24

Creating Security Applications Based on The Global Certificate Management System

Nada Kapidzic

Department of Computer and Systems Sciences
Stockholm University & Royal Institute of Technology
Electrum 230, 164 40 Kista, Sweden
E-mail: nadak@dsv.su.se

Abstract: *The Certificate Management System (CMS) is a global network system whose primary services are generation, distribution, storage and verification of certificates. It supports various security applications which use public key cryptography by providing the means for the administration of certificates. This paper describes the interface between security enhanced application and the CMS. The interface layer with which the security applications are extended is named CMS Client. The CMS Client comprises a set of functions that enable the certification of users of security applications, retrieval of certificates and verification of retrieved certificates. The paper gives a short description of the CMS system and a detailed description of the CMS Client functions. Furthermore it describes how to extend the existing or create new security applications based on the CMS system.*

1. INTRODUCTION

The contemporary world of business is becoming highly dependant on the global computer communication networks. Therefore efficient tools must be provided to ensure the integrity and security of the data that are exchanged through those networks. Such tools are already available and the most effective ones are based on public key cryptography. The use of public key cryptography provides originator authenticity and receiver authenticity and supports data integrity and data security services. In order to efficiently implement a public key cryptography system in a global network, an infrastructure of X.509 Certification Authorities (CAs) has to be established. The CAs, as described in X.509 specification [1], are responsible for certification of users of security applications, i.e. for signing user certificates, thus vouching for the users identity. Furthermore the certificates have to be widely available to security applications in order to support the validation of the signed documents that are exchanged through the network. A system which administers certificates, i.e. provides for their generation, distribution, storage and verification is in this paper named Certificate Management System (CMS). It provides support to all security applications that need global secure interconnection. The structure of the CMS, its establishment, as well as functions and protocols between its components are described in detail by Kapidzic and Davidson [3].

J. H. P. Eloff et al. (eds.), *Information Security — the Next Decade*

This paper focuses on the interface between the security applications and the CMS system itself. It presents solutions for creating new security applications based on a global CMS or extending existing ones with the services of the world-wide certificate hierarchy. The interface between security application and the CMS system, named CMS Client, is introduced. It provides security applications with functions for authentication and certification of users, verification of certificates and retrieval of the needed certificates. The structure of the CMS system and its Clients is shown on figure 1.

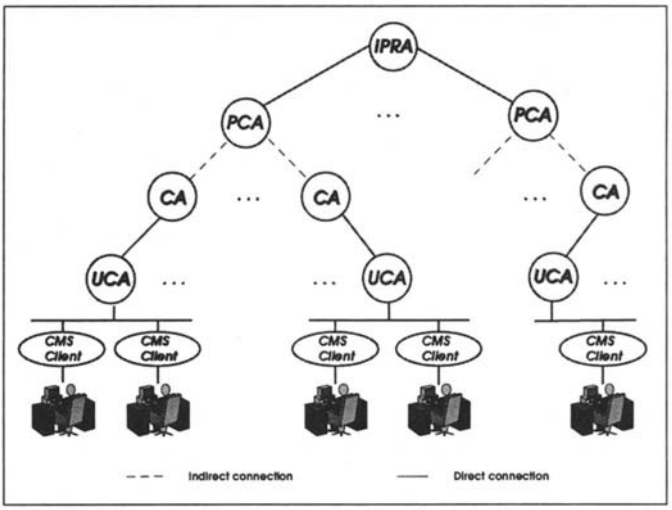

Figure 1 - The CMS System and its Clients

2. THE CERTIFICATE MANAGEMENT SYSTEM

The CMS described in this paper is a global system which performs all functions needed for efficient certificate administration. Its design is based on the X.509 framework and PEM RFC 1422 "Certificate Based Key Management" [5]. Furthermore it is extended with an extra functionality for storage and distribution of certificates. The CMS system performs those functions although they are, according to X.509, the responsibility of the X.500 System of Distributed Directories [2]. The CMS implements them for reasons of functional completeness and current unavailability of the X.500 System.

The CMS consists of a number of CAs which form a single root certification hierarchy (see figure 1). Every CA in the hierarchy is certified by only one parent CA, and there is only one root CA at the top. The roles and responsibilities of CAs depend on their position in the hierarchy. Following RFC 1422, the root CA is named IPRA and it certifies Policy Certification Authorities (PCAs). Every PCA represents the root of the certification sub-hierarchy and specifies the certification policy which is followed in its sub-hierarchy. The PCA

certifies CAs which, on their behalf, certify other CAs following the same security policy, down to the lowest level CAs. The lowest level CAs can only certify users and are therefore in this paper named User Certification Authorities (UCAs). Furthermore, UCAs are assigned additional responsibility for storage and distribution of user certificates, thus making the CMS system functionally complete and immediately operable. The CMS Clients use services of the UCAs and PCAs.

CMS administrative functions are divided into four groups: *Establishment*, *Certificate Retrieval*, *Certificate Update* and *Certificate Revocation*. Each group of functions is based on a number of special CMS messages which are exchanged between the agents of the system, i.e. CAs and CMS Clients. Every message is the result of some CMS action at the originator side and causes the CMS action at the receiver side.

Establishment involves registration and certification of CAs and users of the CMS system. The registration is an off line process which is done before the certification. A PCA registers the Distinguished Names (DNs) [2] of all CAs in its sub-hierarchy, together with information of CAs' position in the hierarchy. Furthermore, every UCA registers DNs of users which are in its certification domain. Every CA or user knows its parent in the CMS system, and every CA knows its children. Certification is based on the CMS messages: *Certificate Signature Request* and *Certificate Signature Reply*. It involves communication between a CA or a user being certified (through his/her CMS Client) and its parent CA. The entity being certified generates its certificate and sends it in *Certificate Signature Request* to its parent CA for signature. The parent CA verifies the identity of the requester, signs the certificate and sends it back in the *Certificate Signature Reply* message.

Certificate Retrieval enables automatic retrieval of user certificates. In the CMS system it is the responsibility of UCAs to store certificates of users that they certify. The CMS Client requests certificates from UCAs on behalf of the user. The functions in this group are based on the exchange of four CMS messages: *Certificate Request, Certificate Reply, Resolve Certificate Request* and *Perform Certificate Request.* Depending on the availability of the address of the owner of the required certificate, the CMS Client sends either a *Certificate Request* message to the UCA or delegates the request to the PCA by sending it a *Perform Certificate Request* message. As a result the CMS Client automatically receives a *Certificate Reply* with the required certificate. The third possibility is to send the *Certificate Request* message directly to the owner of the needed certificate. In that case the reply is not automatic, but depends on owner's action.

Certificate Update is a process which takes place when one CA in the hierarchy changes its pair of keys due to the risk of a current secret key being compromised. If one CA changes its keys, the whole certified sub-hierarchy under it is affected. Certificates of first level subordinators must be signed and forwarded to them. Furthermore the rest of the CMS agents in the sub-hierarchy must receive the new certificates. This involves another two CMS messages: *Certificate Re-Sign* and *Certificate Path Update.*

Certificate Revocation is a group of functions which handle revocation of certificates, storage of Certificate Revocation Lists (CRLs), and retrieval of CRL lists. Every CA periodically issues a list of certificates that it has revoked (i.e. invalidated). The PCA serves as a depository for CRLs issued by all CAs in its hierarchy. CMS Clients request CRL lists from the PCA when they are needed in the process of verification of certificates. All these actions are performed by use of four CMS messages: *CRL Storage, CRL Storage Confirm, CRL Request* and *CRL Reply.*

3. FUNCTIONS OF THE CMS CLIENT

The CMS Client is a part of the CMS system which issues requests for services of the CMS system on behalf of security applications (i.e. users). CMS Client functions can be roughly divided into the following four groups: *Authentication, Key Generation and Certification, Certificate Verification* and *Certificate Retrieval.* These functions do not completely correspond to the structure of functions of the CMS system itself, since the CMS system needs additional functions for its operation. Nevertheless, there is a certain correspondence between the functions of the CMS Client and the CMS functions. *Certificate Retrieval* is a client function of CMS *Certificate Retrieval* group of functions. *Key Generation and Certification* corresponds to the certification part of *Establishment* group of CMS functions. *Authentication* depends on both registration and certification part of *Establishment* phase. *Certificate Verification* depends on both *Retrieval of Certificates* and *Revocation of Certificates.* Furthermore the CMS Client functions are very interdependent. For example *Authentication* highly depends on previous *Key Generation and Certification* and successful *Verification* of user's signed *Certificate.* The following sections give a detailed description of the CMS Client functions.

3.1. Authentication

Every user which starts using a security application based on the CMS system must first be authenticated by the CMS Client. During the authentication the user is assigned a CMS status which can have one of the following values: NON-REGISTERED, REGISTERED, NON-CERTIFIED or CERTIFIED. Depending on the values of the assigned status, the user is allowed or not allowed to perform functions of the CMS Client and security application which it supports.

The status NON-REGISTERED is assigned if the user has not been registered by its UCA. The status REGISTERED is assigned if the user has been registered by the UCA, but has not yet generated his/her pair of keys and certificate. If the user has both been registered and has generated his/her pair of keys he/she is assigned status NON-CERTIFIED. If, on top of that, the user has his/her certificate signed by the UCA and has received it and verified successfully, then status CERTIFIED is assigned.

Users who have status NON-REGISTERED have the lowest allowed scope of activities, whereas users with status CERTIFIED have the highest one. NON-REGISTERED status does not allow usage of any security or CMS function. With the status REGISTERED users can perform *Key Generation and Certification.* (As a consequence of successful key generation the status is automatically changed to NON-CERTIFIED.) With status NON-CERTIFIED users can perform all CMS Client functions, but still none of the security application functions. Finally, when the user certificate has been successfully verified by *Certificate Verification*, he/she is assigned status CERTIFIED, which allows the use of all functions both of the CMS Client and of the security application.

3.2. Key Generation and Certification

This CMS Client function performs the generation of RSA keys and creation of a certificate. The generated secret key is stored following the PCA's security policy on a disk, a diskette or a smart-card. The public key is transformed into a self-signed certificate and sent to the user's UCA for signature, using the CMS message: *Certificate Signature Request* (as

shown in figure 2). If this function is completed successfully user's CMS status is set to NON-CERTIFIED.

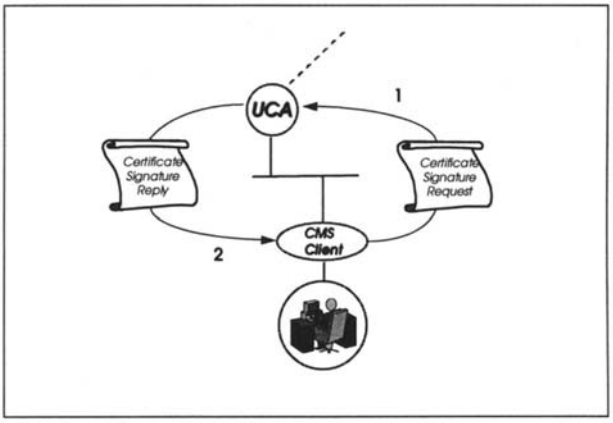

Figure 2 - Certification of a User

The UCA, upon receipt of *Certificate Signature Request* message, processes it according to the PCA's security policy following the procedure described in RFC 1422. It verifies the identity of the requester and the integrity of the received certificate. In case of successful verification it signs the certificate with its secret key and returns back the *Certificate Signature Reply*. The return message contains the user's signed certificate, the UCA's certificate and all certificates in the certification path up to the top of the CMS hierarchy. Having full certification path enables easy verification of the retrieved certificate by the CMS Client on the user's side.

The *Key Generation and Certification* function can be activated not only when the user initially starts using a security applications based on the CMS system, but also when the secret key is suspected to have been compromised.

3.3. Certificate Retrieval

Security applications need certificates in order to perform receiver authenticity and verification of signed documents. Therefore the CMS Client must provide retrieval of certificates from the CMS system. The *Certificate Retrieval* function of the CMS Client sends a request message with the identification of the requested certificate. The destination of the request depends on the availability of certificate owner's address and it can be sent to the UCA, the PCA or directly to the owner.

If the address of the owner of needed certificate is available, then it is the choice of the user of the security application whether the request will be sent to the UCA or directly to the owner. In both cases a *Certificate Request* message is sent. The message contains the identity of the owner of the needed certificate. The reply will be received in the form of *Certificate*

Reply message which contains the needed certificate and all certificates in its certification path up to the top of the hierarchy. If the request is sent to the UCA (as shown on figure 3), the reply will be received automatically.

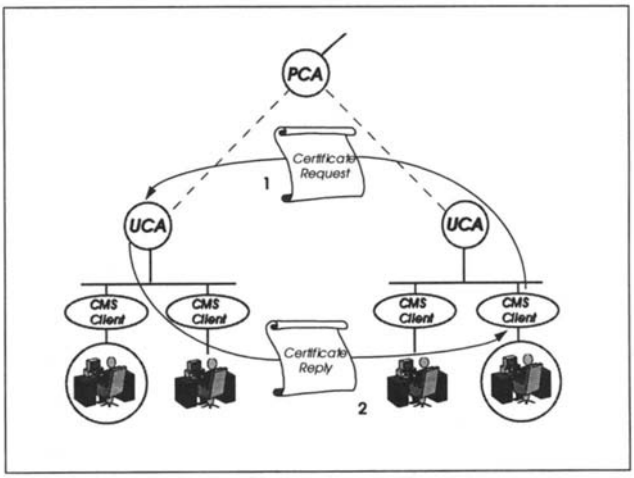

Figure 3 - Certificate Retrieval from UCA

If the address of the certificate owner is not available, then the CMS Client sends a *Resolve Certificate Request* message to the PCA (as shown on figure 4).

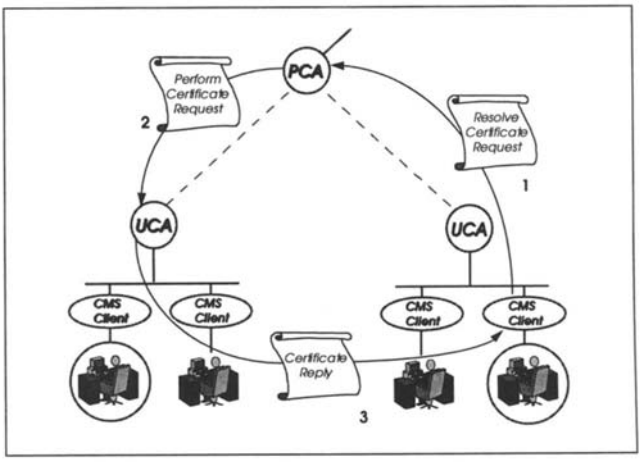

Figure 4 - Certificate Retrieval from PCA

Resolve Certificate Request contains the identification of the certificate owner in the form of a DN. Since every PCA administers all CAs in its sub-hierarchy, it has the ability to find the address of the UCA from the identification of the certificate owner. The PCA sends a *Perform Certificate Request* message to the UCA with the identification of the owner of the certificate and the address of the requester. The UCA, upon receipt of that message, behaves in the same way as if it has received a *Certificate Request* directly from the user. In other words, it sends a *Certificate Reply* message to the user as in the previous case.

3.4. Certificate Verification

The CMS Client must verify the integrity of every certificate that the security application uses. The integrity verification proves that no one tampered with the contents of the certificate, i.e. changing the public key, the validity date or the identity of the certificate issuer and owner. In other words, the signature of the certificate must be verified. In order to verify the signature CMS Client must posses the certificate of the issuer (the UCA). That certificate must also be verified in the same way using the next higher certificate in the certification path. This process is iterative and terminates when the trusted certificate is encountered, i.e. the certificate which has previously been verified in the same way, or when the top of the hierarchy is encountered. In the CMS system, which is hierarchical, the common point of trust is IPRA and its certificate is well known and available from different sources. Therefore the process of verification will in the worst case be completed when IPRA's certificate is reached. The validity dates of every certificate must also be checked.

The CMS Client receives certificates as messages *Certificate Signature Reply, Certificate Reply* and *Certificate Path Update* or from the security application itself. The *Certificate Verification* function performs the described process of verification and stores verified certificates in the local database for later use. If any of the certificates from the certification path are missing the *Retrieve Certificate* function is used to request it from the CMS system.

Furthermore the verification of a certificate must also include checks against current CRL list of the issuer to ensure that the certificate has not been revoked by the issuer. The CMS Client periodically retrieves all needed CRLs and stores them in the local database. If the current CRL list is not locally available at the moment of the verification, the CMS Client sends a *CRL Request* message to the PCA. The PCA responds with a *CRL Reply* message which contains all the requested CRL lists.

4. CMS CLIENT'S STRUCTURE AND APPLICATION PROGRAM INTERFACES (APIS)

The CMS Client is a part of the CMS system which enables an arbitrary security application to use services of the CMS system. It is designed in a modular way, so that the parts that rely on the underlying platform or message transport mechanism are modifiable without changing the main CMS Client part (see figure 5). It has a clearly defined set of APIs towards the security application and the communication layer. It includes a set of functions which handle a database of certificates, CRLs and users' identification in the form of DNs. Furthermore, it has an interface towards the user, in the form of dialogues and menus, which enables the user to directly issue commands to the CMS Client.

The CMS Client interface to a security application consists of a set of APIs which the security application uses to invoke the functions of the CMS Client. It is possible to either

build new security applications which use the CMS system services or extend existing ones with the CMS Client by adding calls to the CMS Client APIs. The most important CMS Client APIs for a security application are:

- *GetCMSStatus*: get the status of the authenticated user in order to decide whether the user is allowed to use functions of the security application,
- *GetCertificate*: obtain the needed certificate. The certificate is either found in the local database or retrieved as described in the previous section. It is possible to get the certificate together with its full certification path.
- *VerifyCertificate*: verify the certificate that has been retrieved by the security application itself. The verification is performed as described in the previous section and the certificate is stored in the local database after being verified.
- *CMSDialogue*: invoke the user interface part of the CMS Client.

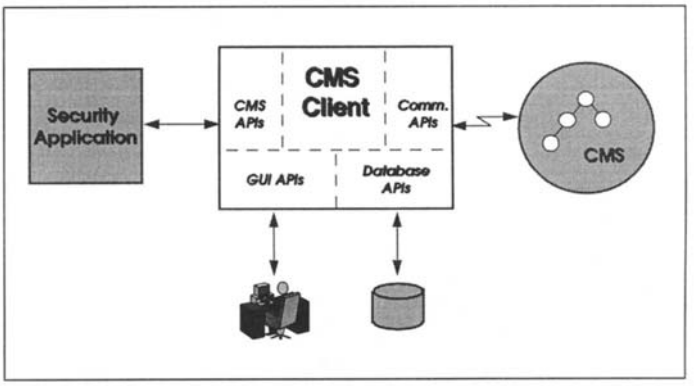

Figure 5 - CMS Client APIs

The CMS Client can also function as an individual application. It has a Graphical User Interface (GUI) which enables a user to directly invoke functions of the CMS Client. Using the CMS Client GUI it is possible to:

- Invoke an authentication dialogue which activates the *Authentication* function.
- Start generation of keys and certificate. The *Key Generation and Certification* function is activated only if user's CMS status is at least equal to REGISTERED.
- List the certificates in the local database.
- Explicitly request a certificate if it is not available in the local database.
- Reply to *Certificate Request* message.

The interface to the communication layer consists of APIs for sending and loading CMS messages. The interface is transparent to the underlying communication platform and is adjustable to different message communication mechanisms.

The database handling performs local storage of certificates, DNs and CRLs. These functions are structured as a separate module which may be ported to different computer platforms without imposing changes to the main CMS Client module.

The main CMS Client module is itself created in the form of functional modules. It is based on three main security modules: *Generalised Security Library (GSL), Smart Card Library (SCL)* and *Certificate Management Library (CML). GSL* contains security mechanisms such as prime and large number generators, symmetric cryptographic mechanisms, asymmetric cryptographic mechanisms and various message integrity codes. *SCL* performs RSA operations with smart cards of different vendors. *CML* implements coding and signing, verification and decoding of certificates according to the X.509 specifications.

5. CURRENT STATE OF IMPLEMENTATION AND FUTURE DEVELOPMENT

The following are the implementations based on the described certificate management system that already exist:

- an implementation of the CMS system which currently uses e-mail as the communication mechanism for the exchange of CMS messages,
- an implementation of the CMS Client for the PC MS-Windows platform in the form of several DLLs, and
- an implementation of one security application that uses services of the CMS system in the described way, namely an implementation of Privacy Enhanced Mail (PEM) system as specified in RFCs 1421 - 1424 [4-7].

Future development is planned in three directions. One is porting the CMS Client software together with the PEM application to UNIX and Macintosh platforms. Another direction is extending an implementation of the secure EDIFACT system [8], which is currently based on a two-level certification hierarchy, to use services of the global CMS. Finally, new communication mechanisms for the exchange of the CMS messages are being considered.

The author is one of the main designers of the CMS system extensions (as compared to the RFC 1422 model) and the CMS Client structure and its APIs.

REFERENCES

[1] CCITT Recommendation X.509, "The Directory - Authentication Framework", November 1988

[2] CCITT Recommendations X.500 - X.521, "Data Communication Networks Directory", November 1988

[3] Kapidzic, N., Davidson, A., "A Certificate Management System: Structure, Functions and Protocols", Internet Society Symposium on Network and Distributed System Security, February 1995

[4] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part 1: Message Encryption and Authentication Procedures", RFC 1421, DEC, February 1993

[5] Kent, S., "Privacy Enhancement for Internet Electronic Mail: Part 2: Certificate Based Key Management", RFC 1422, February 1993

[6] Balenson, D., "Privacy Enhancement for Internet Electronic Mail: Part 3: Algorithms, Modes and Identifiers", RFC 1423, TIS, February 1993

[7] Kaliski, B., "Privacy Enhancement for Internet Electronic Mail: Part 4: Notary, Co-Issuer, CRL-Storing and CRL-Retrieving Services", RFC 1424, RSA Laboratories, February 1993

[8] UN/EDIFACT Security JWG: "Recommendations for UN/EDIFACT Message Level Security", January 1993