` also stores validators and DCMS administrators; validators being agents of the group ".", and DCMS administrators being agents of the group "CA". The entries of the Group-DB can only be modified by DCMS administrators by direct writing to the core `Group-DB`. The format of a `Group-DB` entry is as follows:

GROUPID	Unique group ID
OWNER	Name of an agent for the group
DESC	Description of the group

Note that several entries with the same GROUPID may exist, but only one should have a non-empty DESC field. The OWNER field contains the name of the owner of authorized certificates, so entering "Greulich, Andreas" as owner labels all certificates of the form "Greulich, Andreas :....:" as being agents for this particular group (validators for the "." group and administrators for the "CA" group). In this case, the colon is just a separator that is discarded (simply to avoid having to modify the database when certificates are renewed). Finally, note that each GROUPID/OWNER pair is unique and must appear only once in the entire database.

- The `Memberships-DB` links the other two databases (the `Certificate-DB` and `Group-DB`). As such, each entry is keyed with a CERTID and a GROUPID, and provides state information for the certificate (identified with CERTID) with regard to the corresponding group (identified with GROUPID). A certificate can either be internal or external, in the former case being either in the pending, issued or revoked state. There might be only one entry for a given CERTID and GROUPID, except when the GROUPID is ".", as there may be several validators for a certificate. The `Memberships-DB` is readable by anybody, and anybody may also append to it because each new certificate request automatically generates pending memberships for all groups a user selected, plus one "pending" membership for group "." indicating the certificate is not yet validated. Only SSL-authenticated agents, validators and DCMS administrators may modify entries belonging to "their" (and only their) groups. The format of an entry in the `Memberships-DB` is as follows

CERTID	Link to the `Certificates-DB`
GROUPID	Link to the `Group-DB`
STATUS	Certificate status
BY	Name of the person who created or modified the database entry
TIMEMOD	Time of the creation or modification

When a user requests a certificate from a DCMS or PECAN frontend, the following steps are usually performed:

1. A `Certificate-DB` entry is created with a unique CERTID, containing all fields, except CERT and EXPIRED. The value of the STATUS field is set to $PENDING (meaning that the certificate is in pending state).
2. A `Memberships-DB` entry is created with a GROUPID value set to "." and a STATUS value set to $PENDING (meaning that the certificate requester is waiting to be validated).
3. For each group the certificate requester selected, a corresponding entry with status value set to $PENDING is created in the `Memberships-DB` (meaning that the certificate requester is waiting to be granted membership to specific groups). Note that steps 2 and 3 are not fundamentally different, since "." can also be seen as a special group.

A special case occurs if the connection of the certificate request was mutually authenticated between the user and a PECAN frontend (through the use of SSL with client authentication). In this case, the user has already been issued a certificate at some earlier point in time. All validations and granted group memberships that have been granted to the non-revoked certificates of the requester are automatically inherited. For example, if the owner "Greulich, Andreas" of the certificate "Greulich, Andreas :SEQ=2:" with CERTID "A111111" requests a new certificate "Greulich, Andreas :SEQ=3:" (with CERTID "A222222") using an SSL-authenticated access, demanding membership to groups A and B, while "Greulich, Andreas :SEQ=2:" is already a member of A (but external to B), the generated certificate automatically inherits its $ISSUED state for group A. The fields of the corresponding entry in the `Memberships-DB` are initialized to the following values:

CERTID	A222222
GROUPID	A
STATUS	$ISSUED
BY	INHERIT-A111111
TIMEMOD	...

If more than one non-revoked certificate with a matching owner name exist (such as "Greulich, Andreas :SEQ=1:" with CERTID "A100000"), multiple inheritance may happen ($ISSUED memberships are inherited, whereas $REVOKED memberships are ignored). So, if "Greulich, Andreas :SEQ=1" is issued, but "Greulich, Andreas :SEQ=2" revoked, the issued state is inherited. For obvious reasons, revoked certificates must not be considered for inheritance. In the example given above, "BY" would be set to "INHERIT-A111111-A100000."

The inheritance mechanism should make certificate renewal easier, as group memberships need not be regranted by the corresponding agents. Note, however, that this subject is still under investigation, and that security inconsistencies can't be excluded at this point in time.

3.2 Database Synchronization

From time to time, the `Certificate-DB` and `Memberships-DB` must be synchronized among the various frontends and the core (note that the `Group-DB` is stable and must not be synchronized). The synchronization process can happen on a regular basis (for example, twice a day) or on an administrator's request. In the PECAN implementation, database synchronization is performed using the SSH `scp` tool to securely copy the frontend `Certificate-` and `Memberships-DBs` to the core, where they are synchronized as follows:

1. The new entries in the frontend `Certificate-DBs` are added to the corresponding core `Certificate-DB`.
2. The entries in the frontend `Memberships-DBs` for a particular CERTID/GROUPID pair that appear in more than one DB (core plus all frontend DBs) are sorted chronologically, and the most recent one is chosen. For each entry it is then checked if the BY field matches an agent in the appropriate `Group-DB`. BY fields starting with INHERIT and AUTO are just taken for granted ("AUTO" is only added to allow importing or therwise issued certificates).
3. The log files from all frontend DBs are collected into a central logfile. Any intrusion detection algorithm would be applied to this central logfile (intrusion detection algorithms have not been implemented so far).

After the synchronization process, the new core DBs are replicated and redistributed to the frontends. In addition, a certificate-signing procedure can be started at this point in time. The corresponding script operates on the core DB only and performs the following actions: All certificates in issued state (meaning that their STATUS field is set to $ISSUED) that are validated (i.e., they have at least one "." membership granted) and have all their memberships in issued state are considered as candidates. Their requests plus a list of all candidates is extracted and shown for acknowledgment to an administrator. If the administrator gives his acknowledgment, all certificate candidates are signed in one step. Obviously, this interactive step must be skipped if the certificate-signing procedure is run in batch mode. In either case, the resulting certificates are put into the appropriate fields in the `Certificate-DB`.

3.3 Access Control Lists Extraction

To support group-based access controls, PECAN can also be used to extract access control lists (ACLs) from its databases (using a Perl script called `acls.pl`). More specifically, for each group A, a file `A.acl` can be extracted that contains the CN fields of all issued certificates with granted membership to this group (multiple certificates that belong to the same user will appear several times). For example, a file `A.acl` may look as follows (specifying two users):

```
Greulich, Andreas :SEQ=1:
Oppliger, Rolf :SEQ=3:
Oppliger, Rolf :SEQ=4:
```

Such a file is suitable for Stronghold's `SSL_Require` directive. Something like the following script may work in Stronghold's (and other Web server's) HTTPD configuration file (`httpd.conf`):

```
<VirtualHost *:1943>
SSLFlag on
SSLVerifyClient 2
<Location>
SSL_Group A
    "cn INFILE /opt/WWW/ACLs/A.acl"
SSL_Require 'A'
ErrorDocument 403 "Your certificate is
      not member of the A group. To
      join it, please contact Mr. X."
</Location>
ProxyPass / http://A.bfi.admin.ch/
...
</VirtualHost>
```

Furthermore, a certificate revocation list (CRL) can be extracted and used in a similar directive.

3.4 PERL Scripts

As mentioned above, PECAN is currently implemented as a set of several Perl scripts:

- The script `CA.cgi` implements the PECAN frontend (the user interface);
- The script `sync.pl` implements the synchronization procedure mentioned above;
- The script `sign.pl` implements the signing procedure mentioned above;
- The script `acls.pl` implements the ACL extraction process mentioned above;
- The script `index.pl` is used for backward compatibility. It reads in SSLeay-style `index.txt` for "old" certificates, automatically validates them, and adapts memberships specified as attributes in the CN field (meaning that "Greulich, Andreas :SEQ=1:A:B:" automatically gets memberships to A and B). BY-fields are put to "AUTO";
- The script `importOldCerts.pl` scans trough directories containing ".pem" files and adding their certificates into the core DB, if latter contains issued certificates without certificate data whose CN fields match. This script is usually used together with `index.pl` to include old certificates to the core DB.

4 Areas of Further Study

Obviously, the most important component of the DCMS architecture is the signing key that is required to issue digitally signed public key certificates. In one way or another, the DCMS administrator must have access to this key, and this access must be controlled accordingly. The key is either stored in the DCMS core and protected accordingly (e.g., encrypted with a key derived from a password or pass phrase that is known only to DCMS administrators), or it is provided by a DCMS administrator for temporary use (e.g., using smartcard technologies). In practice, the first option is preferred if the process of issuing digital certificate must be automated and included into a batch processing file. In general, however, it is less secure than the second option. In this case, it is up to the DCMS administrator to secure the key, and to provide it to the DCMS core whenever needed.

With regard to protecting the private signing key, one could think of using secret sharing or group signature schemes:

- In a secret sharing scheme, a secret (such as a private signing key) can be shared among many users (or DCMS administrators) in such a way that any qualified group of users can reconstruct the secret, but any unqualified group of users have absolutely no information about the secret (in the case of a perfect secret sharing scheme). This concept was first proposed in [10,11], and has been subject to a lot of cryptographic research and development meanwhile.
- In a group signature scheme, a digital signature can only be computed if a qualified group of users and holders of corresponding secrets) cooperate. More precisely, each user (or DCMS administrator) holds a signing key that allows him to compute a partial signature for a certificate. All partial signatures together constitute the signature for the certificate [12].

Note that using secret sharing or group signature schemes is equally useful in centralized and decentralized (and distributed) certificate management systems. It is assumed that the problem of protecting private signing keys for generating certificates will become more important in the future.

As of this writing, the database synchronization scheme used in the prototype implementation is fairly simple. In fact, database synchronization must be initiated by a DCMS administrator (e.g., twice a day). In large domains, it may be required that databases are resynchronized periodically. Ideally, database resychrnonization is made a permanent task that is performed in the background. Any database replication and synchronization technique can be used. Note, however, that the star topology of the DCMS should be taken into account when optimizing the corresponding database replication and synchronization schemes. Due to the star topology, the DCMS frontends need not be synchronized among each other.

In this paper, the DCMS architecture has been described for a corporate environment (for example, an intranet setting). In practice, the situation is much more complicate with groups including members of several companies and organizations. Consequently, issues related to cross-organization and inter-domain DCMS will be important in the future. For example, how can DCMS administrators cooperate across organizational borders, and how can agents do the same? These questions are left for further study. Similarly, the question of how to efficiently revoke public key certificates is addressed in [8] and not further elaborated in this paper.

Finally, a field that is left for further study is related to the granularity of group membership. Note that in this paper, group membership has always implied that a user (who has been granted group membership) has obtained privileges with regard to this group. Consequently, group membership is a boolean value (either group membership is granted or not). This granularity may not be sufficient for real-world access controls, where a user can be a member of group and have several roles within this group (e.g., leader, moderator, secretary, ...). Against this background, one can reasonably assume that applying role-based access control models to the DCMS architecture will lead to new insights that can be used to further refine the DCMS architecture and its application for access controls. This approach may actually lead to a group- and role based access control model [13].

5 Conclusions

To overcome the problems related to build and operate a scalable public key infrastructure (PKI), this paper proposed an architecture for a distributed certificate management system (DCMS) that can also be used for group-based access controls. In short, the DCMS consists of three components: a DCMS core, one or several DCMS frontends, and several databases that are periodically synchronized with a core database. The DCMS core is operated by administrators with corresponding privileges, whereas the DCMS frontends are operated by agents that have the privileges to either verify the identity of requesting users, or to confirm their memberships to specific groups. Consequently, the operational task of running a CA or PKI is distributed among severalpeople, each of them responsible only for specific tasks.

Meanwhile, the DCMS architecture has also been prototyped and is being used by the IT Security Group of the Swiss Federal Strategy Unit for Information Technology (FSUIT). The corresponding prototype implementation is called PECAN (PErl Certification Authority Network). It consists of a database system and several Perl scripts. Experience has shown that a distributed approach to managing certificates in a corporate environments offers many advantages with regard to the scalability of the resulting solution.

References

[1] R. Oppliger, *Internet and Intranet Security*. Artech House Publishers, Norwood, MA, 1998

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

[3] L.M. Kohnfelder, *Towards a Practical Public-key Cryptosystem*, MIT S.B. Thesis, May 1978

[4] R. Oppliger, G. Pernul, and C. Strauss. Using Attribute Certificates to Implement Role-based Authorization, submitted for publication

[5] R. Oppliger, Authorization Methods for E-Commerce Applications. Proceedings of the International Workshop on Electronic Commerce held in conjunction with the 18th IEEE International Symposium on Reliable Distributed Systems (SRDS '99), Lausanne (Switzerland), October 19 - 22, 1999

[6] ITU-T Recommendation X.509: *The Directory - Authentication Framework*, 1988

[7] S.T. Kent, Internet Privacy Enhanced Mail, *Communications of the ACM*, Vol. 36, No. 8, August 1993, pp. 48-60

[8] R. Oppliger, *Security Technologies for the World Wide Web*. Artech House Publishers, Norwood, MA, 1999

[9] R. Oppliger, A. Greulich, and P. Trachsel. Der Einsatz eines verteilten Zertifikat-Managementsystems in der Schweizerischen Bundesverwaltung. Proceedings of the German Informatics Society (GI) Working Conference "Verlaessliche IT-Systeme" (VIS '99), Essen (Germany), September 22 - 24, 1999

[10] A. Shamir, How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, November 1979, pp. 612 - 613

[11] G.R. Blakley, Safeguarding cryptographic keys, Proceedings of AFIPS 1979 National Computer Conference, pp. 313 - 317

[12] C. Boyd, Some Applications of Multiple Key Ciphers, Proceedings of Eurocrypt '88, pp. 455 - 467

[13] R.S. Sandhu and E.J. Coyne, Role-Based Access Control Models, *IEEE Computer Magazine*, February 1996, pp. 38 - 47