Flexible Certificate Management in Public Key Infrastructures

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung der Grades
Doktor-Ingenieur (Dr.-Ing.)

von

Evangelos Karatsiolis
aus Athen, Griechenland

Referenten:	Prof. Dr. Johannes Buchmann
	Prof. Dr. Stefanos Gritzalis
Tag der Einreichung:	11. Januar 2007
Tag der mündlichen Prüfung:	27. Februar 2007

Darmstadt, 2007
Hochschulkennziffer: D 17

To my parents

Acknowledgements

Many people have contributed to the preparation of the thesis at hand. I thank

Prof. Dr. Johannes Buchmann for guiding me during my research and offering me a position at his research group. His instructions, encouragement, and support were valuable for accomplishing the thesis.

Prof. Dr. Stefanos Gritzalis for reviewing the thesis and providing me with constructive remarks.

Fraunhofer Institute SIT for supporting the first three years of my research.

My FlexiTrust colleagues Marcus Lippert, Markus Ruppert, and Alexander Wiesmaier. We worked very close together all these years in the FlexiTrust project. Their support, creative ideas, stimulating discussions, and proposals have affected this thesis significantly.

Tobias Straub for his proposals and our fruitful discussions.

Thilo Planz for the support he provided me in the FlexiTrust project.

Georgios Raptis for our everlasting discussions (mostly in Greek).

Martin Döring and Ilona Zemella for helping me with the preparation of the German abstract.

Alexander Wiesmaier for reading the first draft of the thesis and providing me with helpful comments.

Christoph Ludwig, Marita Skrobic, and Ralf-Philipp Weinmann for their kind help in many different aspects.

Ilona Zemella for supporting me with her love and understanding.

I am grateful to my parents and brother, who have supported me in various ways through the years. Their love has helped me and accompanied me throughout my life.

Zusammenfassung

Public Key Infrastrukturen (PKI) dienen der Absicherung einer Vielzahl von Anwendungen und Prozessen. Dazu gehören zum Beispiel E-Mail-Kommunikation, elektronischer Handel, Zugriffe auf Rechner und Netzwerke und digitale Identitäten für die Benutzung im E-Government oder im Gesundheitswesen.

Die verschiedenen PKI-basierten Anwendungen stellen unterschiedliche Anforderungen. Diese werden bestimmt durch das gewünschte Sicherheitsniveau, die Anzahl der Teilnehmer, die Software, die Hardwaregeräte, die Komplexität der Installation und durch viele andere Parameter.

Die vorliegende Arbeit konzentriert sich auf das Zertifikatsmanagement in einer PKI und schlägt verschiedene Lösungen vor, um die oben genannten Anforderungen auf flexible Art und Weise zu erfüllen.

Um die mit dem Zertifikatsmanagement zusammenhängenden Probleme zu lösen, führen wir die Certificate Management Authority (CMA) ein. Die CMA ist eine neue PKI-Komponente. Sie verwaltet Objekte, die von anderen PKI-Komponenten erstellt werden. Diese sind Zertifikate, Sperrlisten, PIN-Briefe, PSEs, u.a.

In der vorliegenden Arbeit werden der Entwurf und die Implementierung der CMA diskutiert. Die CMA benutzt Certificate Management Plugins. Diese Plugins können wiederverwendet werden, um interoperable PKI-Lösungen zu realisieren. Die Sicherheit der CMA wird bewertet.

Die Kommunikation in einem Trust Center wird untersucht. Ein neues Kommunikationsprotokoll wird entworfen und implementiert. Es dient der Kommunikation zwischen beliebigen Komponenten eines Trust Centers. Das Protokoll unterstützt Sicherheitsmechanismen wie digitale Signaturen, Verschlüsselung und das Vier-Augen-Prinzip. Es erlaubt visuelle Lesbarkeit der Nachrichten und die Darstellung typischer Daten in einem Trust Center.

Eine wichtige Aufgabe der CMA ist die Verteilung von PKI-Informationen. Dafür werden typischerweise LDAP-Verzeichnisse eingesetzt. Es wird ein Leitfaden für den optimalen Einsatz von LDAP-Verzeichnissen in einer PKI bereitgestellt. Die Anforderungen des deutschen Signaturgesetzes werden dabei berücksichtigt.

Zusätzlich verwenden wir LDAP-Verzeichnisse für weitere PKI-Managementfunktionen. Eine dieser Funktionen ist der Besitznachweis für Verschlüsselungs-

Schlüssel. Wir realisieren den Besitznachweis, ohne Bestätigungsnachrichten verwenden zu müssen. Darüber hinaus schlagen wir ein Protokoll zur Übermittlung von Software-PSEs an die Benutzer vor.

Abstract

A public key infrastructure (PKI) secures lots of applications and processes. These are for example the electronic commerce, email communication, access to computers and networks, or digital identities for use in e-Government or the health care sector.

The various PKI based applications have different requirements. These depend on the security level, the number of participants, the software or hardware devices, the complexity of the installation, and many other parameters. This work focuses on the certificate management in a PKI and proposes various solutions to meet these requirements in a flexible way.

In order to deal with the problems related to certificate management we design the certificate management authority (CMA). This authority is specified as a new trust center component involved in organising the workflow and the tasks that remain after the creation of a PKI product, like a certificate or a revocation list. Its design and implementation is discussed. The certificate management plugins, that the CMA is based on, can be (re)used to provide interoperable PKI solutions. We also give a security analysis of the CMA.

The new authority requires to rethink the communication possibilities within a trust center. A new protocol for communication inside a trust center is designed and implemented. It addresses the problems of communication of arbitrary trust center components. It enables human readability of the messages, security mechanisms like digital signatures and encryption, it supports dual control, and expresses typical data in a trust center.

One basic task of the CMA is the distribution and dissemination of PKI information. Typical solutions are based on LDAP directories. A best practice guide for these directories regarding PKI purposes is given. We further concentrate on the German Signature Act and see how to meet the directory related requirements in this context.

We will use the LDAP directories for other PKI management functions, too. One function is the proof of possession for encryption keys. This scheme realises the indirect method of the CMP messages, but without the need for any confirmation messages. We propose a second scheme for delivering software personal security environments.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit - mit Ausnahme der in ihr ausdrücklich genannten Hilfen - selbständig verfasst habe.

Wissenschaftlicher Werdegang

Okt. 1995	–	Okt. 2000	Studium der Elektro- und Computertechnik an der Universität Patras in Griechenland
Apr. 2001	–	heute	Doktorand am Lehrstuhl Prof. J. Buchmann, Fachbereich Informatik, Technische Universität Darmstadt
Jun. 2001	–	Jun. 2004	Wiss. Mitarbeiter am Fraunhofer Institut für Sichere Informationstechnologie, Darmstadt
Jun. 2004	–	heute	Wiss. Mitarbeiter am Lehrstuhl Prof. J. Buchmann, Fachbereich Informatik, Technische Universität Darmstadt

Publications

- V. Karatsiolis, M. Lippert, and A. Wiesmaier. *Using LDAP Directories for Management of PKI Processes.* In Proceedings of Public Key Infrastructure: First European PKI Workshop: Research and Applications, EuroPKI 2004, volume 3093 of Lecture Notes in Computer Science, pages 126-134, June 2004.

- A. Wiesmaier, M. Lippert, and V. Karatsiolis. *The Key Authority - Secure Key Management in Hierarchical Public Key Infrastructures.* In Proceedings of the International Conference on Security and Management, SAM '04, pages 89-93, June 2004.

- V. Karatsiolis, M. Lippert, A. Wiesmaier, A. Pitaev, M. Ruppert, and J. Buchmann. *Towards a Flexible Intra-Trustcenter Management Protocol.* In The Third International Workshop for Applied PKI, IWAP 2004, October 2004.

- V. Karatsiolis, M. Lippert, and A. Wiesmaier. *Planning for Directory Services in Public Key Infrastructures.* In Proceedings of SICHERHEIT 2005, volume P-62 of Lecture Notes in Informatics, pages 349-360, April 2005.

- A. Wiesmaier, M. Lippert, V. Karatsiolis, G. Raptis, and J. Buchmann. *An Evaluated Certification Services System for the German National Root CA-Legally Binding and Trustworthy Transactions in E-Business and E-Government.* In Proceedings of the 2005 International Conference on eBusiness, Enterprise Information Systems, e-Government, and Outsourcing, EEE'05, pages 103-108, June 2005.

- A. Wiesmaier, V. Karatsiolis, M. Lippert, and J. Buchmann. *The Workshop - Implementing Well Structured Enterprise Applications.* In Proceedings of the 2005 International Conference on Software Engineering Research and Practice, SERP'05, pages 947-953, June 2005.

- M. Lippert, E. Karatsiolis, A. Wiesmaier, and J. Buchmann. *Directory Based Registration in Public Key Infrastructures.* In The 4th International Workshop for Applied PKI, IWAP 2005, pages 17-32, September 2005.

- S. Fritsch, V. Karatsiolis, M. Lippert, A. Wiesmaier, and J. Buchmann. *Towards Secure Electronic Workflows.* In Proceedings of Public Key Infrastructure: Third European PKI Workshop: theory and practice, EuroPKI 2006, volume 4043 of Lecture Notes in Computer Science, pages 154-168, June 2006.

- M. Lippert, V. Karatsiolis, A. Wiesmaier, and J. Buchmann. *Life-cycle management of X.509 certificates based on LDAP directories.* Journal of Computer Security, 14(5):419-439, 2006.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

Public key infrastructures (PKIs) secure many aspects of modern communication. PKI products like certificates are the basis for enabling different security mechanisms. The TLS protocol [DA99] which is designed to secure internet communication uses certificates. Certificates can be met also in cases like downloading or updating of software programs, signed applets, secure email [Ram04b], virtual private networks, and many more other security solutions (see also Section 3.1). Commonly met profiles for certificates are the X.509 [IT97] and the PKIX [HPFS02].

The typical design of a PKI consists of three components, namely the Certification Authority (CA), the Registration Authority (RA), and the end-entity (see [AF99, Sec. 1.2]). The first two form the trust center which is a place where entities can request a certificate as well as a personal security environment (PSE). This design principle derives from the fact that the CA is usually offline. Therefore, registration is not located at the CA but to an extra authority. In addition, the registration process may be distributed in different registration offices while the CA is usually centralised.

This design however has some problems. Since the CA is offline it is difficult to manage its products like certificates or certificate revocation lists. These are transferred somehow from the CA to a system which is able to perform this on behalf of the CA. But even if the CA is online the same management needs still remain. There is no specification available on such a system. In this work we introduce the specification for this system called certificate management authority (CMA). We specify the requirements of this authority and see which tasks are assigned to it.

In addition, CMA is used to provide a backend for revocation authorities or systems that provide certificate status information, like OCSP [MAM$^+$99]. OCSP servers are online and an offline CA cannot be used to provide the necessary management of certificates and revocation information they require.

CMA also relieves the RA or different registration offices from managing the CA products, which is a common practice today. This practice however, requires that the products of the CA are transferred to the RA, something which is becoming more difficult if many registration offices exist. On the RA side, special software is installed, that administers the whole process. This requires special configuration as well as skilled personnel. The CMA offers a technical alternative solution to this process.

Moreover, CMA is used for realising business logic related to certificate management. Various organisations have diverse processes. A modular and flexible CMA is employed to implement and realise all these processes.

The design and implementation of the CMA will be presented. We will further examine its security properties. This is important since CMA is part of a security infrastructure. This is the PKI. In order to analyse the security of the PKI, both the PKI as a whole and its components are evaluated. Therefore the security requirements on the CMA in a certain context are identified and examined. This context is the German Signature Act. The security requirements of the CMA are expressed by certain security functions. We see how to to achieve and realise these functions.

CMA processes certain data that other trust center components produce. This data is transferred to the CMA in a flexible and secure way. A communication protocol among trust center components is needed that possesses these properties. Most typical solutions are the PKCS#7 [RSA93] or the CMP [AFKM05] format.

PKCS#7 has the problem that it provides a coarse transport mechanism for the data it encapsulates. It is a data container that does not have any mechanisms to distinguish among them. Security operations like signature and encryption are performed on this data. It can be used for any trust center component. CMP on the contrary expects three entities in the PKI configuration, namely the RA, the CA, and the end-entity. Therefore it is not suited for communication among arbitrary trust center components . In [Gut03, Sec. 5.2] Gutmann reports problems on the CMP. These regard its older version specified in [AF99]. In the newer version [AFKM05] some of these problems still exist.

In order to offer an alternative to the existent solutions in which these problems do not appear, we specify a new protocol. This protocol is based on the XML language and is used for the purposes of communication inside a trust center. The protocol does not make any assumptions on the number of entities, components or authorities in a trust center. It supports diverse PKI functions like certification or revocation. The data that can be transported is typical PKI products like certificates, revocation lists, or personal security environments. But it can be extended to other products as well. It has security mechanisms like digital signature and encryption.

We further use the messages of the protocol as the input format to the certificate management plugins. These plugins deliver services related to certificate management. They realise the tasks that are performed by the CMA. Due to their pluggable nature a very flexible mechanism to support certificate management is enabled. Moreover, a library of those plugins can be created in order to provide an extended functionality. Their operational and design considerations as well as the implementation of those plugins is shown. These plugins provide a solution to the problem of supporting various use cases and tasks that are assigned to the CMA.

Another important task in a PKI is the proof-of-possession (PoP) of private keys. This mechanism is needed when an end-entity sends its public key for certification to a trust center. The trust center determines if the corresponding private key exists, if it is in possession of the end-entity, and if the end-entity can use it. In case of signature keys, a signed request to the trust center (for example a PKCS#10 [RSA00] request) is sufficient. In case of encryption keys PoP is performed either by revealing the private key to the trust center or decryption of a value, that is usually the certificate. The first method is practically never used. The second method is used by most trust centers. It can be performed directly or indirectly. The direct method requires that the decryption takes place during registration by the physical presence of the key user. But this is not always possible. The indirect method allows that the decryption takes place after certification. It requires that the end-entity sends an extra confirmation message to the trust center. In large scale installations this can be difficult to administer.

In order to provide a method for realising the indirect method, we employ the LDAP directories. The scheme uses the properties of those directories, like access control lists, schema updates, attributes, or secure communication over TLS. The advantage of the protocol is that it does not require any extra messages to the trust center and therefore a certification request requires less time and iterations. This scheme is used in a smart card based project in the University of Darmstadt.

We will further use the LDAP directories for managing another PKI function that the CMA performs. This is the delivery of software based PSEs. One solution to this problem is to hand it over to the end-entity at the trust center. But this requires the entity's physical presence. Moreover, it does not scale with the number of participants. Another solution is to send it with email. But some users may have a full mail box that cannot accept any mails. Others may have a badly configured firewall that will reject the email. Our approach is to place the PSE on the entity's entry on the directory. Following this approach all the above mentioned problems will be removed. Furthermore, this scheme has two side-applications. The first one regards usability since the software PSEs can be installed and used automatically at the client side. Moreover, it enables a key on demand service, in which the users are able to access their keys from various computers whenever the need to use it.

Since publication and dissemination of PKI information is an important task that the CMA performs, we concentrate on the LDAP directories as the core component to achieve it. We see what are the best practices in LDAP directories to be used within the PKI technology. We examine their possibilities and limitations and propose the best current solutions. A planning guide for directory services will be given. We further analyse their use in a special context, that is the German Signature Act. We see what are the requirements for the directory and what features are needed in order to meet them. In order to realise all the necessary functions we design and implement an LDAP API for supporting PKI. This API supports publication of diverse PKI products and uses most of the LDAP security features. It meets miscellaneous requirements that have already been met in various PKI environments and installations.

The above described problems motivated this thesis and they constitute its main focus. This work is organised as follows. In Chapter 2 we discuss public key cryptography. A list of PKI-driven applications and the building blocks of a PKI is given in Chapter 3. In Chapter 4 we motivate the existence of CMA in a PKI and see what are the specifications of this component. We propose a new protocol suited for communication inside a trust center among arbitrary components in Chapter 5. A best practice guide for directory services and its use in the qualified signature context is discussed in Chapter 6. Two LDAP based schemes to support proof-of-possession for encryption keys and delivery of software personal security environments are described in Chapter 7. In Chapter 8 we give the design and implementation of the CMA. In Chapter 9 we examine the security properties of the CMA. Current installations of the CMA and its use in a concrete PKI architecture are discussed in Chapter 10. We conclude this work in Chapter 11 by giving a review of its contributions and possible future developments.

Chapter 2

Public Key Cryptography

Public key cryptography was discovered by Diffie and Hellman in 1976 [DH76]. Until that point, security of communication was based on secrets exchanged among the communicating parties. Moreover, the algorithms and techniques used were undisclosed and secret. From that point of time open research on cryptography has gained more interest. Especially, the new public key based techniques gave the ability to successfully address two problems. One for the key management and the other for providing digital signatures. The first digital signature scheme was invented in 1978. This is the RSA cryptosystem described in [RSA78]. Since then, the techniques and problems in public key cryptography have become more complex.

In the following we see which are the cryptography related security goals in a modern communication system. We will further take a look at the building blocks that are used to achieve them.

2.1 Security Goals

While designing a communication system, the security goals of this system must be defined. Examples of security goals are the integrity, confidentiality, or authenticity of the data. Once the security goals are identified a technical realisation of those is needed. As we will see, PKI is a technology that can reach these security goals with its techniques and standards.

2.1.1 Authentication

Authentication is the goal of identifying whether the information and data that reach a receiver, originate from a certain source.

Authentication is needed for example when users try to login into their computers. They have to provide some kind of information, that a computer programm verifies and if this is correct, it allows them to login.

The typical authentication protocol between two parties requires that the one party performs an action in order to authenticate itself or some data. The other party verifies this authentication information, and if verification is successful it is ensured for the authenticity of the other entity or origin of the data.

With public key cryptography there is the ability to provide strong authentication. Strong authentication refers to the authentication that is derived with the use of some private and secret information, namely the private key. The technique used to achieve this is called digital signature. With a digital signature the origin as well as the integrity of the data can be ensured.

2.1.2 Integrity

Integrity is needed in order to identify whether some piece of information has been altered or not. This is important for example while uploading or downloading a file on a server. With integrity check someone can be sure that the data has not been altered, damaged, or corrupted during this transfer. This could be a software download or the uploading of a paper on a conference server.

Digital signing of data offers also integrity services. The employment of cryptographic hash functions within the digital signature provides this kind of services. Message Authentication Codes (MACs) can also be used in order to achieve integrity of data.

2.1.3 Confidentiality

Confidentiality of data is the goal of keeping this data secret. This may be needed during storing, archiving, or while sending this data to another user. For example patient data must always be kept secret and only an authorised doctor is allowed to see this data. Or the number of a credit card in an online transaction must be sent encrypted. The technique used to achieve this goal is encryption.

Confidentiality is the oldest security goal. The first cryptographic mechanisms and techniques were implementing some kind of encryption. Modern techniques either use public key cryptography or symmetric schemes.

2.1.4 Non-repudiation

Non-repudiation is when an action cannot be denied from the persons who performed it. In addition non-repudiation of data is when the origin of the data can be assigned to a certain entity. The technique to support this is the digital signatures. Public key cryptography is the only technology that can reach this security goal. In cases where the originality of a digital signature is questioned, a non-reputable signature proves the identity of the entity who has calculated this signature.

Electronic signatures that are legally binding (equivalent to the handwritten signatures) must always be non-reputable. Such signatures are for example signatures that can be verified from a qualified certificate. As qualified certificate we denote a certificate that is conform to a signature act. The Signature Act in Germany [Leg01a] describes the properties of a qualified certificate.

Non-repudiation is the most debated security goal. It is said that it cannot be technically achieved [Ell02]. However, a careful user as well as secure procedures, both human and technical, as described for example in various signature acts used in several countries can achieve this goal. Furthermore, it is argued that since non-repudiation makes the users of the key responsible for its use, the users may never want to possess such a key. This would have as a consequence that the users will not use PKI because it is a non-repudiation enabling technology. But it is in the interest of the users to have such a technology for their advantage. In the electronic world where contracts or documents must be digitally signed, it is extremely important for the users to be able to prove that they have signed the contract or are the owner of a document. This is achieved by non-repudiation.

2.2 Public Key Cryptography

2.2.1 Digital Signatures

Digital signatures are used in order to provide authenticity and non-repudiation. They are based on public key cryptography. When users want to calculate a digital signature they do so by using their private key. In order to verify this signature the corresponding public key is needed. The private key is known only to the user that digitally signs, while the public key is known to everybody. This public key is usually found inside a digital certificate like an X.509 certificate. The notion and need of certificates was described in the work of Kohnfelder [Koh78] in 1978.

The most prominent among all signatures algorithms is the RSA Algorithm. RSA was discovered in 1978 from Rivest, Shamir and Adleman [RSA78]. Since then RSA has become a de facto standard. Almost every security application that uses public key cryptography is implementing this algorithm. In many cases and in many products it is the exclusive algorithm to be used. Although other algorithms and schemes exist, that are more efficient than RSA, RSA survives the competition. This is due to its wide spreading in security products, its simplicity, and its security.

The security of RSA is related to the problem of factoring numbers. If these numbers are not large enough RSA can be broken. In order to overcome this problem the key size used in the algorithm is becoming larger as factoring advances. In the quantum computer era the factoring and the discrete logarithm problem will be solved in polynomial time as proved by Shor in [Sho94]. This is a big threat for many cryptosystems including RSA. Alternatives are needed. Public key infrastructures, both the trust center and clients, must be able to use other schemes, too.

In the following we describe the RSA algorithm:

The RSA Algorithm

1. Find p and q, two large prime numbers.
2. Compute $n = pq$
3. Find e with $1 < e < (p-1)(q-1) = \phi(n)$ such that $gcd(e, \phi(n)) = 1$
4. Using the extended Euclidean algorithm, find d with $1 < d < \phi(n)$ such that $ed \equiv 1 \bmod \phi(n)$
5. Output the pair (n, e) as the public key.
6. Keep d as the private key.

Using d Alice can now digitally sign (and decrypt) messages. Signing with RSA will be discussed in Section 2.2.3 and encrypting in Section 2.2.5.

Other Algorithms

Apart from RSA other algorithms are used, in order to get digital signatures. One is the Digital Signature Algorithm (DSA). DSA was published as a FIPS under the name Digital Signature Standard [NT94]. Its security is associated to the discrete logarithm problem (DLP). The specification for the algorithm can be found in [NT94].

There is a variant of DSA on elliptic curves. It is called ECDSA. Its security is related to the difficulty of the elliptic curve DLP (ECDLP). Since this problem is considered more difficult to solve than the DLP, the keys sizes in this cryptosystem are smaller for the same level of security. This scheme is described thoroughly in [JMV01].

2.2.2 Cryptographic Hash Functions

Hash functions have an arbitrary message as input and a message of fixed length as output. If they are also one-way they are called cryptographic hash functions.

If $m \in \{0,1\}^*$, then the cryptographic hash function h is a function such that:

$h(m) = s$ where $s \in \{0,1\}^n$, with n the fixed length of the hash function output. The one-way property is that from s it is difficult to calculate m.

Usual lengths for the output of a cryptographic hash function are 128, 160, 192, 256 or 512. Most known and used functions are MD5 [Riv92], SHA1 [ErJ01] and RIPEMD-160 [DBP96]. New cryptanalytic attacks on MD5 show that this function has become insecure [WFLY04], [WY05], [Kli05a], [Kli05b]. Attacks on SHA1 have become easier [WYY05]. Therefore, the SHA-224, SHA-256, SHA-384 and SHA-512 variants (also known as SHA-2 family) should replace this hash function in the near future. These are specified in [NT02].

Properties

A hash function that is used in cryptography must have the following properties:

- Pre-image resistance

  This denotes that given the hash value $h(m)$ of a message m, it is difficult to calculate m.

- Collision resistance

  It denotes that it is difficult to find two messages m_1 and m_2 that their hash values are the same.

- Second pre-image resistance

  This denotes that given a fixed message m_1 it is difficult to find another message m_2 that has the same hash value.

Anderson extends the catalogue of requirements and properties for hash functions in [And93]. The security of cryptographic hash functions is extremely important. If these functions become insecure the schemes using these functions will become insecure, too. This includes the electronic signatures.

2.2.3 Calculating an Electronic Signature

Digital signature algorithms are not used in their physical form in order to produce an electronic signature. They are used in conjunction with cryptographic hash functions. Firstly, a user hashes the message to be signed with a cryptographic hash function. Afterwards the user digitally signs the hashed value by using the private key of the signature algorithm. Verification of the message is performed by hashing the original message and comparing this value with the value obtained by using the public key over the signed message.

In the case of RSA this is the following:

- Signature Creation
  1. Calculate the hash value $h(m)$ from the message m.
  2. Calculate the signature s $= (h(m))^d \bmod n$ with the private parameter d.
- Signature Verification
  1. Calculate the value $v = s^e \bmod n$ with the public parameters e and n.
  2. Calculate the hash value $h(m)$ from the original message m.
  3. Compare v and $h(m)$. If they are the same then the signature is correct. Otherwise false.

In this example we have used the RSA cryptosystem in its initial form. In practice in order to work with RSA based signatures the PKCS#1 [RSA02] standard is being used. This allows the encoding of the values used in the signature process. One advantage of this is that greater interoperability is achieved. For example the signatures have a fixed length. Moreover, the signature scheme becomes more secure. For a brief discussion on the security properties gained and the attacks they provide defence from see [RSA02].

Therefore RSA and other cryptosystems are not usually being used in their initial pure mathematical form for creating an electronic signature. In addition the hash functions they are used in conjunction with, affect the signature scheme. That is the RSA with the use of SHA1 is another signature scheme than RSA with the use of MD5. These schemes have been assigned well known names or aliases when they are used in various applications. For example "SHA1withRSA" is the alias for a PKCS#1 encoded RSA signature with SHA1 used as the hash function. More standardised naming and addressing of the algorithms is used in practice. This is the OID (Object Identifier) referencing.

The OID is a global, unique sequence of numbers that identifies an object. Such an object can be any arbitrary object, in this case a signature algorithm. For SHA1withRSA this identifier has the value "1.2.840.113549.1.1.5". Therefore, applications that are aware of this OID know the algorithm being used (for example in a cryptographic protocol). In the previous example therefore, the signer apart from the signature and the message, should also send the OID of the algorithm used in order to enable verification of the signature.

2.2.4 Message Authentication Codes

Message Authentication Codes (MACs) can be used in order to ensure the integrity of a message and in addition provide authenticity for this message. They use a symmetric secret key as a parameter. This key is called symmetric since it can be used for encryption and decryption. Thus, the recipient of the message possesses exactly the same key as the sender of the message. The first needs it for calculating the MAC value of a message and the second for verifying this MAC value.

MACs cannot be used for signature purposes. This is because at least two parties possess the same secret key. Therefore, it cannot be distinguished which one has calculated the MAC value. In addition, verification requires also that the key will be revealed.

A typical application that uses MAC is the PKCS#12 container. PKCS#12, defined in [RSA99], offers the possibility to store secrets and private information (usually private keys) in a special container. It has security mechanisms to provide a secure store and specifies a standard format for providing interoperability among applications. PKCS#12 offers two modes for integrity of information and two for privacy. These modes can be combined and therefore four mechanisms exist in total. One integrity mode is based on a password. This mode uses a MAC algorithm. The password is used in order to calculate the symmetric key for the MAC algorithm. The algorithm that is used in this standard is the HMAC.

HMAC is described in [KBC97]. This MAC algorithm is based on hash functions. Other MAC algorithms that are widely used are the CBC-MACs. These are based on symmetric block algorithms operating in the CBC (Cipher Block Chaining) mode. Another MAC function is the UMAC described in [BHK$^+$99].

2.2.5 Encryption

Public key cryptography is also used for encryption. For encrypting a message, the sender must know the public key of the recipient. This key is used in order to encrypt a message. Then the message can be sent over a possibly insecure channel. The recipient is using the private key for decrypting the message. In the case of RSA this is as follows:

- Encryption
    1. Encrypt the message m with the public key of the recipient e. That is $c = m^e \bmod n$. Send c to the recipient.
- Decryption
    1. Decrypt c by using the private key d. That is $m = c^d \bmod n$.

The RSA cryptosystem can be used for encryption and digital signing. As in the case of digital signing, the RSA cryptosystem is not being used in its raw form. The PKCS#1 encoding of the messages is employed for the encryption process. That provides also defence against certain attacks [RSA02].

In most cases public key encryption is used only for small messages. This is because it is slow in comparison to secret key techniques. Such techniques are the Triple-DES (3-DES) or the AES algorithms. Public key encryption in this case may be employed for encrypting a secret key that will be used in the rest of the communication for encrypting the data. This is met for example in the TLS [DA99] protocol.

2.3 Further Reading

We have briefly visited the cryptographic building blocks we will use in the rest of this work. For more information, details and security analysis see [Buc01] and [Sma03]. For a work covering many cryptosystems and cryptographic techniques see [MvOV97].

Chapter 3

Public Key Infrastructures

Public key infrastructures are used for providing security services in lots of applications. The PKI is a technical realisation of different aspects of public key cryptography like being able to sign, verify signatures, encrypt, or decrypt. It is an infrastructure in which public key based cryptography can be operated.

PKI is an infrastructure. This means that it needs certain protocols, building components, software and hardware programs, and techniques that are commonly accepted in order to operate and function properly and securely. We will see the services that a PKI offers, the entities it consists of, and what are the responsibilities of each entity. We will also present the most typical PKI products like X.509 certificates. We will give the notion of a trust center, and see its operating considerations. But first we will take a look at some PKI enabled applications and see where PKI can be employed in order to provide security solutions.

3.1 Applications

A PKI is the technical realisation of an infrastructure in which public key cryptography can be operated. Many schemes require the use of public key cryptography, like digital signatures. There are a lot of applications that are based on these schemes. Legally binding signatures for example are using digital signatures. Therefore, a PKI enables these applications. In the following we see some of these PKI enabled applications.

3.1.1 Network Security

The network traffic and communication on the internet can be secured with the help of a PKI. Particularly the confidentiality, authenticity, and integrity of the information are important. The technology that is used to perform this is the SSL and TLS protocol.

The TLS protocol, is a protocol that secures the communication of two parties by enabling them to authenticate to each other. Moreover, it encrypts the data and traffic between them. The specification of the protocol can be found in [DA99]. TLS is the successor of the SSL protocol which provides almost the same security services. Further these protocols can be used in a variety of other protocols like HTTP [Res00] or LDAP [HMW00]. For more on SSL and TLS see [Eck04, Chap. 12.5] and [Res01].

Actually, these protocols make possible a family of applications. We briefly visit two of them:

E-Business

Most of the web-based electronic commerce requires the use of a secure technology. For example a web-site selling some products must identify itself as the correct seller. Moreover, when the clients must pay for an item, they have to provide some private information like their name or address. These must be secured for confidentiality. This is also needed if the payment is done electronically. In this case the credit card numbers or other banking information must not be sent in clear. All these can be achieved by TLS.

Secure Examination Registration

Another case where TLS can be used is for secure sign up for an examination over the internet. In this case the students must be sure that they send their application to the correct examination authority. Likewise, the examination authority must know the identity of the student and be sure that exactly this student is signing up for an examination. The server and client authentication mechanisms of TLS can provide these mechanisms.

3.1.2 Document Archiving

Document archiving and especially long-term archiving is another application of digital signatures and PKI. Long-term archiving of documents is usually needed in government administration, medical documents, or adminstration documents, for example in the corporate environment.

In the federal state of Lower Saxony (Niedersachsen) in Germany about 130 millions of paper pages are used for purposes of state administration every year.[1] These must be archived for 5 up to 30 years. After this period, about 3% to 5% of these documents will be permanently archived. The processes, procedures and workflows in the administration are being digitised and the documents are transferred to an electronic form. But the traditional signatures on the document must also be digitised. Therefore electronic signatures are employed.

[1] `http://www.izn.niedersachsen.de/master/C5252172_N5505837_L20_D0_I3654280.html` (date of access 28.10.2005).

A project was initiated in Germany for addressing the problem of long-term archiving of electronically signed documents. This is the "ArchiSig" project.[2] One problem related to long-term security of digital signatures is that the cryptographic primitives may become insecure. Another is the possible lack of information needed to verify such signatures after a long time. In [BPRS02] the technical aspects of these points are being discussed in the context of the German Signature Act [Leg01a]. In [BvdHH$^+$02] a prototype implementation of a PKI in the health care sector is presented. This PKI will provide the necessary infrastructure, in order to support electronic signatures for the electronic records of the patients.

3.1.3 Code Signing

This is one of the PKI enabled applications that standard users are often confronted with. The goal of code signing is that the users can verify the source of the code and install programs only from trusted parties.

A software manufacturer must digitally sign the software that it distributes. When the software reaches the user, in the form of an installation CD or as software download or update from the internet, the user must first verify the signature of the software. Only if the signature is valid the user should install this software. In the Windows operating system there is the possibility to digitally sign the Windows installer packages. For further reading on software download code signing for Windows see [FFW99, Chap. 4]. In some Linux systems the whole distributions, or updates, or patches can also be digitally signed.

Some special form of code signing can also be met in applets. Applets are Java programs that run in a Java enabled browser. The binary distributions can be signed in order to access resources on the client system. A typical example is the reading and writing of files in the client filesystem. While an unsigned applet is not allowed to perform such operations, a signed applet may be. Other examples are the establishing of connections to the internet or communicating with external ports (for example the serial or parallel port).

Another application of the code signing can be located in WebStart [SUNb]. WebStart is a technology that allows to write and distribute applications over the network. A client connects to the internet and downloads the WebStart application. After that the application runs locally on the client machine (and not in the browser like in the case of applets). In addition it has a flexible update mechanism. The client always checks for a newer version on the host (where the WebStart application can be located). As long as the application remains the same, the client can be run locally without accessing the network. But as soon as the application is updated, the client can also be updated with the newer version. This enables an easy adding of new features or fixing bugs in the program.

[2] `http://www.archisig.de/` (date of access 09.02.2006).

A WebStart application must be digitally signed if it needs to access restricted resources to the client system. Most WebStart applications available, do require such an authorisation and therefore they are usually signed. An example for such an application is the Card Manager. This is an application that communicates with a smart card in order to perform certain functions with it. This application has been developed in the Technical University of Darmstadt for the purposes of administrating the student card.[3]

3.1.4 Secure Email

With secure email the possibility to send signed and encrypted messages is denoted. Many email clients support secure email. Typical applications of this is the sending of encrypted mails containing sensitive data (for example parts of source code or a system's specifications) in the corporate environment. Another application is the authenticated communication between communicating parties. A concrete example is that of students sending a signed email to the university's secretary in order to declare a change in their address or ask for some kind of certification.

S/MIME is the most widely deployed standard to provide security in email. It is specified in [Ram04b]. Its current version is the 3.1. S/MIME is based on X.509 certificates as this is specified in [Ram04a]. This standard can also be used for other purposes like securing electronic data interchanged (IDE) which denotes the data exchanged among different corporations and institutions. For more on S/MIME and related security mail standards see [FFW99, Chapt. 5]. Apart from S/MIME, Privacy Enhanced Mail (PEM) and PGP can be used to provide security services for emails.

But other applications can be driven from email security. Email security can be used against unsolicited commercial e-mail (SPAM). Only messages that contain a valid signature may appear in a person's incoming mailbox.

3.1.5 VPN

Virtual Private Networks (VPN) are networks that operate over public networks but enable security mechanisms for the communication inside this network. VPNs can be built with the help of IPSec [KA98]. IPSec addresses the problem of providing security at the IP (Internet Protocol) level. The sixth version of the IP (IPv6) specifies two different security mechanisms that must be supported [DH98]. This means that security mechanisms are very important for the internet communication and therefore they are part of its specification.

A typical application of VPN networks is for securing connections from one network to another. This could be a LAN or a Wireless LAN. Another application

[3]It can be downloaded from `http://www.tu-darmstadt.de/hrz/chipkarte/dokumentation/cardmanager/` (date of access 09.02.2006).

is the secure connections for intra-organisation purposes like that of connecting remote branches. For inter-organisation purposes VPN connections may also be employed.

Usually a PKI is needed for the authentication parts of the VPN connection. The clients can authenticate themselves using strong authentication with the use of public key technology. The users possess an X.509 certificate that enables verification of their signatures. PKI based authentication offers more security than the more simple variants. For more on this see also [Car02, Chap. 11].

3.1.6 Filesystem Encryption

The filesystem of a computer can also be encrypted with the use of PKI technology. Usually, in this case, a symmetric key is encrypted with a public key. With the symmetric key the filesystem encryption is achieved. This is a hybrid scheme in order to use the best properties of each technology. The public key encryption for key management and the symmetric encryption for speed.

In most of the standard operating systems a filesystem encryption scheme exists. In Windows OS this is the Encrypting File System (EFS) scheme. EFS offers the possibility to encrypt and decrypt portions of the filesystem. The details as well as an analysis of the EFS can be found in [FNG03]. Similar schemes exist for other operating systems. A survey on file systems that use cryptography is given in [WDZ03]. The authors also discuss and compare the performance of some of those systems. In [JCB03] the design and implementation of an encrypted file system suited for NAS (Network Attached Storage) is presented.

3.1.7 Login

The login into a computer system can also be secured with the use of certificates. Many variants of secure login use smart cards. This application has attracted attention lately, especially in the corporate environments. But also in the university context is of great importance in order for the students to login at the computer labs.

A smart card based login system was implemented from Microsoft at their Windows 2000 Server product. This is called Smart Card Logon and was implemented for authentication and login purposes at a Windows operating system. Windows XP Professional support this feature, too. At the moment this solution supports only domain and not local accounts. In the TU Darmstadt the Novell Client is used for providing login services to the computers found all over the university campus. This system also uses smart cards. It requires a PKCS#11 [RSA04] library for communicating with the card.

3.1.8 Electronic Voting

Digital certificates and PKI can also be used for electronic voting. The voting process can be performed from the distance without requiring that the voters must visit special voting centers. Simple digital signatures as well as blind signatures [Cha82] (to ensure the anonymity of the voter) are employed to implement the voting schemes. But other techniques exist, too.

A project regarding electronic voting in Austria is using PKI technology to enable these services to the citizens. A detailed description of the project's goals, but also technical issues can be found in the internet site of the project.[4] The proposed carrier for the keys is a smart card which is already in use in Austria in the form of a citizen card.

3.1.9 Measuring Data Exchange

Measuring data from machines and services must be further processed in several cases. For example in the case of power supply providers, the consumption of the consumers must be measured and a bill must be sent to them afterwards. The authenticity and integrity of this data is of great importance. Since this data is usually in electronic form, digital signatures can be employed to provider these security services.

A description of a project, called SELMA, dealing with this problem can be found in [DHIR05]. Digital signatures and X.509 certificates are employed in this project in order to realise a secure electronic measuring data exchange. A digital signature is calculated from the measuring machine over the raw measuring data. In the next steps this data is processed only after a successful signature verification and a new signature is calculated after every processing step. Lastly, the billing information is sent to the end-user also in a signed electronic form. For more information on this project see [DHIR05].

3.1.10 Citizen Card

Various e-Government projects are based on digital identities stored on smart cards. Many European countries issue or plan to issue citizen cards. In [CWP04] a report on the Belgian citizen card can be found. In addition, signature acts in many countries require certain devices to be used as the secure store for the keys as well as the secure place where the signature calculation will be performed. Such devices are smart cards among others.

[4] `http://www.e-voting.at` (date of access 13.04.2005).

3.1.11 Electronic Tax Declaration

In Germany the tax declaration can be performed electronically. The framework on which this is performed is called ELSTER. ELSTER requires that all declarations must be sent digitally signed to the authorities. The authorities provide a special programm (which can be downloaded from their web site) with which all tax declaration data can be signed and transmitted to them. This programm is called ElsterOnlineManager (EOM).

EOM is working with smart cards able to calculate digital signatures. This digital signature replaces the handwritten signature and has the same value. At the moment not all federal states in Germany support the digital signing and transmission of the tax declarations. The EOM will be substituted with a newer system in 2006. For more on this system visit the ELSTER site.[5] In the federal states of Hesse (Hessen) and Berlin tax account related data can be retrieved online.

3.1.12 Virtual Post Room

This is another project in the German e-Government planning. The main idea of the project is to develop a central communication gateway in order for citizen to securely communicate with the German authorities. Not only citizens but other authorities and corporate institutions can communicate using this gateway. The data that is exchanged between those communicating parties is of private nature and it is encrypted. In addition it is digitally signed to identify the person who send this data. PKI delivers its services in this case, too. The certificates needed to perform the above must be qualified certificates according to the German Signature Act [Leg01a]. A description of this project as well as its security concepts can be found in [MM05].

3.1.13 Paperless Work Certifications

Lots of certifications must be issued during a person's professional life. This could be a certification containing salary information or a certification regarding employment or pension issues. At the moment about 60 million of these certifications are issued in Germany every year. Almost all of them are in paper form. The administration of this information is quite difficult to achieve.

With the JobCard[6] project in Germany, it is planned that all employees will receive a smart card. This card will contain a qualified digital certificate. With this certificate the digital signatures of the users can be verified in order to authenticate themselves. Afterwards, they can request a certification regarding their employment. The whole process is digitised, compact, secure, and not time

[5] `http://www.elster.de` (date of access 13.04.2005).

[6] `http://www.itsg.de/download/BroschuereJobcard.pdf` (date of access 13.04.2005).

consuming. More applications are planned. The beginning of the project was announced for 2006 but considerations whether it should be combined with the health insurance card have delayed the project.

3.1.14 Health Care Sector

The health card is another PKI enabled project. This card will substitute the currently existing health insurance card in Germany. This card will hold all previous data needed in the health care system. Apart from this, it can also be used for the electronic issuing of prescriptions. These prescriptions must be secured with digital signatures and encryption. Moreover, the doctors will be able to access patient data by authenticating themselves. In [Cha99] a discussion on the advantages and disadvantages of the introduction of a smart card in the health care sector are discussed.

3.1.15 Electronic Measurements

Electronic measurements from devices like scales require monitoring. This is important since involving parties should be able to examine the correctness of the measurement, because transportation and billing of the goods is associated to the values measured from the machines. In order to achieve monitoring of the measurements, special controlled machines are usually used. These are controlled from the corresponding measurement office authority. PKI can be used instead to secure the data itself for authenticity and integrity. A description of such a process can be found in [WRL+06]. Such a system is already in use from the Schenk Process GmbH company located in Darmstadt. This process is patent pending.[7]

3.1.16 Electronic Workflows

Various workflows in organisations and companies are digitised in order to achieve paperless administration, faster procedures, high availability, and other features. The security of the workflow is important because design goals like confidentiality or authenticity exist. In traditional workflows these goals are achieved by hand-written signatures, policies, and various access control mechanisms. When these workflows are digitised, PKI is employed for achieving these goals. In [FKL+06] it is shown how to transfer traditional workflows in a university to electronic ones, what are the security requirements and how to address them with the use of PKI. The prototype implementation of a complex workflow, that is the appointment to a professorship, is given.

[7]European Patent: EP 1 450 144 A2.

3.2 Public Key Infrastructure

The goal of a public key infrastructure is to enable certain security services and achieve security goals by deploying the public key technology. The applications given above, are enabled through the installation of a PKI.

Like any technical infrastructure PKI has specifications in order for the users to be able to use it. Moreover, it bases its services upon products. We will examine what are the specifications of PKI and its products.

3.2.1 The Infrastructure

A PKI is a software and hardware based infrastructure. Once it is installed, its services can be used either from the hosting organisation or other external organisations. This means that it is not necessary to build and install such an infrastructure whenever its services are needed. The most typical example is the use of SSL. Although most of the organisations that are working on banking or e-commerce operate their web services over SSL, they do not have a PKI installation. Instead they are using PKI services from external organisations.

PKI is mostly a commercial product. There are very few open source and free available PKI solutions. One is based on OpenSSL [Ope05]. This PKI is a minimal one. It provides only the basic functionalities. Another project is the OpenCA [Lab05]. This project offers more functionality. It takes about two weeks to install it for two software engineers according to [FMS$^+$05]. Both projects cannot be used in their status today in order to address complicated problems that appear in a PKI. The complexity of a PKI software scales with the options available. For operational reasons mainly, but also security considerations, PKI services are usually distributed. This is a common practice in almost every large scale infrastructure independent of its type. We see how the services in a PKI are distributed.

3.2.2 Registration

The purpose of registration is to obtain an entity's data, verify them, and initiate the certification process. This data will be used later for binding an entity to a public key.

The first task of registration is to obtain the data. This can be done for example by obtaining it directly from the entity. This usually requires the entity's physical presence at the registration office, if this data are not already known. Another way to obtain this data is by an electronic database that already contains it. The data is then extracted and used.

The second task is to ensure that this data is correct and valid. That is to identify and authenticate the entity. Failing to do so will be a major risk in the security of the whole infrastructure.[8] One way to achieve a correct identification is by checking an official document like an identification card or a passport. If the data is obtained from an electronic database, then unauthorised changes in this database must be restricted. In addition, a mechanism to ensure that this database is the one that contains this information should also be present.

The authority responsible for the registration is called the registration authority (RA). After a successful registration, the certification follows.

3.2.3 Certification

Certification is the process of binding an entity to a public key. The data identifying the entity comes from the registration process. The public key is provided by the entity or it is created during registration or certification. In the last case the private key must be delivered to the end-entity.

The binding of the entity's identity and its public key is realised by using a digital certificate. This certificate contains at least these two values. It is electronically signed by an authority in order to state that the information it contains is correct and has been examined by this authority. Therefore an application that trusts this authority trusts the information contained in the certificate. That is that a public key belongs to a certain entity. Moreover, since the certificate is signed, the integrity of the data that it contains is achieved. Responsible for the signing of the certificate is the certification authority (CA).

Special security mechanisms are applied to the CA. This is because of the importance of the task the CA is assigned with and the security considerations associated with it. Special protection is needed for example for the private key that the CA uses in order to sign the certificates.

3.2.4 Revocation

Revocation is the action of revoking a certificate and cancelling therefore its validity. This is usually initiated from the entity that is using this certificate. The revocation information must be available for the PKI entities. For achieving this, a known position where such information can be obtained must be provided. This can be realised in the form of a list that contains all revoked certificates and it is located in an LDAP directory. It may also be in the form of a positive or negative answer upon a question whether this certificate is valid or not.

The online certificate status protocol (OCSP) offers such services. OCSP is defined in [MAM$^+$99]. It is a protocol that allows clients (OCSP clients) to query a server (OCSP server) and get signed responses about the status of a certificate, that is whether it is revoked or not.

[8] This has already happened. For a discussion on the security risks see `http://www.microsoft.com/technet/security/bulletin/MS01-017.mspx` (date of access 14.11.2005).

All these methods must be protected for authenticity and integrity. Therefore, electronic signatures are employed in these methods, too.

3.2.5 Distribution

All public products and information of a PKI must be publicly available for all users. The most commonly used mechanism to achieve this is to publish such information on a public directory. This is usually an LDAP directory [HM02]. From this directory this information can be accessed from all PKI participants.

Private information may also have to be sent to end-entities. This is met for example when the private keys are created in the CA. Secure and reliable mechanisms are needed in this case.

3.2.6 Miscellaneous Tasks

There are other miscellaneous services and tasks in a PKI. Some of them are related for example to validation. A server performs validity operations for a certificate on behalf of a client that chooses to delegate this task to the server. This includes verification of the certificate's signature, locating the certification path to the higher level CA, as well as evaluation of revocation information. A specification for such services is the XML Key Management Specification (XKMS). But other services exist or will appear in the future

3.2.7 The RA-CA Model

In order to realise the core services that a PKI offers the RA-CA model has been introduced. The end-entity is a third component. This is the one using the PKI services. A description of each component can be found in [AFKM05, Sec. 3]. Many other specifications employ this design. This is for example the profile specification of certificates and revocation lists [HPFS02]. Also in [HP01, Chap. 5] the same design is used.

The RA is responsible for the registration. It verifies the data of the entity and creates a request for certification to the CA. Additionally, it may produce revocation requests for certificates, either initiated by itself or the end-entity. It can be distributed in the form of local registration authorities (LRA) in order to meet certain operational needs inside organisations. The requests are gathered from every registration office and they are sent to the CA. See Chapter 5 for a discussion on the possible ways for transferring messages between RA and CA.

The CA is dealing with the certification. The data that comes from the RA is used to create a certificate. Additionally the CA may also sign revocation lists. It is usually centralised and special protection mechanisms are applied to it. The certificates and revocation information are extracted from the CA and are disseminated to the end-entities. The RA or human administrators are responsible for this.

Alternatives to this design have been proposed in [FS03, Chap. 19]. According to this design the RA is actually a subordinate CA to the root CA. This design however is not discussed in details. Often, the presence or absence of certain components is debated among PKI practitioners. Reasoning for the existence and necessity of an RA can be found in [AFKM05]. The work at hand is about introducing a new authority and component in the infrastructure. This is the CMA. The CMA completes the RA-CA model where this cannot provide satisfactory solutions (see Section 4.1). Other authorities and components have already been proposed. One is the key authority [WLK04]. It is responsible for administrating the private keys of the infrastructure. Another is the decryption authority described in [Bao00]. It is involved in decrypting messages for the users. The validation authority (VA) exists also. The VA is responsible for providing statements whether a certificate is valid or not. Actually, the verification and validation steps are delegated from the end-user to this authority.

3.2.8 End-entities

End-entities are the ones who are using the PKI services. They possess a certificate. By possessing a certificate they can use PKI driven applications like secure email, VPN and many more. They interact with the RA in order to request and get a certificate. The more transparent and easy for those entities the use of certificates is, the greater the acceptance of the infrastructure will be.

End-entities can be physical persons and machines. In the case of SSL server authentication the end-entity is the web services hosting computer. In the case of qualified electronic signatures the end-entity is always a physical person.

3.3 PKI products

3.3.1 X.509 Certificates

The certificates are the most prominent among the PKI products. Without certificates the whole PKI would be infeasible. They are signed documents that certify that a public key belongs to a certain entity. Certificates provide a practical realisation of public key cryptography. The were first introduced in the work of Kohnfelder [Koh78]. The X.509 certificates are the certificates used in the PKI applications we have seen so far. The got their name from the X.509 specification [IT97]. The specification mostly used for purposes of internet communication today is the [HPFS02]. It specifies the format as well as the processing of certificates. In Figure 3.1 we see an X.509 certificate.

An X.509 certificate has three parts. The TBSCertificate (To Be Signed

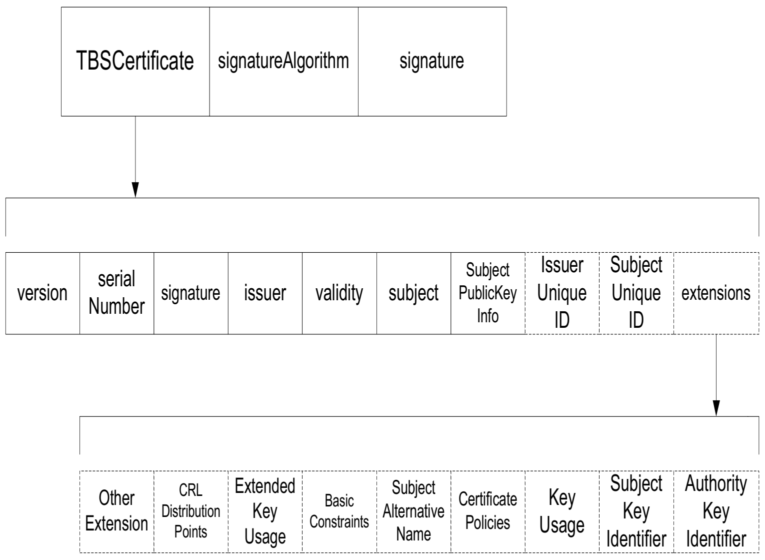

Figure 3.1: An X.509 Certificate

Certificate), the algorithm that is used to sign this part, and the signature over the TBSCertificate. Thus, all information inside the TBS part is protected against changes and it can be authenticated. The CA signs this part.

Inside the TBS part of the certificate the name of the owning entity can be found as well as its public key. Further information are related to the validity period of the certificate, or its serial number, a unique number among other certificates issued from the same CA. The name of the CA can also be found. In order to address other needs, a certificate may contain extensions. One extension for example denotes whether this certificate belongs to a CA or not. Another extension shows which is the usage of this certificate. More extensions exist.

A certificate is used for verifying signatures calculated with the private key corresponding to the public key contained in the certificate. The signature on the certificate itself must be verified, too, as well as the signature of the certificate that is used for verifying this signature. This process continues until a trusted (usually self-signed) certificate is found. The signature on this certificate can be verified with the public key contained in the certificate itself. Another step in this verification process is to examine whether this certificate is revoked or not. This is often done be examining the certificate revocation lists (CRL). In case of encryption the same verification procedures must be performed. In this case the public key is extracted from the certificate in order to encrypt data for the certificate's owner.

3.3.2 Certificate Revocation Lists

Certificates have limited validity in time. But in some cases it is desired to end the validity of a certificate before its expected one. One reason for that could be a change in the affiliation of the certificate's owner or a supposed compromise of the private key. Exactly this action, ending a certificate's validity before its expiration, is the revocation. This revocation information must be known to every PKI participant. For addressing this problem the certificate revocation lists have been introduced. They are also contained in the X.509 specification.

X.509 CRLs are signed documents containing a list of revoked certificates. They resemble the certificate structure with a TBS field, the signature algorithm, and the signature. They contain fields like the number of the CRL, the issuer, the time of issuance, or the time when a newer CRL will be available. The revoked certificates are represented in the CRL by their serial number. Other information regarding revoked certificates like the revocation time or the reason for the revocation may be present. Figure 3.2 shows an X.509 CRL.

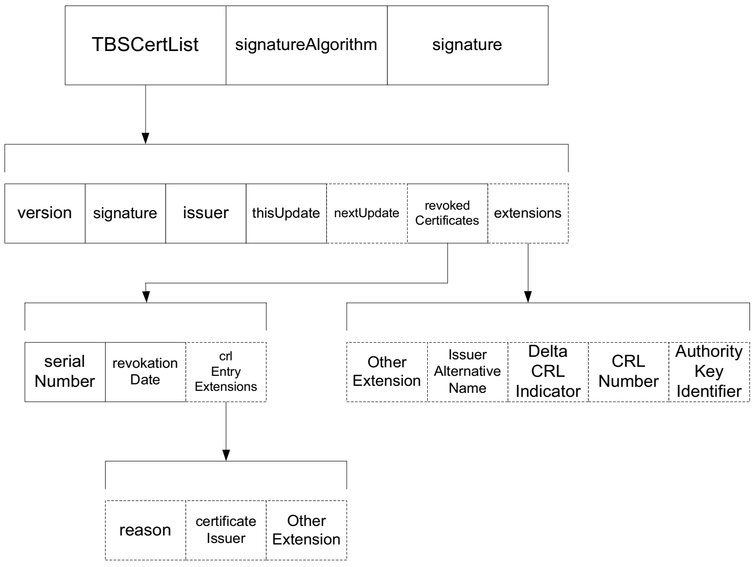

Figure 3.2: An X.509 Certificate Revocation List

3.3.3 Attribute Certificates

Attribute certificates (AC) in comparison to X.509 certificates are rather used for authorisation than authentication. Although authorisation based on X.509 certificates is possible, it may sometimes be difficult to achieve. See [FH02] for reasoning on this. This document is also the specification of attribute certificates.

Infrastructures based on attribute certificates are called Privilege Management Infrastructures. For the design of such an infrastructure see [BB04]. The authors discuss various designs for a PMI and they show the differences between a PKI and a PMI. An overview of the X.509 PMI can be found in [Cha03].

3.3.4 Personal Security Environments

Personal security environment (PSE) denotes a device which contains private keys and certificates. The use of a PSE is mandatory for an end-entity in the PKI. The private key that corresponds to the public key contained in the certificate is found in such a device.

Special countermeasures and security properties must be applied to the PSEs. The secrets contained inside must be protected for integrity and confidentiality. That is, it is not possible to change or read the private key. The PSEs can be found in two forms. Either in hardware allowing a more secure environment or in software that allows more flexibility and platform independency.

Hardware-Based Tokens

The most known hardware based PSEs are the smart cards. For a brief introduction on cryptographic smart card see [NM96]. Smart cards are small portable cards in the size of a standard credit card. They have a cryptographic processor (that is able to perform big number arithmetic) and an operating system for the management of the resources on the card.

In security applications they are used as a secure place from which the private key is not allowed to leave. Therefore, all operations that require use of the private key like signature or decryption are performed in the smart card. This is the general principle of hardware based PSEs. They provide a secure storage for the key and they perform cryptographic operations. In many cases the key creation takes place on the card. In the other cases the keys must be securely transferred to the card.

Other hardware based PSEs are the hardware security modules (HSM). These are dedicated stand alone hardware devices. They can be in the form of a PCI card. They are used for creating and storing private keys as well as for performing cryptographic operations. In comparison to smart cards they are not portable but they are faster.

Software-Based Tokens

PSEs based on software are considered less secure than the ones based on hardware. However, they are more portable, they allow an easy backup and they are more platform independent. This means that they can be easily used in various operating systems. They do not have some of the drawbacks met in smart cards. These are the need for special hardware (smart cards can be operated only with a special card reader) or the presence of complicated interfaces that resolve the communication with the smart card (like PKCS#11 [RSA04], PC/SC [Wor], or CT-API [DFTT05]).

The most typical software based PSE is the PKCS#12 format. This format allows private keys and secrets to be stored in a secure way on software. For example this could be a file (also called a PKCS#12 file) that contains a key pair and a certificate. This file can be transported on a floppy disc or a USB disc and its format is supported by various clients like email programs and browsers. It is the most common way to store private keys within a PKI that uses software PSEs. It has diverse mechanisms for integrity and confidentiality. Another format for storing private keys is the Java KeyStore. This is a proprietary way to store keys in a format that can be used inside the Java programming language. There are two types. The JKS that is using weak protection algorithms and the JCEKS that is using stronger ones.

3.3.5 Further Reading

For further reading regarding PKIs we refer the reader to [AL99], [HP01], and [Aus01].

3.4 Trust Center

With PKI the whole infrastructure based on public key is denoted. Everything regarding this infrastructure, namely its services, participants, applications and products is referred to as PKI. The whole infrastructure is based on trusting someone for assigning certificates to entities. This part of the infrastructure is called the trust center (TC).

The trust center in a PKI is at a minimum the place where the certification takes place. At this point the binding between the entity's name and public key is done. Trust among PKI participants and this instance must be established. Therefore, the name trust center is derived from this relationship of trust. The trust center as a term is used mostly in Germany. In other countries CA and trust center are used as synonyms. For more on the trust center definition and tasks see [FHK95].

But a trust center is more than the certification authority. It is the registration authority and the public components of the trust center like the directory or the OCSP servers. All these instances are operated from an organisation in a secure manner as part of implementing and enabling a PKI. The certificate management authority is also part of the trust center. It is responsible for the communication of the trust center with the end-entities or the PKI applications. We propose the use of CMA in every trust center since its presence has several advantages. In Chapter 10 we see the deployment of CMA in current trust center installations.

In most cases the trust center is located at one geographical position. This is usually a secure location. It is possible however that distributed local registration authorities (LRAs) exist, since registration is often desired to be distributed.

The necessary level of security for a trust center depends on its purposes. One case is a trust center installed for securing email traffic in an organisation. This trust center can be located in a secure room somewhere inside the organisation where only few people are allowed to enter. This level of security may be in this case acceptable, since the applications planned do not require extra measures to be taken. The CA may be even online, allowing people to receive certificates on demand.

This trust center setup and configuration will definitely be altered for a more secure installation. For example a trust center that issues qualified certificates has more security mechanisms applied to it. We have presented two extreme examples. Most trust center installations lie somewhere in between.

3.4.1 Environment Oriented Installations

Trust center installations do not have similarities. One reason for this is that they have to address different security needs. Another one is organisational issues since different organisations have different procedures. Moreover, the type of the planned applications influences the trust center installation. The scale of the PKI also plays a role. Therefore, different demands and specifications for different trust centers exist.

Usually the security aspects of a trust center significantly affect it. Different requirements may appear, like special signature creation entities, human administration, or even a security evaluation of the software and hardware according to security criteria. Other special demands are related to the registration. Is it distributed? How is the entities' data obtained? Special requirements are also related to the tasks of the CMA. Is an acceptance receipt from the certificate holder required? How is the delivery of personal security environments performed? Many more requirements exist.

These requirements are critical for the deployment of a trust center. This deployment will become more difficult if new adjustments are needed. The development of new features will be time consuming as well as the installation, which in addition is error-prone if many features are newly implemented. As a result the PKI will become expensive. These facts will prohibit wide deployment of PKI services.

Therefore, a flexible trust center software must be adaptable to the environment. Due to this environment dependency the trust center software often needs changes, updates, and new developments. A modular trust center design addresses these requirements easier. It allows its services to be distributed and supports different mechanisms (like security, access control, etc.) to be applied to each component. The CMA has a plugin based design. The plugins are components that can be added, removed, or replaced by other components that provide different functionality. The core functionality remains the same and therefore no new mistakes or security leaks are introduced in the existent software. This is very important in the case of an evaluation of the software. The components that are already evaluated do not need to be re-evaluated and only the newly developed parts go through the evaluation process.

3.4.2 Virtual Hosting

Usually one trust center installation regards exactly one organisation. For example a company that needs certification services installs a trust center for providing these services. Considerations may exist however for the installation of a trust center. These considerations can be for example of financial nature. In this case, this company can outsource its certification services.

A trust center that offers such services hosts the trust center of other organisations. One solution is to install an entirely new software or duplicate an existent one. This requires that this software is written or modified. In addition it probably requires a new hardware infrastructure to be installed for supporting the new software. Moreover, new rooms and personnel may have to be organised, too. This approach is time and resources consuming, error-prone, and difficult to manage and implement. A better approach is the virtual hosting.

In the virtual hosting configuration different trust centers are being hosted in one installation. This means that by using the same rooms, human resources, hardware and software, multiple trust centers and their services are operated in exactly one instance. Therefore, many trust centers are being virtually hosted in one installation.

Virtual hosting can be found inside an organisation, too. For example, in the Technical University of Darmstadt there are two virtually hosted trust centers in one installation. The first virtual host is responsible for certifying the servers of the university. It certifies keys stored in software tokens and entities that are machines. On the contrary, the second host certifies keys found in hardware and entities that are physical persons. More virtual hosts can be installed. For example a third host can provide certification for the physical entities and keys that are stored in software. The virtual hosting configuration therefore offers great flexibility in organising the certificate management tasks in a trust center.

Chapter 4

Certificate Management Authority

We present the certificate management authority (CMA). The CMA is a new PKI component. Tasks related to certificate management like certificate delivery or publication are assigned to this component. We see the reasons for introducing a new authority in the PKI environment. We define the CMA and specify its requirements. These will be used for designing it. The concrete tasks that the CMA performs are presented. The considerations associated to the use of the CMA are discussed.

4.1 Motivation for the CMA

The goal of this Thesis is to design and implement a flexible, efficient, and secure certificate management authority.

The main reason for the presence of CMA are certain operational difficulties and security considerations that appear in a PKI. Particularly since offline CAs are often employed for security reasons, the CMA helps to extract the CA products and to process them further. It also relieves the registration offices from tasks related to certificate management. It offers services to OCSP servers or other instances by organising the necessary information workflow for their certificate stores. It enables realisation of various use cases about certificates and offers solutions in a virtual hosting configuration.

4.1.1 Offline CA

It is often desired that a CA is offline. This is due to security considerations. An offline system is much more difficult to attack than a system which is connected to a network. Even if complex mechanisms are used to prevent such attacks the possibility exists and is greater than in an offline system.

The extraction of the products of an offline CA cannot be automated and manual interaction is needed. Moreover, special rules apply to the visits at the CA. These may include the presence of two or more persons at the same time, only daily visits and other mechanisms, too. Other tasks like directly publishing the certificates on an externally available LDAP directory are impossible. Automatic delivery of the products to the end-entity cannot be performed and this task is somehow assigned to the human administrators. Therefore, if the CA is offline operational difficulties exist.

In order to avoid these problems a CA may have to be online. But this solution contains security risks. This is because an online system is more vulnerable to attacks than an offline system. If the certificate management of an offline CA is easy to achieve, then offline CAs would be easier and more often employed. Therefore a higher security level would be achieved.

All tasks related to certificate management are gathered in the CMA. The CMA is responsible for managing the products of the CA, which is offline. If the CA is online, the CMA can still provide the same services. In the rest of this work we will see ways to extract and transfer the CA products and how to perform a flexible and efficient certificate management.

4.1.2 Decentralized RA

In many PKI environments the registration of the users is not done in a central registration authority but in various registration offices. This is a distributed registration process. Nevertheless, keeping all the services that a PKI is providing distributed could lead to a PKI which is very difficult to administer and use. An example for this is the case of every registration office publishing the certificates to each own internet directory. PKI clients would have difficulties in locating certificates. Many tasks would be repeated for every directory like the publishing of a CRL. Many administrators would have to be trained for performing these tasks. This publishing process would be more error-prone than an automated technical solution. Every office would have to operate and administer its own directory. But more problems and operational difficulties are associated to such a solution.

The CMA offers a technical approach to these problems. All certificate management tasks, or at least most of them, that must be performed by the different registration authorities can be assigned to the CMA. Therefore, lots of problems regarding certificate management that appear due to distributed or local registration authorities, can be solved by the presence of a CMA.

4.1.3 Revocation and Certificate Status Information

The most common mechanism to provide certificate status information is the OCSP [MAM+99]. Others may appear or exist already. These mechanisms require a backend from which they can retrieve all information they need for their correct functioning. The CMA is used for organising the information flow for this backend and is responsible for maintaining it. Such maintenance functions are updates or the synchronisation with other resources.

OCSP servers require information about all certificates that a CA is issuing. It is important that every newly issued certificate becomes known to the OCSP server immediately. A persistent data store is required for this task. This can be a database, an LDAP directory, or some other mechanism. The CMA is responsible for maintaining this backend. This is a difficult task for a CA which is offline. In this case an administrator must extract the PKI products and update the OCSP server. But there are lots of security risks. For example the administrator may never publish the certificate to the backend. This process has also scalability problems. As the number of certificates increases the task of an administrator becomes more difficult. Even if the CA is online, an automatic update of the backend store may still be impossible due to a restricted network configuration. CRLs or other revocation information are also maintained from the CMA to the backend. In certain environments, it may be desired that the newest revocation information reaches certain clients as soon as this is issued. For the same reasons as in the OCSP case, this task is delegated to the CMA.

The revocation authority also operates with the presence of a similar backend. This is an authority that issues CRLs on behalf of a CA. This authority needs an update mechanism for the newly issued certificates from the CA that it is working for, in order to accept revocations for them. The backend that this authority uses could be the same as that of the OCSP. The CMA would also control the information flow in this case. Therefore the same information can be used for the proper functioning of other trust center components. The CMA could also be used as a revocation authority itself.

4.1.4 Virtual Hosting

Several problems exist in the virtual hosting configuration. In this trust center configuration, one CA installation hosts various virtual CAs. Those different CAs may have various policies, processes, and workflows. For example, one virtual CA produces software based tokens (like PKCS#12) only, while another virtual CA produces hardware based tokens. These products have different management procedures. The software based tokens may be sent with an email, something that cannot be done in the case of smart cards. All those processes and different requirements can be mapped to certain procedures in the CMA. The CMA can serve a big number of those virtual CAs and realise their certificate management workflow.

A mechanism to distinguish among the products of different virtual CAs is needed. By this, the product of each CA will be processed as this is scheduled for this CA. In the virtual hosting configuration such a mechanism is required.

The CMA removes this task from an administrator and processes the products according to the needs of the correct CA. If this would be resolved manually many mistakes would be done. While the number of virtual CAs increases, this task is becoming even more difficult. The administrator must ensure that a product is processed as desired. Also some of the remaining tasks will be the same among the different virtual CAs. Therefore every CA will have to perform the same task. Alternatively these tasks could be performed only once by the CMA.

4.1.5 Realisation of Business Logic

After the creation of a PKI product by a CA, there are lots of different tasks remaining for this product. The way these are performed may vary. For example delivery of the certificates can be done per email, downloaded from an LDAP server, an HTTP or an FTP server, or even just be picked up from the registration office. The CMA is responsible for realising all these mechanisms. Different trust center installations require different tasks in order to realise the intended applications with the certificates. For example the certificate may have to be published encrypted to a directory in order to achieve proof-of-possession as this is described in [KLW04a]. In other cases it may have to be published only after an activation.

Possible system updates (for example to realise a new mechanism) will affect only the CMA and not the CA. Such updates will happen, because the above mentioned tasks are subject to changes. The reason for this is that the technologies change or new protocols and mechanisms appear. In cases where the CA system is evaluated a re-evaluation will be avoided. A modular and configurable design at the side of the CMA is required in order to meet the special needs of every environment.

4.2 Certificate Management Authority

A lot of tasks must be performed as soon as a certificate or a CRL is issued by the CA. It is the duty of the CMA to encapsulate all these tasks and provide a technical and reliable solution. This solution must adapt to new requirements and changes, since the tasks related to certificate management are also subject to changes.

4.2.1 CMA Definition

In this subsection we give the definitions of the components we are using in our PKI design. The components are the issuer and the CMA.

Definition The *Issuer* is a system that issues certificates, revocation lists, and personal security environments and creates passwords and PINs.

Usually the issuer is a certification authority (CA). Its basic task is to sign certificates. The private key of this issuer is kept secret and very well protected. In addition it is not possible to use it without proper authorisation. This is achieved through many technical mechanisms. One of them for example is that the issuer is completely offline. Another possibility is that the use of the private key requires dual-control. With dual-control we mean that in order to activate the signing key, two human operators are needed. Human processes and policies are perhaps required, too, for protecting this key. For example a special procedure may has to be held for the presence of the administrators in the room where the CA is located.

Other issuers may exist as well. One is for example the revocation authority, an authority employed with the task of issuing CRLs. The revocation authority may be online in order to provide the best possible freshness of revocation information.

In every PKI environment an issuer is present. In such an environment certificates and CRLs are produced and therefore at least one CA must be present. This CA is the issuer. Through use of its private key it certifies PKI products by signing them. For accomplishing this task high security is required and lots of technical considerations are taken into account. Apart from this, it can become a very time consuming task as the number of requests increases. The tasks that follow the signing of the products, are delegated to the CMA.

Definition The *certificate management authority* or CMA is a PKI operating component assigned with the task of managing and administrating products of issuers on their behalf.

The tasks that are associated with certificate management are assigned to the CMA. The services that this component offers are performed securely only by this component. The CMA is an authority. An authority is an instance that has certain rights to perform tasks which are accepted by other entities only if they are performed by this instance. Moreover, the delegation of tasks of the issuer to the CMA, make the CMA also an authority. This is because the issuer is an authority itself since it issues certificates, CRLs, and other products. The CMA is dealing with certificates, CRLs, smart cards, software tokens, PIN letters, and other cryptographic related products.

This definition of the CMA is quite general. In the following we will give the requirements and tasks of the CMA.

4.2.2 Requirements

Security

The CMA is a PKI component and part of a security infrastructure. Therefore, security is an important factor for its design and implementation. Security has several aspects in the case of the CMA. One is the transfer of the products from the issuers. For transferring the products from the CA system one or more administrators are required, if we assume that the CA is offline. In the online case this may be automated. The products of the CA are placed inside a container. The data in this container are secured to remain authentic, unaltered, and secret. Therefore input to the CMA are messages with a special syntax. These messages are used for communication purposes inside a trust center.

Another security aspect is the functioning of the component itself. This means that the design and installation of the CMA are secure. This includes the use of secure protocols. When LDAP or HTTP are used as protocols, the more secure variants, namely LDAPS and HTTPS are deployed. Moreover techniques like dual-control are employed. In this technique, two administrators are needed in order to operate the component. We discuss the security requirements of the CMA in more details in Chapter 9.

Virtual Hosting

The CMA is able to work for many issuers. This requires only one CMA installation with the ability for hosting different issuers. The component is able to deliver its services to many CAs without requiring new installation. It is in position to meet the special needs of the issuers. The tasks that the CMA has to fulfil differ among the various virtual hosts. The CMA is able to realise their requirements. The whole management of the products from the CA system is undertaken by the CMA. The CMA is pre-configured to accomplish certain tasks for every virtual host.

Configurability

The CMA does not make any assumptions on the kind of services it offers. It is able to be configured to perform certificate management tasks of an issuer, in a given time. A minimum set of tasks is defined for the CMA. After that, every issuer decides which of these tasks are important for its functioning, how they must be performed and at which time. An example of this is the following. One issuer issues certificates and corresponding PKCS#12 files. It requires that the certificate and the PKCS#12 file must be sent with an email to the end-user. Afterwards the certificate must be published on an LDAP directory. Another issuer which is working with hardware based tokens (like smart cards) requires that the certificate will be published on the directory only after the user has accepted the certificate and smart card. Although the core functions are the same (LDAP Publishing), the way and time this is done differs among issuers. For achieving great flexibility the CMA is configurable for performing these tasks.

Flexibility

CMA offers a big variety of core functionalities that are important in certificate management. In Section 4.2.3 we will see which are its core tasks. Tasks that are related to certificate management fall into the CMA's area of responsibility and therefore the CMA is able to implement them as soon as this is required. For example one mechanism for disseminating PKI information is LDAP. Another mechanism could be the certificate store proposed by Gutmann [Gut00], [Gut06]. For a general discussion on available mechanism see [HP01, Chap. 9] and [AL99, Chap. 11]. While only the first mechanism is implemented and used, if the second one is also needed then it is able to be implemented and integrated into the CMA.

Some of the tasks regarding certificate management are also automated. A flexible mechanism to meet all different requirements as well as the new ones that will occur is present. Flexibility is also needed in the cryptographic algorithms. Some algorithms may become insecure and must be exchanged. Bigger key sizes will also be required. As the state of the art in cryptography advances the CMA is adjustable to these changes. Moreover the CMA is able to be integrated into existent workflows and procedures that are already established in an organisation.

Scalability

The CMA is able to operate for small, medium, and large scaled infrastructures. It is in position to handle different loads and adapt to the load requirements of different environments. Special interest has the case of large-scaled infrastructures. Such infrastructures are built for example in the case of citizen cards.

Availability

Tasks like providing the backend for OCSP services are assigned to the CMA and therefore availability is another requirement. The CMA is always online in order to achieve this. The services of the CMA are always available. In case of failure the CMA is able to continue with the processing of a request. The issuers delegate their certificate management tasks to the CMA which is in position to accomplish them in a correct and standard conform way.

Standards Conformity

The CMA conforms to the PKI standards. This includes its products and its services. Lack of interoperability will hamper the use and acceptance of a PKI. All duties of the CMA use the proposed standards from different standardisation organisations. But this is needed also for the products of the issuers. These conform to specifications like [IT97] and [HPFS02]. In addition the CMA supports the PKCS#7 or CMP messages as its input format. Issuers that export their products to the CMA according to the above mentioned formats can then be supported and therefore the CMA can operate for any issuer.

4.2.3 Tasks

In this subsection we present the concrete tasks that are assigned to the CMA. These are mostly standard tasks that every trust center supports. For an overview of these tasks see general PKI descriptions like [AL99] and [HP01].

Archiving

The first task of the CMA is to archive certificates. For this purpose a persistent store like a database is needed. Upon request these certificates can be retrieved from the CMA without requiring any interaction with the certification instance. In addition, CMA can archive personal security environments. Therefore key backup can be accomplished. If due to security reasons this is not acceptable (CMA is online), the key backup must take place either at the certification instance, or at a backup component, or at the client side. But if it is desired that the backup takes place in the CMA, this function is supported.

Delivery

An important step in the certification process is the delivery of certificates and PSEs to the end-entities. Responsible for this is the CMA. The mechanisms that can be implemented differ among various configurations. This can be done by email, the physical presence of the end-entity, or by placing the certificate in a well known place like a corporate directory.

Publishing

The certificates and CRLs that are issued in a PKI are published in a public directory. This is needed to enable the PKI participants to access public keys (for example for encrypting messages or verifying signatures) and revocation information. The publication is therefore a necessary management step for a certificate or a CRL. The CMA is responsible for realising it.

Certificate publication has several requirements. The publication place is an LDAP directory or an HTTP server. CRLs must be published as soon as possible. Some certificates are not allowed to be publicly available due to restrictions. Special mechanisms are present that distinguish between certificates that are publicly available and those that are not. These two types of certificate will be handled differently. Such restrictions may be found for example in the German Signature Act [Leg01a].

Certificate Status Information Backend

OCSP servers as well as revocation authorities need a persistent store from which they will obtain the information that they need to function properly. Moreover, this information must be available to them as soon as possible. The CMA offers these services in an efficient and secure way. This backend is a database or an LDAP directory. Both can be secured with SSL to provide a high level of security. The CMA can update the backend in an automated way as soon as the products are extracted from the issuers. As the number of requests and the number of issuers in a system increases, so the complexity increases too.

In order to notify the OCSP or the revocation authority for a new certificate or revocation, a notification mechanism is employed. This is done by sending a special request to the OCSP or the revocation authority. Alternatively this can be performed on demand. This means that every time the OCSP server for example is queried for a certificate that it does not have in its cache, it queries the backend for obtaining the new information. But other mechanisms may have to be implemented, too.

Renewal Notification

Certificates and CRLs expire after a certain period of time. If a certificate or a CRL is about to expire, the CMA sends notification messages either to an administrator, the end-entity, or the registration office. This is combined with the archiving task of the CMA. The CMA possesses all information about the certificates and CRLs issued from different issuers. Therefore, the renewal notifications can be placed in the CMA tasks. Furthermore, renewal notifications are totally automated, which leads to a more flexible PKI product management.

CRL Management

Another task for the CMA is to provide CRL and revocation management mechanisms. Apart from notifications about the renewal of the CRL, another task is the notification of the end-entities with newer certificate revocation information. In [MR00] McDaniel and Rubin propose a mechanism called Revocation On Demand (ROD). This mechanism delivers a CRL to the clients as soon as this is issued. This scheme is suited for the corporate environment. Such a mechanism is implemented in the CMA. The CMA also functions as a revocation authority itself. In this case the CMA is an issuer itself and therefore new security requirements occur that its design addresses.

Miscellaneous Tasks

CMA is in the position to perform more tasks than the ones already described. The above tasks define the minimum set of functions that the CMA implements.

But more tasks may have to be assigned to the CMA. For example one task could be an automated backup of the whole directory where the certificates are located. Another task can be to offer some kind of validation services. The implementation and design of CMA must be well prepared to meet future requirement in a flexible way.

4.2.4 Considerations

The presence of the CMA introduces extra complexity into the PKI. The reason for this is that the CMA is a new PKI component that does not exist in current trust center configurations. We analyse this extra complexity and see what are the main considerations that arise from the presence of the CMA. We further see how its design addresses these considerations and how the benefits gained by its presence outweigh the introduced drawbacks.

The CMA is a software component. It can additionally use extra hardware components (like card readers, smart cards, or HSMs). Therefore it needs one or more extra machines on which it can run. This increases the costs for the setup of the PKI. Moreover it increases the complexity for the administration and maintenance of the trust center machines. Therefore it is difficult to create a "thin" trust center setup.

The CMA can be operated on the same machine like the RA. It can also be run on the CA machine if the CA is online. In these two cases the above mentioned problems do not exist. If the CMA runs on an exclusive machine, it reduces the workload from the other machines. In a large-scaled infrastructure this is a significant benefit. In a small-scaled infrastructure it is a minor gain. Therefore the CMA produces extra workload in small-scaled infrastructures and only if it cannot be combined with the other trust centers components.

The CMA also introduces new developments to the other components. For example the CA extracts its products in a format known to the CMA or does not perform certain tasks anymore. The new developments increase the cost for the establishment of a PKI.

The various PKI components are often required to adjust to new developments (for example to realise a new protocol). In the case of the introduction of the CMA the new developments are mostly limited to the communication. Therefore they are not expected to be significant and the cost is kept small. In addition they have to be performed only once.

The new component introduces additional complexity to the trust center operators. They have to be trained for operating the new component. This includes tasks like starting or stopping it, evaluating the logs, transferring input files (in an offline setup for example), or extracting its products.

While the operators need to interact with a new component they do not need to administrate the products of the CA. This is performed by the CMA. The only task of the operators is to transfer the CA products to the CMA, which is programmed to manage them according to the needs of every CA. In a trust center without a CMA, the operators manage the CA products. This requires also special training for the operators and often visits to the CA. Technical difficulties and security risks exist in this case (see also Section 4.1).

The CMA issues new security considerations. Firstly, because it is a new link in the security chain and if it is weak the whole security is endangered. Secondly, because it manages "sensitive" products, namely the certificates and revocation information issued by an issuer. Thirdly, due to the fact that the CMA in a usual setup is online, it is more open to attacks. Lastly, a secure communication channel with the other components is needed.

The presence of the CMA increases however the security in a trust center. This is because it enables the CA to be offline in an easy way (see Section 4.1.1). It is possible to evaluate its security as a technical component. Such an evaluation is found in Chapter 9. The secure communication with the other components is addressed in Chapter 5.

The CMA causes additional communication costs with the other trust center components. This requires a sufficient and secure protocol to be used. Moreover more messages are produced to serve the communication among the components. As the number of requests in a trust center increase so the number of exchanged messages is increasing, too.

The communication of the CMA with the new components is addressed in Chapter 5. The specification of a protocol can be found there. The protocol is scalable and can be used in offline communication. If the components have network connection with each other there is the possibility to automate the communication. If the CMA is not present, the interaction with the PKI end-entities is done by the CA operators. In this case a communication channel between the operators and the end-entities is established. This requires also communication costs. In the case of the CMA, this channel is transparent to the operators.

Chapter 5

Communication in a Trust Center

The CMA is a new PKI component. Its purpose is to manage and administrate the products of issuers which are CAs or other certification or revocation instances. Therefore a means of communication with these CAs is needed. Moreover, other components (like the RA) are sending messages and data to the CMA. In order to address this situation, the communication possibilities of those components with the newly introduced CMA are examined and analysed. This Chapter is addressing this problem, namely how to communicate in an efficient and secure way with the CMA. Parts of it have been presented at the Third International Workshop for Applied PKI (IWAP 2004) [KLW$^+$04b].

5.1 Introduction

Trust center software has to provide a great variety of services each carried out by complex processes like registration, key generation, certification, personalisation of personal security environments, key backup, and many more. Each service has different requirements that are posed by the PKI environment. The requirements may alter when the trust center has to adapt to different situations. A common technique in software engineering to meet the demands of an adaptive software is to break it up into components, each covering certain operational aspects. These components can then be distributed according to the needs of the special environment. Furthermore, certain components can be replaced to reflect changes in the environment. A component based approach to complex processes requires that the components are able to communicate to each other. Therefore a proper mechanism that is able to resolve all communication requirements is needed. Furthermore, this communication is a grand issue to the overall security of the trust center. Moreover, some of the components may be offline (like the CA), a fact that makes the communication even more difficult.

The type of communication among parties without access to the internet or any kind of network communication is called "sneakernet". The sneakernet is met often in operational public key infrastructures. This is because some components are offline. In these cases the communication is done manually by humans with the help of a floppy disc, USB token, CD, or another portable media. The sneakernet communication has the disadvantage that it can not be automated. Such communication requires a format that will describe the messages that are exchanged among the offline and the other components. Moreover, this format must be able to transfer the necessary data. If such a format does not exist, the different components are not able to communicate and the data cannot be described fully. In the case of the offline CA these messages must also be secured while entering or exiting the CA. The security considerations are integrity, authenticity, and confidentiality among others. For specifying such a suitable format a protocol is needed. A protocol that addresses the security requirements, is efficient, compact, and portable.

We propose the Intra Trustcenter Protocol (ITP), a flexible and secure management protocol which is well suited for the communication among arbitrary trust center components. The ITP takes a look into an area which current standards (like PKCS#7, CMP, or XKMS) do not address, namely the communication among arbitrary components within a trust center. We see which are the already existing protocols and discuss their properties. The proposed protocol provides a realistic solution to the sneakernet problem. We list the requirements for intra-trustcenter communication which serve as design criteria for our protocol. Moreover, we motivate the use of XML for specifying the syntax of our protocol, since XML is a well accepted means for specifying data structures. XML comes with sophisticated security mechanisms which we employ in order to guarantee integrity, authenticity, and confidentiality. We can further use these mechanisms to enforce operational security constraints like roles or dual control. Furthermore, the syntax and details of the protocol are given. We present two case studies for the use of ITP derived from applications that can be met in some PKI environments. The protocol is analysed regarding its flexibility and security along with its deployment considerations. We also give an outlook on further developments of the protocol.

5.2 Certificate and Certification Protocols

The existing proposals and standards concerning certificate and certification management can be divided into three groups.

The first group consists of two standards. The "Cryptographic Message Syntax Standard" or PKCS#7 [RSA93] and the "Certification Request Syntax Standard" or PKCS#10 [RSA00]. They were developed by RSA Laboratories and describe message formats in ASN.1 [IT02]. PKCS#10 is only suitable for an end-entity to provide a to be certified public key to a trust center and thus is not general enough for intra-trustcenter communication. PKCS#7 comes in two modes: In the more general one it can be used as a container for transporting arbitrary data in a secure way. This can also be employed within a trust center. This mode does not consider the transported content. All data is placed inside a container and the protocol does not offer any possibility to further express and describe them. For achieving this, internal representations are stored in the container. These representations are mostly proprietary. In the special enrollment mode it is only usable for delivering certificates and certificate revocation lists and thus not general enough.

The second group contains two protocol standards published by the Internet Engineering Task Force (IETF). These are the CMP [AFKM05] and the CMC [MLSW00] specifications. They, in turn, are based on the CRMF [Sch05] and CMS [Hou99] formats which play a similar role as PKCS#10 and PKCS#7, respectively. They are all specified in ASN.1. These protocols focus on scenarios where communication takes place among an end-entity, a certification authority, and optionally a registration authority. Thus, they cannot be employed for the communication among arbitrary components. Gutmann discusses these protocols in [Gut03] and chooses CMP instead of CMC [Gut03, Sec. 3.3] for building a PKI architecture. This paper considers the older version of CMP [AF99]. It identifies however in [Gut03, Sec. 5.2] common problems when CMP is used.

A rather new development is the "XML Key Management Specification" [W3C01] which was proposed by the W3C and arose from the need for a key management protocol for the XML signature [W3C02b] and encryption [W3C02a] standards. It is not appropriate for intra-trustcenter communication since it focuses on the communication between end-entities and trust centers.

Only the CMP and PKCS#7 specifications can be actually considered related work to our protocol, since they can be used for the purpose of communication inside a trust center. However, as we already pointed out there are some considerations for using these protocols inside the trust center and our approach offers an alternative to these two specifications.

5.3 Protocol Criteria

This section focuses on the properties that a communication protocol among trust center components has. The criteria that apply to the protocol itself (how it is) are named, along with a discussion on the motivation behind them.

5.3.1 Design Criteria

We see what are the criteria that apply to a protocol for intra-trustcenter communication. This list is partly based on the design goals of other protocols (e.g [AFKM05], [W3C01]) and partly on what we have encountered while setting up trust centers for different projects.

1. *Generality* The protocol is able to handle all kind of data which may be passed among trust center components. It does not put any restrictions upon the number and type of components the trust center is composed of. The protocol allows to express the data in a structured manner when possible (i.e. not just binary). Signatures will still be binary data.

   As the security and flexibility mechanisms of the protocol apply to all messages, all possible messages are presentable within the protocol. In order to be able to apply various mechanisms, like encryption, to different parts of the message this is well structured.

2. *Extensibility* The protocol allows to be extended by new messages, data types, or components.

   This is necessary to be able to meet future requirements, like new trust center products or special request types.

3. *Independency* The protocol is independent from the structure and workflow of the trust center. This includes the independence of any means of transportation, including transport media, online versus offline transport, security, and protection issues.

   This is to meet all possible connection types for the trust center components and all policies which might be applied.

4. *Automation* The protocol supports the automation of the processes in a trust center.

   The trust center may automate some processes. The protocol does not hinder this. This includes especially the authentication mechanisms.

5. *Scalability* The protocol scales with the size of the public key infrastructure, the security level, and the complexity of the trust center installation.

   Different trust centers differ in complexity, size, and security level. In addition these parameters may change during the lifetime of a trust center. The protocol deals with these facts.

6. *Traceability* The protocol allows the tracing of messages, applications, and products. The messages are human readable where possible. Encrypted parts are binary data.

   As trust centers are highly security sensitive applications there will be much auditing and quality insurance means applied to them. The protocol supports this. Additionally this eases the debugging or the error search.

7. *Security* The protocol messages (or parts of it) are able to be secured (i.e. authentic, unaltered, and secret). Additionally the protocol supports common authorisation techniques (e.g. dual control, delegation). It is possible

to achieve this with any cryptographic algorithm or data structure, regarding both the communicated data and the protection of the protocol messages itself.

As the intra-trustcenter communication is a major issue in the security of the whole trust center the protocol supports all means for protecting this communication. As the field of cryptography develops, the protocol is able to adjust the means of protection to the current state of the art.

5.4 Why to Use XML?

XML is ideal for describing structured data. Applications using XML can define a special syntax in order to describe the data that should be used inside the application. For example a special tag `<X509Certificate>` can describe an X.509 certificate while the tag `<revocationPassword>` describes the revocation password that an end-user would like to use. This flexible data format is perfectly suited for defining the structures of ITP.

This data format is in addition portable. Data described in XML can be exchanged among applications which are XML aware. Almost all programming languages support XML. Introducing document type definition (DTD) or XML schema, the tags and structure of the XML document are constrained to the rules applied from the DTD or schema. This can be used to avoid errors in the XML structure.

XML documents can be read from humans as well as they can be parsed and understood from computers. They are both human and machine readable. This is a significant property of an XML document in cases where humans should examine the correctness of data. For example the subject distinguished name (DN) that should be placed in the certificate can be such data. This can be described in the `<subjectDN>` tag (e.g. `<subjectDN>CN=Alice PKIUser, O=MyOrg, C=DE</subjectDN>`). While a machine can examine whether this tag contains correct data it cannot examine whether the name Alice PKIUser really exists. A human administrator on the contrary can examine the credentials of the entity and decide whether such a value for this tag is valid or not.

XML supports also digital signatures [W3C02b] and encryption [W3C02a]. Moreover it supports different cryptographic mechanisms (like RSA, DSA, or ECDSA [BWKKW05]) providing therefore security related to different mathematical problems (factoring and discrete logarithm problem). Another possibility that the signature framework has to offer is that XML structures can be multiply signed from one, two or more different entities (this can include also physical persons) as well as ability to sign certain portions of the whole structure. For example in the case of dual control the presence of two signatures is required.

Lastly, by using XML the use of ASN.1 structures for exchanging such messages is prevented. Applications can avoid the step of implementing these special and in several cases complicated structures just to serve the communication among trust center components. The description of the ITP messages in XML is quite simple to implement and use.

5.5 The ITP Protocol

ITP consists of XML messages exchanged among components inside the trust center. The syntax of an ITP message is found in Listing 5.1.

```
<itpMessage version="1.1" id="123456789">

    <sender>Registration</sender>

    <recipient>Certification</recipient>

    <application id="12345678">
        <profile id="ProfileId">
            <element1Name type="binary">
                <value id="0">MiXAbc+...</value>
            </element1Name>
            <element2Name>
                <value id="0">12345</value>
            </element2Name>
            <!-- ... more elements -->
        </profile>
        <ds:Signature>
            <!-- signature elements... -->
        </ds:Signature>
        <!-- ... more signatures -->
    </application>

    <!-- ..... more applications -->

    <ds:Signature>
        <!-- signature elements... -->
    </ds:Signature>
    <!-- ..... more signatures -->

</itpMessage>
```

Listing 5.1: Syntax of an ITP message

5.5.1 Application

The most significant element is an `<application>` which encapsulates all data that is exchanged among the various components. An `<application>` represents a request for a trust center component to deliver a certain service.[1] Every component can form an `<application>` and send it to any other component. The basic mechanism is the following: In order to execute a certain task, one or more structures of type `<application>` are generated. Each `<application>` is routed from component to component. Each component performs its tasks as these are defined by the type of `<application>`. Afterwards this component creates a new `<application>` and sends it to the next component. This new `<application>` is based on the older one by adding, changing, or removing parts of it. Thus, an `<application>` contains the pieces of data needed to accomplish a task as well as it determines how this data are processed. The syntax of an `<application>` is found in Listing 5.1.

An `<application>` mandatorily has the attribute `id` which uniquely identifies it. If the same application travels among different components this `id` always remains unchanged. By this traceability of applications can be achieved. Furthermore, replay of already processed applications can be prevented. This feature is important since a trust center may have to guarantee that certain services are carried out at most once per request (e.g. a request for just one certificate should under no circumstances lead to more than one certificate being issued).

The tag `<profile>` determines by its unique attribute `id` the kind of application. The most basic ones concern issuing certificates and processing revocations, but others may be added as well. It is also possible to provide `<profile>`-ids for proprietary communication purposes inside the trust center. This could be done for example for communicating meta-information within the trust center.

The `<profile>` also defines, how the applications are processed and which components are involved at which stage of procession. An example for this is whether a key-pair will be created in the trust center or it will be provided by the end-entity. In the first case it defines whether some kind of key backup should be performed. In the second case it defines the necessary proof-of-possession data. Every component interprets the `<profile>`-ids and performs its services according to them. This mechanism enables ITP to transport arbitrary requests.

Within the `<profile>`-tag the pieces of data are included. Most prominent among them are certificates and CRLs. Apart from this other data can be met like passwords, end-entity data, or trust center functional data. For example this could be the last and first name of a person for an application to the registration component or the subject DN for an application to the certification component. In any case a way to express this data is needed. This is a special XML tag that

[1] We have chosen the name `<application>` instead of `<request>` since every request is actually a form (like an application form). From this form every component extracts information that it needs and fills the form with the information it produces.

describes each of them. The usual data that is exchanged among the components possesses its own tag. It is possible however to define new tags that represent data which is needed for an application to be complete.

Every tag has an attribute called `type`. This attribute denotes the type of the tag. There are two possible values at the moment for this attribute: binary for binary objects like certificates or CRLs and boolean for objects that get values true or false. Such an object is a field describing whether a certificate should be published or not. If the attribute is missing the expected type is a string of letters. Inside the tag the subordinate tag `value` is contained. This holds the value of the encapsulating tag. It has an `id` attribute to uniquely identify the different values that the encapsulating tag can get. For example, if two certificates are contained in the application, then the certificate tag contains two `value` tags. The first one with the `id` set to '0' and the second one with the `id` set to '1'.

5.5.2 Application Signature

An `<application>` may contain one or more elements `<ds:Signature>` of type enveloped signature according to the XML signature standard as described in [W3C02b]). These signatures are calculated over the `<application>` by the last component which altered its content. By this, the changes are authorised. The destination component which imports the application may verify the signature of the application. In addition, it is possible for components to sign certain portions of an `<application>`. XML signatures allow signing of a given path inside an XML structure. This gives a component the ability to sign only the data it creates and actually is responsible for. In addition there is the possibility to calculate and append more that one signature over an `<application>`. This `<application>` can travel among the trust center components, which they verify the signature only over data that they need to operate. Furthermore dual control can be enabled with two participants (trust center administrators for example) signing the XML structure and appending afterwards both signatures at the `<application>`.

We see that with the use of XML signatures, as well as the ability to define a new `<application>` and express all possible data inside a trust center we achieve two goals of ITP. These are security and flexibility.

5.5.3 ItpMessage

The element that is root of every ITP message is the `<itpMessage>`. Two attributes are contained in an `<itpMessage>` element. The first one is the `version` attribute to denote the version of ITP in use. Current version is "1.1". The second one is the `id` attribute. The `id` attribute holds a number which is unique for each message. Therefore each component can recognise whether it has already processed this message or not. In addition an `<itpMessage>` must contain the `<sender>` and `<recipient>` elements.

The `<sender>` element is used for describing the component that sends the `<application>`. This can be an IP address or even a symbolic name known inside the trust center. Name resolving based on an IP address is for example useful in an environment in which the trust center components operate on different hosts. If this is based on the symbolic name, it can be used in cases in which the components are found in the same host. The same rules apply to the `<recipient>` element. This is used to hold information about the component to which the ITP message is destined for.

Another possibility is to use the DN of the certificate assigned to every entity if all components possess such a certificate. This is very likely to happen since the various components may have to sign an application. The DN is used in this case only for naming purposes and not for locating the correct certificate for verifying signatures. Therefore, including the serial number of the certificate is not necessary. The information for locating a certificate is contained in the XML signature fields. This overcomes the CMP problem of pure DN-based identification described in [Gut03, Sec. 5.2]. The problem in this case is that the correct certificate cannot be located if a DN is assigned to more than one certificate (the same entity holding multiple keys). This problem does not appear in the XML signatures employed in ITP since they contain information that help verifying a signature, like the certificate itself.

Inside the trust center a peer-to-peer communication may exist, where every component communicates with any other component. For example, the CA should be able to receive messages from different RAs, send messages to a backup component as well as to the CMA. We see that this protocol is flexible in the addressing of arbitrary peers and the problem of defining the source and the destination of the messages is solved.

An `<itpMessage>` is a container for `<application>` elements. Inside an `<itpMessage>` one or more `<application>` elements are contained. For example the registration component sends three certification requests of one user at the same time. One can be a request for an encryption key-pair, the second for a signature, and the third for a non-repudiation. In this case it could envelope three `<application>` elements inside the `<itpMessage>`. A new `<application>` (see also Subsection 5.6.1) can be defined, which represents exactly this use case and therefore only one `<application>` is needed, demonstrating the flexibility of the proposed scheme. A complete ITP message can be seen in Listing 5.1.

An `<itpMessage>` can also be secured with an XML signature. The same properties as in the signing of an `<application>` element apply in this case.

5.6 Case Studies

In order to demonstrate the use of the protocol we construct two scenarios. Both scenarios are taken from applications that are met in a PKI environment.

5.6.1 A Typical Application

This scenario consists of a Registration and a Certification component (which is offline) as well as the CMA. All components have multi-client capability and they can operate for different virtually hosted CAs. In addition, two trust center administrators (operators) must sign all incoming requests to the offline Certification component.

Alice, who belongs to organisation A, wants to get three certificates. One that can be used for encryption, one for digital signature, and one for non-repudiation. In addition she provides a revocation password to be used in case of revocation. She also wants her certificates to be publicly available.

Alice makes a certification request to the Registration component. This component examines the credentials of Alice. It checks that they are valid and sends a message to the Certification component for the special certification request (known to the system with the id “MultiCert”). This message contains the subject DN for the certificate, the virtual CA that should sign the certificates, the hash value of the revocation password, information whether the certificates should be published or not and the e-mail address of Alice. It then writes the message that has one application and signs the application to provide authenticity. Lastly, two trust center operators sing the message too, in order to be accepted from the Certification component. The message can be seen in Listing 5.2. This message is written on a portable media and is manually transported to the Certification component (since it is offline) by the two operators.

```
<itpMessage version="1.1" id="20040202164445">
  <sender>Registration</sender>
  <recipient>Certification</recipient>
  <application id="20040202164832">
   <profile id="MultiCert">
    <clientName>
     <value id="0">Host A</value>
    </clientName>
    <subjectDN>
     <value id="0">
       CN=Alice,OU=OrgUnitName,O=OrgName,C=DE
     </value>
    </subjectDN>
    <revocationPassword type="binary">
     <value id="0">7c4a8 ... 8941c</value>
    </revocationPassword>
    <email>
     <value id="0">alice@orgunitname.orgname.de</value>
    </email>
    <publiclyAvailable type="boolean">
     <value id="0">true</value>
    </publiclyAvailable>
   </profile>
   <ds:Signature>
     <!-- signature of the Registration -->
   </ds:Signature>
   <ds:Signature>
     <!-- signature of the first operator -->
   </ds:Signature>
   <ds:Signature>
     <!-- signature of the second operator -->
   </ds:Signature>
  </application>
</itpMessage>
```

Listing 5.2: Registration to Certification message

The Certification component imports the message, verifies the signatures of the operators over the application and logs their identities with the subject DN of each operator certificate.[2] It then verifies the signature of the Registration component, creates all three certificates (signed with the key of the appropriate virtual CA) and changes and adds information to the application into a new message to the CMA. It also signs the application to provide authenticity and integrity of the data. The key used for signing should not be the same as the one used for signing the certificates. In addition, the signatures in the messages exchanged among components are operational signatures without any special properties. The message (seen in Listing 5.3) is transferred to the CMA.

```
<itpMessage version="1.1" id="20040202170134">
  <sender>Certification</sender>
  <recipient>CMA</recipient>
  <application id="20040202164832">
   <profile id="MultiCert">
    <clientName>
     <value id="0">Host A</value>
    </clientName>
    <userCertificate type="binary">
     <value id="0">
       Base64 encoded certificate
     </value>
```

[2]Such a requirement may appear when it comes to evaluation of the Certification component.

```
      <value id="1">
        Base64 encoded certificate
      </value>
      <value id="2">
        Base64 encoded certificate
      </value>
     </userCertificate>
     <revocationPassword type="binary">
      <value id="0">7c4a8 ... 8941c</value>
     </revocationPassword>
     <email>
      <value id="0">alice@orgunitname.orgname.de</value>
     </email>
     <publiclyAvailable type="boolean">
      <value id="0">true</value>
     </publiclyAvailable>
    </profile>
    <ds:Signature>
      <!-- signature of the Certification -->
    </ds:Signature>
  </application>
</itpMessage>
```

Listing 5.3: Certification to CMA message

Lastly, the CMA verifies the signature. Afterwards it examines whether the certificates should be published or not (specified in the `<publiclyAvailable>` tag), it publishes them and sends Alice a notification by e-mail with the certificates attached.

In this scenario two messages were created. One from Registration to Certification and one from Certification to CMA. Both refer to the same application. Therefore the `id` of the application must remain the same. Another remark is that the `<subjectDN>` field of the first message is missing in the second message. This field is of great importance for the Certification component, since if this field is missing it cannot issue the certificates, but of no importance for the CMA. In addition the subject's DN is now contained in the certificates.

5.6.2 A Case Study for Qualified Certificates

We present a special application of the ITP in the case of the German Signature Act [Leg01a]. According to §3.1 of the corresponding Ordinance [Leg01b] a qualified certificate may be published on a directory only if its user has accepted it. With directory a mechanism to have the certificates verifiable and available is denoted. This can be done for example with an OCSP server [MAM$^+$99] and an LDAP server [HM02] respectively. The certificate may optionally be available, meaning that the users decide whether their certificates will be public or not. The users must then notify the trust center whether they want their certificates to be published or not. For increasing the level of security an activation password may have to be provided from the users in order to activate their certificates.

Therefore two requirements exist. The first one is that the certificate cannot be automatically published. This can be done only after its user has accepted it. The second one is that this certificate may not be available, namely it cannot be downloaded. In order to implement these special requirements the ITP can be employed. An ITP message can be sent to the CMA component containing information about the certificate that must be published, whether it must become known only to the OCSP or also to the LDAP directory, as well as an activation password. An example message can be seen in Listing 5.4.

```
<itpMessage version="1.1" id="20060809174556">
  <sender>Admin</sender>
  <recipient>CMA</recipient>
  <application id="20050303174542">
   <profile id="Activation">
    <clientName>
     <value id="0">Host A</value>
    </clientName>
    <certificateSerialNumber>
     <value id="0">10245</value>
    </certificateSerialNumber>
    <issuerDN>
     <value id="0">
       CN=MyCA,OU=OrgUnitName,O=OrgName,C=DE
     </value>
    </issuerDN>
    <activationPassword type="binary">
     <value id="0">7c4a8 ... 8941</value>
    </activationPassword>
    <publiclyAvailable type="boolean">
     <value id="0">true</value>
    </publiclyAvailable>
   </profile>
   <ds:Signature>
     <!-- signature of the first operator -->
   </ds:Signature>
   <ds:Signature>
     <!-- signature of the second operator -->
   </ds:Signature>
  </application>
</itpMessage>
```

Listing 5.4: Activation message to CMA

ITP is suited in this case study since it allows one trust center entity (the administrators) to send a message to another entity (the CMA). In addition all important information are contained inside an ITP message. Moreover, the request can be examined from the administrator for correctness. Since the data is human readable, the administrator can check whether the certificate about to be published is the correct one or not. One mechanism to realise this use case is having the administrators publish the certificate on the LDAP server themselves. But this action is error-prone and may violate security requirements. Other mechanisms to notify the CMA can be implemented. This requires their design and implementation. ITP offers the possibility to directly send messages to the CMA, it is already established, and it can be reused by different trust centers.

5.7 Analysis of ITP

ITP aims at a secure and flexible protocol for the communication among trust center components. The mechanisms of ITP meet these two goals.

5.7.1 Flexibility

ITP messages can travel among all trust center components. The sender and recipient tags of a message solve the addressing problem. Every component can set these tags according to the desired flow of information. In addition, the number of the operating components can vary without affecting the protocol itself.

All kind of data inside the trust center can be transported with ITP. Both binary data (like the X.509 certificate) and simple text can be sent with an ITP message.

There is the possibility to define new tags and applications. Therefore, ITP is able to meet new requirements in the trust center communication. This may occur in cases where the communicated data or the services that the trust center offers change.

ITP messages are XML messages and thus can be sent with various mechanisms. These may include HTTP(S), TCP/IP, email, or a floppy disc. In addition, the messages themselves are portable among applications since they are described in XML. Moreover they can be read by humans as well as machines.

The messages and applications can be tracked. Every message and application has a unique id. This enables the components or the administrators to keep a history of past messages or applications. In addition, the flow of information inside the trust center can be tracked. This is especially useful in cases where auditing or special logging is required.

5.7.2 Security

ITP offers integrity, authenticity, and confidentiality to the communication of the trust center's components. It further supports authorisation mechanisms.

Parts of the messages can be encrypted for providing confidentiality. Candidates for such an action may be values like the private key of an end-entity, the revocation or activation password, or personal data of the end-users.

It is possible to calculate a signature over a message, an application, or certain data inside an application. Therefore authenticity as well as integrity for these values is provided.

ITP messages may be signed from one or more components or administrators. This enables dual or multi control. This kind of control is met very often in high security trust centers.

5.7.3 Deploying ITP

ITP, as described here, can be used by trust centers which seek a method for arranging their information flow inside the trust center. This is independent of the number and nature of components and services as well as the kind of information communicated among them. This is a significant step, since the communication among arbitrary trust center components with the requirements mentioned in Section 5.3 is difficult to achieve and existent solutions do not provide this functionality (instead of, most of them are standardised). At the moment ITP can be used only for intra-trustcenter specific implementations. A newer version of ITP is needed that specifies the ITP messages, in order for them to describe most of the tasks inside a trust center in a standardised way that will enable interoperability of different implementations. In Chapter 8 we see that ITP is used for communicating messages between a CA and the CMA. Moreover, we see its use for exchanging messages needed in the CMA workflow. The ITP messages will provide the necessary communication means among the so called certificate management plugins also presented in Chapter 8.

5.8 Future Work

We proposed ITP, an XML-based protocol for intra-trustcenter communication. We introduced the design criteria of the protocol, argued on their usefulness and reasoned that these criteria have been met. We specified the 1.1 version of ITP along with examples extracted from real scenarios.

The ITP protocol can still be used as a proprietary solution. The next goal is to design and implement ITP version 2.0, a fully-fledged protocol with extended functionality and strictly defined rules and structure (work towards a standardised version). The use of criticality flags for the various tags of the protocol messages should be investigated. Towards this direction considerations on whether all messages should default to non-critical or which actions should be taken if a criticality rule is violated should be made. A mechanism for error handling should also be integrated in the protocol with clearly defined exceptions (like signature verification failed or application has already been processed) and exception identifiers. In addition a more advanced signing policy mechanism can be integrated. With this mechanism it can be regulated whether components are allowed to change data from the initial application and therefore destroy the original signature and its validity. Special care should be taken on examining the trade-offs between simplicity and extended functionality.

Chapter 6

Directory Services

Basic task of the CMA and certificate management is the publication of certificates and revocation lists. This is needed for supporting the PKI users to locate certificates for encrypting messages or verifying signatures. Certificate publishing is a difficult task because there are many possible realisations for different installations. Moreover a successful planning for directory services provides a successful support for PKI tasks like certificate searching or revocation information retrieval. This Chapter discusses the possible solutions for effective and flexible certificate management as far as publishing concerning. Parts of this Chapter have been published in [KLW05].

We see what the purpose of directory services in a PKI is. We discuss the possible solutions for supporting such a service. We further concentrate on the LDAP based services and see how they can be used to successfully publish PKI information. We provide a guide for public key infrastructure designers and administrators when planning for directory services with LDAP. We analyse the available mechanisms and propose a best practice guide for their use in PKI. We then take a look into the German Signature Act and Ordinance and examine the parts related to directory services. Finally, we translate those to the LDAP directories practices.

6.1 What Are Directory Services Good for?

In a PKI there is the need that public information like the certificates or the CRLs are publicly available. For example, in order to verify a signature on a document the certificate of the signature originator is needed. The certificate of the CA that has issued the user certificate is also needed. If these are publicly available then their retrieval is easy. In the encryption case, especially when the certificates have not been exchanged in previous communication, the target certificate must be located and downloaded in order to encrypt a message.

Apart from the certificates also revocation information is needed for checking whether a certificate has been revoked or not. In contrast to certificates that may have been exchanged between communication partners, CRLs are issued by the CA and therefore it is difficult that all users will get one CRL as soon as this is ready.[1] In addition, certificates are longer valid than CRLs. A typical validity period for a certificate is one year and for a CRL several hours.

The services for making the PKI products publicly available is often called directory services. This is because a kind of directory is being used to store such information. The interface to this directory is known to various clients and thus, they can locate and download the information they need. The most typical solution for providing such directory services in a PKI is based on the LDAP protocol [HM02]. Other solutions exist, too.

6.1.1 Possible Solutions

There are different solutions for supporting the directory services in a PKI. We briefly see their properties as well as the advantages and disadvantages of each approach. One solution for providing directory services is by employing an X.500 directory.

X.500 Directories

These directories derive from the X.500 specification [IT01]. It is a client-server protocol with which the clients can access information stored on the server. This information is arranged hierarchically. X.500 requires a full OSI protocol stack and therefore the more lightweight version of it, called LDAP, was specified. LDAP runs over TCP/IP. Today, LDAP is being commonly used, especially on the client side. On the server side X.500 based directories may exist with an LDAP front end (gateway).

X.500 has a very good support for publishing X.509 certificates and CRLs. Actually, these directories and the X.509 certificates are related to each other since the very first application planned with the certificates was the secure access to such directories. It allows different mechanisms like special certificate matching rules or returning of matched values. These mechanisms were missing in LDAP. However, newer specifications allow such possibilities in LDAP. But this results into a more complex LDAP protocol. For a study on considerations about using LDAP for supporting PKI and differences to X.500 see [Cha03].

[1]In some cases however this may be desired. In [MR00] the authors propose the revocation on demand. This is practically a CRL push service. In this scheme a CRL is sent to a client as soon as it is issued. In [FFW99, Chap. 3] the same scheme is discussed as a solution to the revocation management and distribution problem.

DNS

The storing of certificates and revocation information in the DNS is specified in [Jos06]. The basic idea of this scheme is to store the certificate and CRL in a special resource record called CERT. This scheme supports also PGP and SPKI[2] certificates.

A further discussion on the DNS certificate publishing is given in [Jos01]. Moreover, it contains a performance analysis of the protocols as well as the difference between them. It further discusses security considerations for DNS.

Web Servers, HTTP, FTP

Web servers can also be used for disseminating PKI information. The certificates and the CRLs are placed on the web server and then they are accessed over HTTP. The certificate or CRL is saved in this case as a file, preferably with the ".cer" and ".crl" extension respectively, and placed somewhere on the server. After that it can be downloaded from any client with such capabilities. In this case a very usable feature is that typical browsers can install the certificates and CRLs in their internal store by visiting the corresponding link.

The same principle applies to the FTP transfer. In this case the PKI information is placed on an FTP server. The same naming is proposed. Both HTTP and FTP ways to transfer certificates are specified in [HH99]. This document also describes the MIME types for certificates and CRLs which are "application/pkix-cert" and "application/pkix-crl", respectively.

But this solution does not support the search for certificates. It is just being used for transferring the PKI information. Gutmann proposes a relational database as certificate store in [Gut00]. This would allow more dynamic access of information as well as search functions. This work has also been specified as an RFC in [Gut06]. It provides a more general approach for searching and fetching certificates over HTTP. This approach enables the searching and retrieving of certificates (X.509 and PGP) and CRLs.

Evaluation

X.500 in contrast to LDAP is heavyweight. LDAP is more easily deployed both at the server and the client side. This is because LDAP simplifies many of the X.500 features. Although this causes LDAP to lose some functionality, new developments in the LDAP protocol preserve the features of X.500 that support PKI. Therefore LDAP is preferred than X.500.

LDAP allows more features than DNS. These are for example support for more security protocols, for more complicated searches, or for granular access control [Jos01].

[2]Very few infrastructure deploy such certificates.

The publishing of certificates and CRLs on web or FTP servers has the drawback that it does not allow certificate searching. Therefore this solution is rarely employed. When it is employed it is used for the installation of certificates and CRLs.

The access of certificate stores via HTTP does not allow the full LDAP certificate searching possibilities. For example it does not allow to search for a certificate according to its keyUsage. If the certificate store that is used is a database, then the namespace for storing the certificate attributes is proprietary while in LDAP it is standardised. An LDAP server can also be used as a certificate store.

6.2 LDAP and PKI

LDAP directories are used to hold certificates and CRLs in order to provide dissemination of PKI information. Many organisations already operate LDAP directories. One of them is Bundesnetzagentur,[3] the authority responsible for the German root CA, according to §3 of the German Signature Act[4] [Leg01a]. Bundesnetzagentur uses an LDAP directory to provide public availability for qualified certificates as §2.12.b SigG requires. A report for using Microsoft's Active Directory, which supports LDAP, along with PKI in the corporate environment for 300.000 users is found in [GSB$^+$04].

Typical clients (like email clients) already have LDAP interfaces to retrieve CRLs and certificates. Nevertheless, the way that LDAP behaves in some cases, complicates its support to PKI. Chadwick [Cha03] makes an in-depth study on this behaviour. Gutmann [Gut00] suggests that a relational database is a better solution than LDAP as far as storing of certificates and CRLs concerning.

Here we concentrate on the use of LDAP. We will show that LDAP, if it is operated properly, offers solutions regarding directory services that have many advantages. Moreover LDAP is a commonly used standard for supporting such services. Many PKI clients as well as PKI practitioners already employ LDAP directories.

6.2.1 Storing Certificates

In LDAP a problem is how to successfully publish certificates. One question is what is the best schema (see [WCHK97, Sec. 3] for the definition) choice. One aspect of this is about whether there is support for a user that possesses more than one certificate. Another aspect is about the presence of mandatory steps

[3]The complete name is Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen. It is the former RegTP which stands for Regulierungsbehörde für Telekommunikation und Post.

[4]We abbreviate the German Signature Act as SigG from the German term which is Signaturgesetz.

(that will reduce flexibility) or collisions with the schema already used. This question is answered both for user and CA certificates. Second question is how to successfully arrange the information to enable clients to search for the correct certificate.

User Certificates

Information is hierarchically organised in an LDAP directory. In Figure 6.1 we can see the entry of the user `CN=Alice, O=Org, C=DE, DC=MyOrg, DC=DE`. The values between the commas are called relative distinguished names (RDN) and the whole value is called a distinguished name (DN). Every entry has a relative distinguished name. By moving from this entry to the root, and concatenating the relative distinguished name of every entry that is met, the distinguished name for this entry is built. Relative distinguished names are usually of the form `attribute=value` but they can also be multi-valued and presented like `attribute1=value1 + attribute2=value2`. This representation gives the ability to create unique RDNs among siblings in the directory. For example, in the case of certificates the issuer name and the serial number together form a unique representation [HPFS02, Sec. 4.1.2.2]. This representation could be used on the LDAP to create these unique entries in the form of a multi-valued RDN.

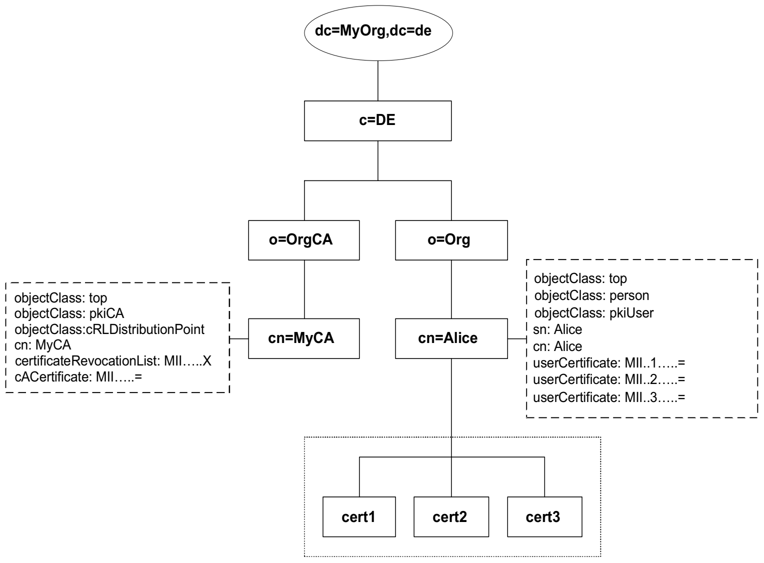

Figure 6.1: Entries on LDAP for a CA and an end-entity

Every entry on the LDAP has attributes and every attribute has one value, or more in case of not single-valued attributes (see [WCHK97] for details). All entries have an attribute called *objectClass*. This attribute describes the type of the entry on the LDAP. Based on the values of the objectClass attribute, the entry can have different properties. New properties can be applied to the entry by extending the values of the object class. In Figure 6.1 the attributes for an end-user and a CA entry are shown. Usually, but not necessarily, the subject distinguished name contained in the entry's certificate matches the distinguished name of the entry in the directory. In the end-user entry the *person* object class as well as the *pkiUser* object class are present. The first one gives the entry the ability to hold attributes like the surname or the telephone number. The second one enables the entry to hold the *userCertificate* attribute which represents the X.509 certificate for this entry. The userCertificate attribute holds the certificate in its binary form in the directory.

The schema choice is important in the case of certificate publishing. The solution should not make any assumptions on the current status of the directory as well as it should not be planned to support only PKI. LDAP directories can also be used to hold information that is not related to PKI. One choice in order to store a certificate on an entry is to use the *strongAuthenticationUser* object class [Wah97]. The drawback with this approach is that this object class requires that a certificate is present on the entry and therefore it cannot be used in cases where the entry does not possess any certificates (for example, the certificate is removed from the directory after its revocation). In addition, this object class refers to the X.509 strong authentication mechanism and it should not be used just as a certificate container. To overcome these problems the *pkiUser* object class [BHR99] can be used. This attribute allows but does not require that the userCertificate attribute is present. Therefore the previous drawback does not apply in this case. Lastly, the *inetOrgPerson* object class allows the certificate to be published on the entry too. This object class may be used when the directory is not only used to support PKI but other organisational procedures and data.

CA Certificates

The object classes used to enable publishing of CA certificates differ from the ones for end-users. There are two choices. The first one is the *certificationAuthority* [Wah97] and the second one is the *pkiCA* [BHR99]. The certificationAuthority mandates that a CRL, the CA certificate as well as an authority revocation list[5] (ARL) [IT97] must be published on the directory. But very few CAs issue ARLs or the CA represented on this node may decide not to issue CRLs itself. Therefore, this object class is not the best choice. The more flexible pkiCA [BHR99] solves these problems. This object class allows, but does not mandates, CRLs, ARLs, and CA certificates to be published. The CA certificate is represented by the *caCertificate* attribute, the X.509 CA certificate in its binary encoded form. Lastly, both object classes allow the *crossCertificatePair* attribute. This attribute represents the forward and reverse (or issuedToThisCA

[5]A list associated with revocation information for CA certificates only.

and issuedByThisCA) cross-certificate pair for cross certification purposes.

Certificate Search

Certificates are stored in LDAP as binary objects at the entry of their corresponding entity. Therefore locating the correct certificate requires by a client to know this entry. This happens by an email search or by knowing the distinguished name. Then the client downloads the certificate, parses it, and lastly examines whether this is the one it searches for or not. This process however, delegates the certificate search to some kind of side-information search, in fact some information related to the owner of the certificate and not the certificate itself.

A solution to this problem is to publish information related to the certificate directly on the directory. This will enable the direct search for this data. There was a work in progress in the IETF [IET] that specified such meta-data that are published together with the certificate. This work is easy to implement, since the attributes can be extracted from the certificate and published along with it. This simplifies the search for certificates with special attributes, since now on the node the properties of the certificates (like serial number, subject DN etc.) are held as attributes of the node.

Another work in progress in IETF [CL] specifies special matching rules on the directory to locate the correct certificates. This work is more extensive than the first one but more difficult to implement. This implementation must take place at the side of the LDAP vendors and clients. A third solution is the component matching. This is defined in [Leg04]. In this scheme a client searches for a certificate by one or more of its ASN.1 components (like the serial number or the subjectDN). The LDAP server tries to match a certificate by describing the certificate in its ASN.1 representation and matching the two ASN.1 representations, namely the one from the client and the one from its internal store. In this case the certificate meta-data do not need to be stored as attributes on the entry of the certificate. More details on the component matching for use in PKI as well as its implementation in OpenLDAP [Ope04] can be found in [LCZ05].

The first work also proposes that every certificate has its own entry subordinate to the user entry (for a simple and not detailed draw of this idea, see Figure 6.1). This can help solving the following problem. When a user owns more than one certificate, this is arranged in LDAP with a multi-valued attribute, namely the userCertificate. A problem with this approach arises if someone wants to locate only one specific certificate for this user. Then, one must download all certificates and find out the correct one by searching for the correct certificate locally. But if every certificate has its own entry then this problem does not occur anymore. A more generic solution to this problem is a new standard from IETF [CM04] which defines a method, that enables matching of the values (and not of the attributes which is common in LDAP) with a special filter associated only with the values. Current practice should orient on the subentries solution since most LDAP servers and clients do not support the special matching rules and returning of matched values. Long term planning however, should incorporate the most generic solutions which are on the other hand vendor and client dependent. Similar solutions can be applied to CA certificates.

All solutions discussed here, should be used carefully with regard to reliability of the information contained on the LDAP server. This information is not signed information and it could have been manipulated from an attacker. Every client must verify the signature on the certificate and CRL to ensure the security of the data. Nevertheless, LDAP has a number of security features which can be turned on to guarantee that no unauthorised changes occur in the directory. It can then also meet certain requirements of the Signature Act and Signature Ordinance [Leg01b].

6.2.2 Security

Various security mechanisms can be applied to LDAP directories. They address different problems and cover many security aspects of an online electronic system. These attractive security features make LDAP directories a perfect candidate for the directory services in the SigG context. First of all they allow different means of authentication. One is the simple authentication with password. Another possibility is to combine it with SASL [Mye97] for a digest based authentication. For a description see [WAHM00]. Lastly, typical TLS client authentication is also possible.

TLS [DA99] can also be used for securing the network traffic to the LDAP server. The server authentication is a necessary step in this protocol, and therefore the LDAP server must authenticate itself to the clients. TLS can be used to avoid that the passwords will travel in clear in the simple password authentication. For more on this scheme see [HMW00].

Apart from the authentication and confidentiality mechanisms, access control mechanisms are applied to LDAP. The directory administrator can set access control lists (ACLs), in order to allow certain actions to definite individuals and special parts of the directory. This scheme allows granular controls concerning persons, actions, and data in the directory. An application of the ACL possibilities could be that a revocation authority is able to publish only CRLs on the directory, while the certification authority (CA) only certificates. In [Cha00] and [Gre02, Chap. 5] more on the security of LDAP directories can be found.

There are several cases, in which these security mechanisms are applied. For example, in order to avoid that an unauthorised user replaces a revocation list with an older one, or just removes a certificate. Although certificates and CRLs are signed information and therefore their correctness can be proved, the integrity of the data on the directory is very important in order to prevent such a misuse. In addition, using the TLS mechanism of LDAP, the server can authenticate itself.

Thus, the publishing client can check whether it publishes the information on the correct server. This ensures that the certificates and revocation information are published in the correct place, where the clients are expecting this information to be published.

6.2.3 Storing CRLs

CRLs are stored in a special LDAP attribute called *certificateRevocationList* defined in [BHR99]. There are three object classes that allow this attribute, namely the *certificationAuthority* from [Wah97], the *pkiCA* and the *cRLDistributionPoint* defined in [BHR99]. The first one cannot be used when indirect CRLs [HPFS02, Sec. 5] are used, since the entry on the directory is not a CA at all and no CA certificate must be present. The other two object classes allow the certificateRevocationList attribute without mandating this or other attributes. In addition the cRLDistributionPoint object class mandates the *commonName* attribute which gives the ability to build an entry on the LDAP with the CN node. The best choice is a hybrid solution where both object classes are present (not necessarily on the same entry). The pkiCA is then used to publish the CA and cross certificates and the cRLDistributionPoint to publish any revocation information.

In case of indirect CRLs the cRLDistributionPoint object class is the only suitable choice. In this case the CRL issuer is not the issuer of the certificate and the keyUsage on its certificate is for CRL signing. This CRL issuer does not possess any CA certificate and the pkiCA object class as a choice would be just wrong, since it denotes a certification authority. In the case of the German root authority the CRLs issued are indirect ones. This is an implication of the special validity model of the SigG. For more on this validity model see [Bau99].

In an X.509 certificate there is the possibility to state inside the certificate where to find revocation information about it. This is arranged in a special extension called *cRLDistributionPoints* [HPFS02, Sec. 4.2.1.14]. In this extension the place where clients can locate the CRL related to this certificate is provided. This can be an LDAP URL and/or an HTTP URL, among others, which implies also the mechanism used to obtain the CRL. Use of this extension is recommended since it simplifies the revocation information management. The LDAP URL for searching the CRL of the CA entry in Figure 6.1, according to the format defined in [HS97], is:

```
ldap://hostname:389/CN=MyCA,O=OrgCA,C=DE,DC=MyOrg,DC=DE?
certificateRevocationList?base?objectClass=cRLDistributionPoint
```

The value of this LDAP URL notifies the client to do the following:

1. Connect to the LDAP server running on host *hostname*.

2. Connect to the port *389* on this host.

3. Go to the entry *CN=MyCA,O=OrgCA,C=DE,DC=MyOrg,DC=DE*.
4. Search only this entry (base search) whether it contains the objectClass cRLDistributionPoint.
5. If the search matches, retrieve the certificateRevocationList attribute and therefore download the CRL.

Delta CRLs can be published in the LDAP directory, too. The corresponding attribute is the *deltaRevocationList* and an additional object class that can hold this attribute is the *deltaCRL* [BHR99]. The cRLDistributionPoint object class can also hold delta CRLs. However, the way this information is arranged in the LDAP directory creates some management problems.

Every time a delta CRL is issued a full CRL must be issued, too. This is done in order to enable clients, that cannot handle delta CRLs, to have the freshest revocation information. This was a mandatory step in the deprecated PKIX specification for X.509 certificates [HFPS99]. This suggests that the newer CRL replaces the base CRL (it is typical that the newer CRL just overwrites the older one in the directory). This is not a problem if the clients already have the base CRL in their cache. But new clients (or even clients which were offline when the base CRL was placed in the directory) have to work with complete CRLs until they have the chance to locate a base CRL in the directory. To overcome this, the newer CRL must be just added to the directory and not replace the older one. In this case however, the clients must download all CRLs in order to find the base CRL.

In the newer PKIX specification [HPFS02], when a delta CRL is published, it is mandatory that the corresponding base CRL must be found in the directory. The new specification allows also to combine the delta CRLs with a full CRL that is newer than the base CRL. Therefore a client does not need to locate a base CRL anymore. In this case it is sufficient if only two full CRLs can be located on the directory. The base CRL and the latest full CRL. A client can use both. But in both cases (especially as far as the older specification concerning) an LDAP mechanism to distinguish between the different CRLs is needed. This can be done by using the matching of returned values [CM04] and special search filters [CL] or component matching. For CRLs, a similar approach to the one about certificates that are stored in subordinate entries has also been proposed at the IETF [IET]. This work supports easy searching for CRLs on the directory.

6.2.4 Naming Plan

In order to arrange the information effectively in the hierarchical manner of an LDAP directory there is a proposal in RFC 2377 [GHSW98]. This RFC proposes the use of the domain component attribute for the root of the directory. Domain names are unique and hierarchical. Therefore they can be used in order to provide unique names in the LDAP model. In [KWG+98] the appropriate object classes for realising this proposal can be found. In the Active Directory of Microsoft, the use of domain names as the root of the directory is mandatory [Gre02, Chap. 6]. Active Directory must be installed and configured properly in order to use the integrated CA shipped with Windows 2000 Server.

6.2.5 Other Uses of LDAP

LDAP directories can be used for other purposes, too. Many organisations already operate LDAP directories to administrate and manage their information. A common LDAP use is for central authentication purposes in UNIX systems. In [KLW04a] it is shown that LDAP directories can be used for providing proof-of-possession for encryption keys as well as delivery of software personal security environments (PSE). Two other applications are derived from those schemes. One is the activation of certificates. With the term activation we denote the action to make the certificates publicly available. The other application is the private key on demand, in which the users can locate and download their PSEs whenever they want to use it. Guida et al. [GSB+04] have used the Active Directory for the registration purposes inside the PKI, in order to extract identity information for the issuing of certificates. In [LKWB05] an LDAP directory is used for automating registration. An extension of this scheme is in [LKWB06] to be found.

6.3 Directories and the German Signature Act

Qualified certificates must be verifiable and optionally available. The term verifiable implies that a mechanism to determine whether this certificate exists as well as to obtain information about its status is present. The term available implies that the certificate itself is located in a position where everybody can find and download it. Every Certification Service Provider (CSP) therefore, must provide a directory service where certificates and associated revocation information is located. One complete technical solution to this consists of an OCSP server [MAM+99] to keep the certificates verifiable and an LDAP server to have the certificates available.[6]

OCSP servers give signed answers to a client's query about the status of the certificate. There are three possible answers. The first one is good, meaning that the certificate is not revoked. The second one is revoked, meaning that the certificate is revoked and the third unknown, meaning that the certificate does not exist for this OCSP server (it is unknown to it). Therefore OCSP can be used to give information about the status of a certificate. The special extension *CertHash* has been specified by the ISIS-MTT specification [TT], in order to include also

[6]This is the current practice at the German root CA.

an existence evidence of the queried certificate. This extension holds the hashed value of the certificate the clients want to verify. With this OCSP configuration the OCSP meets the verifiable requirement. According to §15.2.2.b SigV, the verification of a qualified signature requires also a proof of existence for the qualified certificate, as well as information about its status, at the signature creation time. Further we show how LDAP can be used in the SigG context, according to its special requirements, for providing availability of qualified certificates. But more mechanisms can also be employed in parallel. A candidate for example is the HTTP store as described in [Gut06]. The above discussion demonstrates that any technical requirements are outside the scope of SigG. This is because the standards and techniques used to satisfy the law are subject to changes.

According to §3.1 SigV, no certificate must be published before its user is identified and accepts his qualified certificate. This means that compliant certification service providers must wait for an activation of the certificate from its user. A typical certification process in the corporate or institutional environment usually automates the publishing of certificates. This automation however, must not be performed in the SigG context. Complying implementations must enable an activation mechanism for the publishing of certificates. Only after the user has accepted his certificate all signatures that he has calculated, even those before the acceptance, are considered in effect. When a certificate is activated, it should be verifiable and if its user requires it, also available. This implies that a certificate upon activation must become unconditionally known to the OCSP but conditionally be published on the LDAP server. Conforming CSPs must distinguish between certificates that must be kept public and those which must not and have the proper mechanism to resolve this.

The above is also discussed in §5.2 SigV. The key owner must explicitly accept his secure signature creation entity and confirm this to the CSP. Then it is the CSP's task to publish the certificate in order to keep the certificates publicly available and verifiable by the public (§2.12.b SigG). After the confirmation from the user, the CSP is obliged to keep the certificate on the directory for at least five years after its expiration year (§4 SigV). In the case of accredited CSPs this time window is thirty years after the certificate expiration.

Compliant publishing components should therefore call only add functions to the directory but never delete functions. In LDAP this is translated to calling add requests and never call delete or modify requests (for the definition see [WHK97]). This gives the ability to the developer to ensure correct function and compliance to the directives. If any remove action must be performed (for example after the thirty years), then this should be done manually with the help of special clients. The presence of a special monitoring mechanism is required. This mechanism will inform the administrator of his actions. For example, this mechanism will first fetch the certificate from the LDAP before the delete request, show this to the administrator, who must check whether this is the correct certificate or not, and delete it after the administrator's acknowledgement.

Moreover, a special mechanism should exist that notifies an administrator in case of a failed operation. If an intended LDAP operation is not successful, the system must notify the administrator for the problem. The problem must be explicitly stated, in order to give the administrator a clear view of which step was not performed. Then, an out-of-band mechanism should be used to perform the unsuccessful step. This out-of-band mechanism must have the same security properties as the regular one. Apart from this, a regular backup of the data on the directory must be performed, for example by creating backup files of the whole directory. This can be done with the help of the LDAP data interchange format (LDIF) [Goo00]. LDIF is a text format that describes the data on an LDAP directory. The importance of the data persistence in the directory is depicted also in §15.3 SigV. The certificate presence as well as revocation information related to it must have a continuous state in the directory. In addition, in the case of an accidental error or serious system crash, the status of the directory can be retrieved as it was before this event.

According to §7.2 SigV, if a certificate is revoked this information must be published in the directory. In the LDAP case this is done by storing the CRL. As in the case of certificates, it must be ensured that the CRL publication was successful and if not, special countermeasures must be taken. For publishing a CRL the activation mechanism is not needed. In this case it is very important that as soon as the CRL is produced from the CA,[7] this CRL is also published in the directory without any delay. Another difference is that in the CRL case a modify request is usually called, in order to replace the existent CRL in the directory. There is no need to keep all CRLs in the directory since the newest CRL contains revocation information of the previous CRLs within the same scope. If delta CRLs are used however, at least a full base CRL that can be combined with them should be found in the directory and should not be deleted.

The access control mechanism of the LDAP is useful, in order to distinguish between a CRL and a certificate publishing. The ACLs in the directory can be configured so that a special certificate publishing user is only allowed to add certificates (namely the userCertificate attribute) and in a special path in the directory. Another user, the CRL user, can publish only CRLs (namely the certificateRevocationList attribute). The other security mechanisms must be used, too. The TLS connection guarantees the identity of the LDAP server and the client authentication ensures the identity of the user publishing data on the LDAP. Then, no unauthorised actions on the LDAP can be performed (like removing certificates or replacing CRLs) and moreover, publication is done only in the correct directory.

[7]Or a revocation authority since the CRLs used in the SigG context are indirect CRLs.

6.4 Conclusion

We analysed the LDAP directories and their use in PKI. LDAP is a reliable technical solution to address the PKI information dissemination problems. We took a look especially in the SigG environment. We discussed a planning for LDAP, in order to meet the special SigG requirements associated with the public availability of certificates and revocation information. A careful planning for LDAP, provides a successful support to PKI as well as to other organisational procedures.

A future work in this direction is to design and implement an LDAP PKI client for management of certificates, CRLs, and software PSEs. A new standard is specified in [MSNP04] which defines a synchronisation method of an LDAP client with the LDAP directory. This technique may be employed for automatically downloading new CRLs and updating certificates of users which are already installed locally.

Chapter 7

PKI Management Schemes

There are certificates management tasks that are necessary in every PKI. Responsible for realising them is the certificate management authority. For some of those tasks current solutions do not provide satisfactory solutions. In order to provide a flexible and efficient CMA we take a look at two tasks. The first one is the proof-of-possession. The second one is the secure delivery of software personal security environments. Parts of this Chapter have been published in [KLW04a].

We present a method for extending the functionality of LDAP servers from their typical use as a public directory in public key infrastructures. In this method the LDAP servers are used for administrating infrastructure processes. One application of this method is a scheme for providing proof-of-possession, especially in the case of encryption keys. Another one is the secure delivery of software personal security environments. We see what are the differences to the current situation and other protocols. We discuss their implementation, their advantages, and the usability gains through their deployment in a PKI.

7.1 Introduction

The task of the RA is to register an end-entity in the PKI by examining its credentials. This entity may either already possess a key pair or not. In the first case, the public key is provided by the entity to the RA during registration and forwarded to the CA that issues the certificate. In the second case, the key pair is produced by the CA and, together with the issued certificate, is delivered to the entity. In either case, the CA has to guarantee that finally the certificate is issued to the entity that owns the corresponding private key.

The problem that arises in the first case is for the CA to determine whether this entity does really own the corresponding private key or not. Therefore, the entity must provide a Proof-of-Possession (PoP) for this private key. In several certification request syntaxes special fields are reserved for this purpose. An important issue with PoP is the number of messages needed for accomplishing it. While for signature keys the straightforward solution is to provide a signature which can be verified by the RA or the CA, for encryption keys such a solution does not exist. One good solution in terms of number of messages is the indirect scheme where the issued certificate is encrypted with the included public key and thus can only be read by the intended recipient. We provide an easy and efficient variant of this scheme using a directory based on the lightweight directory access protocol (LDAP) [HM02]. Furthermore, it can be extended to tightly link the PoP to the registration procedure. Moreover, a successful PoP is necessary for activation of the certificate. Whether or when this activation will take place is under full control of the end-entity.

In the second case, in which the key pair is generated in the trust center, the problem is to securely deliver the private key to the registered and identified entity. A commonly used standard for software personal security environments (PSE) is PKCS#12 [RSA99]. Usually, such PSEs are either handed out face to face or sent by e-mail. For automated certification processes in which management should be simple, the solution of physical presence cannot be used since it hampers the automated process and requires human administration. PKCS#12 has inherent mechanisms to ensure privacy and integrity. Therefore, it can be sent with an e-mail. However, this is not a good solution. This is because the e-mail may be intercepted by a third party which can try to extract the keys (in a password protected PKCS#12 file this may be easy for a weak password). Moreover, the e-mail may never reach the recipient due to an incorrectly configured spam filter or a mailbox which has reach its capacity limit. We suggest a scheme based on an LDAP directory to ensure that only the legitimate recipient will receive his PSE without any eavesdropping from a third party.

7.2 Certification Request Messages and PoP

Common practice for the registration of end-entities in a PKI environment are special certification request message syntaxes. Several specifications address this problem.

The most commonly used one is the PKCS#10 [RSA00]. It defines a syntax in which entities can provide information needed for creating a certificate from the CA. Furthermore, it enables them to include other data that can be used during and after the certification process. This syntax requires the message to be secured with a digital signature. Among other purposes, this is done in order to provide a PoP for the corresponding private key, since the signature can be verified by the receiving party (RA or CA). However, this syntax does not consider keys used for encryption only. Such keys may have to be treated differently due to limitations of the cryptosystem (for example it cannot be used for signing) or different security requirements.

Another specification is the Certificate Request Message Format (CRMF) [Sch05]. While handling signature keys identically to PKCS#10, it provides three methods for PoP of encryption keys. The first is to reveal the private key to the CA. The second one, called direct method, requires exchange of challenge-response messages and thus needs extra messages. The third one, the indirect method, is to encrypt the issued certificate with the contained public key and have the end-entity to send the decrypted certificate back in a confirmation message and by this demonstrate ownership of the corresponding private key.

Similar proposals can be found in Certificate Management Protocol (CMP) [AFKM05]. Special care about encryption keys is taken also in Certificate Management Messages over CMS (CMC) [MLSW00]. CMC specifies a mechanism for PoP which requires more than a single-round trip[1] and therefore complicates the process. Lastly, XKMS [W3C01] considers the PoP problem exclusively for signature keys. For a discussion on PoP see also [ANL03].

The above proposals do not provide satisfactory solutions in cases where the end-entity does not want to reveal its key or the certification process management should be kept minimal and simple.

PoP is of great importance in the case of encryption keys. A certificate containing a public key for which the corresponding private key is not owned by the intended end-entity or does not exist at all is actually unusable. Any message encrypted with this public key cannot be decrypted by the end-entity. This scenario can become extremely dangerous in cases where the original message is destroyed[2] and only the encrypted message exists. Therefore, no one should be able to use a certificate containing encryption keys without prior acknowledgement of its owner.

To solve the PoP problem we propose a scheme similar to the indirect method described above, in which an end-entity is an authenticated user of an LDAP directory, where he can download and activate his certificate from. But in order to do so, he must decrypt a secret (implicit use of his private key), decrypt the certificate and then put it back on the LDAP server. Thus, we can avoid extra confirmation messages to the CA.

7.3 Providing PoP with LDAP

We present a scheme for providing PoP for encryption keys using an LDAP directory.

In this scheme the CA is accepting a certification request for encryption keys. At this point the CA assumes that the private key exists and issues the certificate. After that, it creates an entry on the LDAP directory. For this entry the cer-

[1]A request which is done from the end-entity, processed from the CA, and returned to the end-entity.

[2]This is a typical function of many encryption clients.

tificate is encrypted with the public key that it contains and its encrypted form is stored in the attribute *encryptedUserCertificate* (see Section 7.4). Secondly, the CA randomly chooses an LDAP user password. Its hashed value is placed in the entry's attribute *userPassword* for the simple authentication procedure of the LDAP. It also encrypts the password using the entity's public key and puts it in the special attribute *encryptedUserPassword*. At this point even the user itself does not know his password, since only its hash value and an encrypted version is found on the directory.

We have introduced a new object class called *pkiUserManagement* (see Section 7.4) which can hold the above described attributes. Figure 6.1 shows an example of such an LDAP entry along with its attributes.

The end-entity, that owns the corresponding private key, does the following: First, it downloads the encrypted password and certificate and decrypts them with its private key. Now it can use the password to bind to the directory as an authenticated user and write the decrypted certificate in its entry. With this the PoP is completed and, simultaneously, the certificate is activated enabling other entities to use it. For an end-entity that does not own the private key, both the certificate and the password will remain encrypted in the directory and cannot be used. The communication between the authenticated user and the LDAP server must be secured with SASL or TLS to avoid transmitting the password in clear.

Since the user has to decrypt data, an extra check should be made whether this data is what it is supposed to be, namely a password and a certificate. This should be done in order to avoid that the certificate holder decrypts a previously encrypted message with this key[3] and places it on the LDAP. An attacker could capture such old encrypted messages and place them on the directory (provided that it can gain write access to it) in order to realise this. The check for the password is not necessary because the password will not be revealed and it will only be compared with the one found on the LDAP directory. The check for the certificate is the verification of its signature.

One step that the CA must perform is to force the LDAP directory, by proper access control lists (ACLs), to accept write requests only for authenticated users and only for the *userCertificate* attribute of their own entry. This enforces that exclusively authenticated users can place their certificate only on their own entry in the directory. In addition, these ACLs must be configured in such a way that the user password (although hashed) is not visible to other users. This will eliminate the possibility of dictionary attacks on the password performed by any end-user.

An optional variant of this scheme, which can link both identity and PoP information more tightly together, is the following: The encrypted password (when decrypted) is only the half of the real password[4] used by the user to

[3]This key may have already been used for encryption purposes in the past.

[4]Which is found hashed to the directory.

authenticate to the LDAP directory. The other half of the password is provided to the user during registration. This half works as a shared-secret between the CA and the end-user. The user, in order to authenticate to the directory, has to combine the two passwords.

A second variant is to have the shared-secret function as the (complete) password for the user to authenticate to the directory. PoP will follow with decryption of the certificate without having to decrypt the password. In this case the ability to authenticate to the directory is disconnected from the PoP. Similar is the case in which the user already possesses its LDAP password.

CAs can realise this scheme differently based on their policies. For example, if after three days the certificate is still encrypted in the directory, the user is deleted from the directory leaving him no possibility to use this certificate. Alternatively, after decryption of the certificate the user password is deleted in order for the user to be unable to authenticate to the directory.

7.4 Schema Definitions

We provide the schema definitions for the objectClass and attributes defined for supporting the PoP with LDAP.

Object Class

```
( 1.3.6.1.4.1.8301.3.2.2.1.6 NAME 'pkiUserManagement' SUP top
AUXILIARY MAY ( userEncryptedPassword $ userEncryptedCertificate ) )
```

Attributes

```
( 1.3.6.1.4.1.8301.3.2.2.1.7 NAME 'userEncryptedPassword'
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 SINGLE-VALUE )

( 1.3.6.1.4.1.8301.3.2.2.1.8 NAME 'userEncryptedCertificate'
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 SINGLE-VALUE )
```

7.5 Methods Overview

We discuss the characteristics and properties of the certification request messages presented in Section 7.2 and our proposed LDAP based solution. PKCS#10 and XKMS do not support PoP for encryption keys (Encryption Key Support, EKS) and we will not discuss them further in this section. CMC and the LDAP

	CRMF	CMP	CMC	Proposed Solution
EKS	+	+	+	+
DM	+	+	-	-
PKR	-	-	+	+
SRT	-	-	-	+

Table 7.1: Comparison of the schemes

based proposal specify only one possible method to provide PoP, while CRMF and CMP offer three different ones (Different Methods, DM). Nevertheless, the last ones propose revealing the private key (Private Key Revealing, PKR) as a solution. Furthermore, CMC requires more than a single-round trip where all other methods do not. But CMP and CRMF require that a confirmation message should be sent to the CA. Our proposed method does not require those (Single Round Trip, SRT). In Table 7.1 we can see an overview of these characteristics.

7.6 Secure Delivery of Software PSEs

A variation of the above scheme can be used for delivering software PSEs (usually in the PKCS#12 format). In this scenario the PKCS#12 structure is placed in the *userPKCS12* [Smi00] attribute of the LDAP entry. The connections done to the directory must be secured with TLS. If a user can authenticate to the directory, then he has the ability to download its own PSE. But in order to authenticate, he needs a password provided during registration. Therefore only the intended user can access his PSE.

To enforce this, the CA has to choose special ACLs. Only the authenticated users must have read access to the *userPKCS12* attribute of their own entry, while all others should have no privileges at all. Downloading the PSE is performed with integrity and confidentiality due to the TLS connection.

We have implemented both schemes to provide a proof of concept. We have used the OpenLDAP [Ope04] directory since it gives the possibility to create flexible ACLs and supports TLS and SASL. We have used JNDI [SUNa] at the CA and client side for the LDAP interfaces. Moreover, the PoP scheme has been used in the Technical University of Darmstadt. In this PKI environment the students receive a pre-keyed card. In order to get and activate their certificates they have to provide the LDAP based PoP. Moreover, if a card was not sent to the correct student, the wrong recipient will not get a valid certificate for this card. The incorrect recipient must also know the LDAP data (like the password) of the correct student for using the card.

7.7 Usability Issues

These two basic applications for user management with LDAP enhances the usability of the PKI. In the PoP case it is easier for a human user to decrypt a value than to provide his private key or to engage himself in challenge-response procedures. Decrypting data like the certificate or the password is the intended use of the decryption key and is already implemented in common client software. Furthermore, most existent client software supports LDAP for searching and downloading certificates of others. These features can easily be extended and combined to fetch the encrypted data from the directory, decrypt it, and store the certificate back. Particularly, the last action can be totally transparent to the end-entity.

Also, fetching a PSE from the directory is essentially the same as looking up certificates of others. Thus, no extra development is required for the client software. The usability is increased by the fact, that the client can transparently download the PKCS#12 file and integrate the contained private key at the client side.[5] Instead, if the user has received his PSE by e-mail or floppy disc, he would have to install it somehow manually. Furthermore, only the LDAP connection has to be configured at the client and nothing else. Depending on the CA's policy, also the following strategy may be chosen: The software PSE remains on the directory to serve as backup. In this case the users are not required to permanently store the PSE locally. They can access it from everywhere and anytime. Herewith a key on demand service is realised. Backup is also supported.

7.8 Overview

We have presented two schemes for providing proof-of-possession and secure delivery of personal security environments. Both schemes use an LDAP directory which is already integrated in most PKIs, it is ideal for user management, it has sufficient security features, and many existent clients have already LDAP interfaces which can be extended. We have integrated the PoP solution into the PKI installation at the University of Darmstadt. The current schemes are extended in [LKWB05] for supporting, registration, certificate revocation, and certificate renewal.

[5]Probably asking for an extra password for its internal database.

Chapter 8

CMA Design and Implementation

In Chapter 4 the specifications for the CMA have been given. We use these specifications for designing and implementing the CMA. We see the design of the CMA as a software component, the types of request that are possible to be processed as well as the certificate management protocols that it uses like the PKCS#7 or the ITP. The tasks of the CMA regarding certificate management are explained in details. For realising these tasks the certificate management plugins have been developed. Their specification, design, implementation, and the benefits of this approach are discussed. Moreover we take a look at the design of the CMA for supporting different input formats, virtual hosting, and LDAP directories related functions like publishing of certificates and CRLs. We discuss the technologies that are used for implementing the CMA and finally see whether the CMA design goals have been met.

8.1 Design

We see the design of the CMA. First we give an overview of the main design principle. Afterwards we provide more details about the messages that the CMA is able to process. These messages are the input to the CMA. They carry different kinds of request. We discuss details about the types of request that exist in a trust center and the CMA is able to handle.

8.1.1 Overview

The CMA is designed as a software daemon. In Figure 8.1 we see a UML activity diagram of the daemon workflow.

The CMA daemon is a program that runs in the background and delivers certain services. It can be started as a system service or as a stand-alone appli-

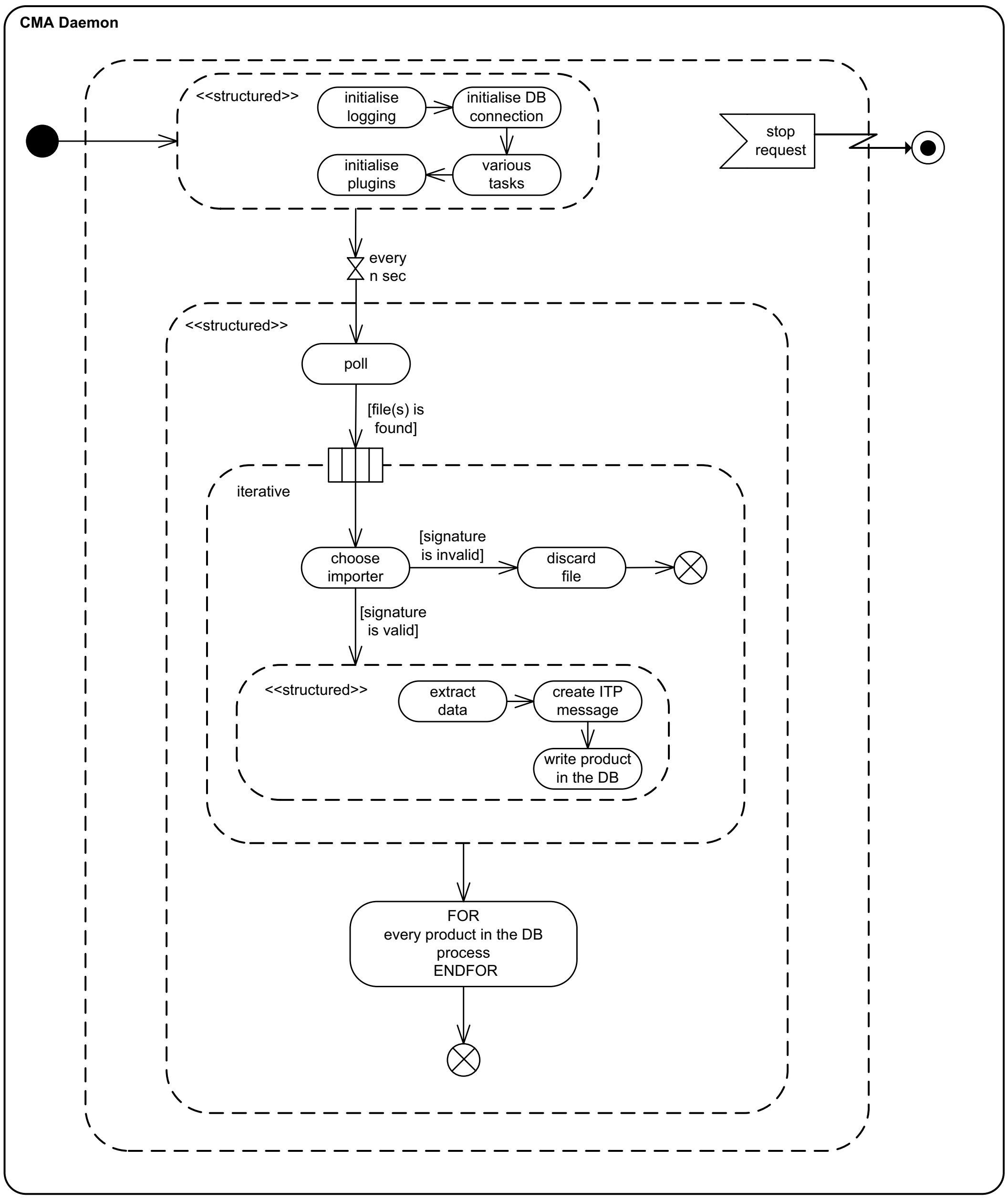

Figure 8.1: Activity diagram of the daemon workflow

cation. For delivering its services it must be triggered with special inputs which we call requests. These requests are created by an issuer. As soon as a valid request is sent to the CMA, it is processed according to the workflow defined for this type of request.

There is a start up phase and a running phase. The start up phase performs all necessary initialisations for the correct functioning of the application. These are the initialisation of the database connections, reading of the configuration files, initialisation of the logging and of other miscellaneous tasks. If any failure occurs this is logged and the operator of the CMA is informed with a console output or by monitoring the log files.

After start up the CMA runs as a daemon. It runs in the background as an application that waits for delivering certain services. In order to do so the daemon performs a regular polling. The polling time is between three and five seconds. This time is configurable.

The design allows different polling variants. The one that is used in most cases is filesystem polling. Another polling mechanism is over a database. One could think of more polling mechanisms based on LDAP directories or events like the insertion of a smart card into a card reader.

In the case of the filesystem polling, the daemon checks one (or more) filesystem directories for new files. These filesystem directories are located at the same machine in which the CMA daemon is running. If these files match the expected format (PKCS#7 or ITP) they will be imported. The input files are tested for integrity and authenticity. Afterwards the daemon extracts all necessary data from these files. These files contain requests that are processed by the CMA.

The first task of the daemon is to identify for which virtual host the request is destined for. If it is destined for a known host, then it imports it, otherwise it discards it. When a request is imported, the daemon checks its type. There are different types of request like certification, revocation, or certification with key generation. Other requests exist, too. For more on the requests see Subsection 8.1.3. Depending on the type of request a predefined set of steps are performed by the CMA. This set of steps realises the management tasks required by the issuer of this request.

The processing steps of a request to the CMA are configurable. This is arranged in special configuration files. The request reaches a plugin component (or simply plugin) which is the first processing step. This component delivers a special service. When the component has delivered its services for this request, the request is moved to the next plugin component. This continues until a request has been processed by every plugin.

The set of services that these plugins offer together for each request is exactly the way that the CMA processes this request. This means that the realisation of the CMA's services is done through many pluggable components. The collaboration of all components fulfils the task of the CMA for one issuer and one request.

8.1.2 Input

The CMA accepts two input formats. One is the PKCS#7 format and the other is the ITP format.

Before the data is transferred to the CMA, it is encoded to the corresponding format and stored in a file. This file is placed in a filesystem directory. This is found on the same machine as the CMA instance is running. The CMA daemon polls this directory regularly in order to examine whether new input files exist. If a new file is found it determines its type (PKCS#7 or ITP) and uses the corresponding importer for reading the file, extracting the data, and storing it in a database as a new request that will be processed. From the moment the request is stored in the database the daemon processes this request further.

Both formats allow the contained data to be signed and encrypted. The different possibilities that the two formats offer are shown in Chapter 5.

The first task after reading a file is to verify the digital signature that protects its content. Every format has its own security mechanisms. In the PKCS#7 messages the data is contained in a SignedData structure. The CMA verifies the signature over the data (see [RSA93] for more details). This data is raw binary data. In the case of ITP messages, an enveloped signature is verified over the data contained in an `<application>` element. If the daemon fails to verify a signature over the input data, it will discard the input.

8.1.3 Requests

There are different types of request in a trust center. The type of request is decided in the RA, since this is the initiator of all requests. The RA sends a request to the CA or KA. When they finish processing the request, this request is forwarded to the CMA. The RA can also send the request directly to the CMA. In Table 8.1 the requests that the CMA processes are found. These are:

- *CRTRequest* is a plain certification request in which the keys are generated at the side of the end-entity. They are not generated in the trust center. The entity sends its public key to the trust center and the trust center issues a certificate for this public key. A PoP is performed in this case.

- *P12Request* is a request in which the keys are generated by the trust center and they are stored according to the PKCS#12 format. Therefore they are stored in software. No PoP is needed in this case.

  In this request the keys are generated by the trust center. This is a logical assumption for keys that are generated in software and many PKIs operate this way.[1] In this request the PKCS#12 file is sent to the CMA in order to deliver it to the user. It is named after the PKCS#12 standard [RSA99].

[1]For example the RBG CA of the TUD is operating like that. See also `https://cert-ra.rbg.informatik.tu-darmstadt.de/` (date of access 28.12.2005).

Request	**Type**	**Description**
CRTRequest	Certification	Request for certification of a public key that has been created at the client side.
P12Request	Certification	Request for key generation in the trust center and certification of the public key. The key pair is stored in software (PKCS#12).
P11Request	Certification	Request for key generation in the trust center and certification of the public key. The key pair is stored in a smart card.
CombiRequest	Certification	Request for generation of more than one key pair for one entity and certification of the public keys. The keys are generated in the trust center.
CRLRequest	Revocation	Request for revocation of one certificate.
CRLRenewalRequest	Revocation	Request for renewing a CRL without revoking any certificate.
DeltaCRLRequest	Revocation	Request for revocation of one certificate and issuance of a Delta-CRL.
CRLCombiRequest	Revocation	Request for revocation of more than one certificate.
ActivationRequest	Certification	Request for activating a certificate.

Table 8.1: Types of request

- *P11Request* is a request about keys that are generated by the trust center and are stored in a smart card. The key generation itself can take place in software or in hardware. In this request information about the smart card is sent to the CMA. It is named after the PKCS#11 standard [RSA04] that defines a protocol for communicating with smart cards.

- *CombiRequest* is a request in which one entity receives more than one certificate. This is meaningful especially when the certificates are for different purposes. The number of certificates for one entity varies. Usually it is either two or three. In the case of two certificates, one certificate is used for encryption and the other one for digital signatures, while in the case of three certificates, the third certificate is used for non-repudiation purposes. However, these certificates may have other uses.

- *CRLRequest* is a revocation request. In this request a CRL is issued. This CRL contains the same number of revoked certificates as the previous CRL with the addition of a newly revoked certificate. This request contains additional data about this certificate.

- *CRLRenewalRequest* is a request for renewing a CRL without revoking any certificate. This CRL contains the same certificates as the previous one, but the nextUpdate and thisUpdate values of the CRL are different. This request is created when the CRL expiration date has been reached and therefore the older CRL is renewed but no additional certificate has been revoked.

- *DeltaCRLRequest* is a request for the issuance of a Delta-CRL. Only one certificate is revoked. The Delta-CRL contains the previously revoked certificates (from the previous Delta-CRL) and the newly revoked one.

- *CRLCombiRequest* is a revocation request for more than one certificate. Every revoked certificate is listed with its serial number and issuer DN. Only one CRL is issued that contains the previously revoked certificates and the newly revoked ones.

- *ActivationRequest* is a request to activate a certificate. This requests contains the certificate to be activated or the serial number and the issuer of this certificate. It may additionally contain an activation password.

We take a look at two requests and compare them. These are the P12Request and the P11Request. We see their differences in three categories. The first one is about whether keys are transferred or not, the second category is about the accompanying data, and the third is about the delivery to the end-user.

In the P12Request the keys are generated in the trust center. In this request the PKCS#12 file is sent to the CMA in order to deliver it to the user. This file contains a key pair. In the case of the P11Request which is a hardware based request the keys are found in the card only. They are created by the user, or the trust center, or the card manufacturer. Thus, no keys are sent to the CMA in this request.

In the P12Request the password that protects the file can be sent within the request. This may be needed for example by the CMA in order to print it on paper that will be sent to the user as a letter. In the P11Request the PIN of the card can be sent, the PUK,[2] or even none of them if this is not necessary.

In the P11Request the delivery of the card is done by the physical presence of the entity in the trust center. The card can also be sent by post to its user. In both cases the CMA needs data to prepare the delivery. In the first case it needs the entity's name for printing it on a letter for enabling sorting of the card. In the second case it additionally needs the postal address of the entity to prepare a letter that will be used in the shipping of the card. In the P12Request the PKCS#12 file can be sent by email and only the email address is needed. Another way to deliver it to the user is by using the scheme proposed in Section 7.6. In this case the entry of the user on the LDAP directory must be sent to the CMA. This information may also be contained in the certificate of the user.

The two requests require different handling. Therefore we distinguish these two requests and their further processing. Among the other requests similar differences exist. These requests are the ones that the CMA processes.

Depending on the type of request the CMA provides the required services. These depend on the certificate management needs of each virtual hosted issuer in each installation. The challenge for the CMA is to be able to deliver all necessary services for each request as well as for each trust center installation.

8.2 CMA Services

For every request that comes from an issuer, the CMA delivers its services. These depend on the type of the request and the issuer. Every issuer has customised requests as well as services. Therefore the CMA realises diverse certificate management tasks for every installation. The following are the services of the CMA.

8.2.1 Certificate Archiving

Certificate archiving is the storing of the certificates in a persistent store like a filesystem or a database. Through archiving, access to a certificate can be achieved at any time. The suited place for this is the CMA and not the CA, since access to the CA is usually limited.

[2]This is a second PIN that unblocks the card, in case the first PIN has been locked due to wrong entries.

Apart from this, certificate archiving enables certain trust center tasks. One of them is to support certificate renewal. If a certificate has expired, then the RA can read the persistent store, find the certificate, extract the public key and initiate a re-certification request for this key. Another task is to support certificate activation. The activation process locates the correct certificate and initiates an activation request for this certificate.

Through certificate archiving other tasks are also enabled. Such a task is the sorting of certificates according to their expiration date. After that the CMA can send notifications to their end-users about the expiration. In addition, archiving of a certificate can be associated with the archiving of side-information regarding the certificate itself. For example, if a revocation password is needed in order to revoke a certificate, this could be stored (possibly hashed) in the persistent store, too.

8.2.2 Key Backup

Key backup can be delegated to the CMA. This is because the access to the CA is sometimes limited and the CMA is more accessible. The installation of an extra authority, like the Backup Authority, requires more communication costs. Moreover operational considerations may be present, like the existence and maintaining of a new instance.

There are environments in which no key backup takes place. These are for example PKIs that are operating in the SigG context. But in other cases this is a mandatory process. Such a case is when there are keys that are used for encryption.

If the private key is created at the client side, the users are responsible for its archival. They may decide to deliver the private key to the trust center to delegate its archival. If the private key is created in the trust center, then usually the backup takes place in software devices like PKCS#12 structures. The authority that has created the key, usually the CA, stores the key in the PKCS#12 format. This is placed in a request towards the CMA.

The CMA extracts the PKCS#12 data and persistently stores them either in a database or in the filesystem. The PKCS#12 data is either protected by a password or a public key as defined in the corresponding specification. Usually the password encryption is used. Either the password is known to the users (they may have chosen it in the beginning of the registration) or it is created in the trust center and becomes known to them (for example with a letter). In the latter case the password becomes known to the CMA. The CMA may decide to re-encrypt the PKCS#12 data with a new password (like a master password used for all files), or a public key, or archive the initial password. This password is stored encrypted.

8.2.3 Mail Notification

Mail notification can be used in various cases for transporting messages and information from the trust center to the end-entities. The main function of this service is to write an email to the end-users that informs them about their PKI status.

The most typical case is to notify the end-users by an email about the issuance of their certificates. The users then, can download the certificate from the directory or get it from the trust center by physical presence. Alternatively, the certificate may be attached to the email. Other useful information may be included in the email. This could be a list of possible applications for the certificate, usage instructions like limitations of its use (e.g. only for signature), or what to do in case of revocation. The CA certificate may also be attached in the email with installation instructions (fingerprint verification for example).

A second use of the mail notification is to deliver software PSEs. The software PSE (usually in the PKCS#12 format) is attached to an email with the certificate and sent to the users. The users can save the PSE and install it at the client side. In this case the mail is usually sent not encrypted since the users do not possess any private key to decrypt it. Therefore, the password protection is enabled. This password is already known to the user. The protocol described in Section 7.6 offers a more usable alternative and extra security mechanisms.

A third type of mail notifications regards revocation. As soon as a certificate is revoked, its user can be notified about the result of the revocation. In addition, further information like a proposal for a new certification may be accompanying the email.

Mail notifications however, may not be sufficient. The intended recipient may never be reached. This can happen due to wrongly configured filters that protect against unsolicited bulk email or due to an email box that has reached its limited capacity. In cases where this is unacceptable, a certified mail may have to be sent to the user.

8.2.4 Delivery of PSEs

The CMA is responsible for the delivery of personal security environments to the end-users. In case of a software PSE there is the possibility to send it with an email. This is part of the mail notification services. Another delivery possibility is by using the protocol described in Section 7.6. A third possibility is by storing the PSEs in the filesystem. Afterwards a human administrator can extract them and deliver them by hand to the user, usually on a portable media like a floppy disc.

The CMA administers also data related to the PSE. For example it prepares accompanying letters that will be sent with the PSE. In addition, it prints PIN letters that contain the PIN that protects the PSE. These features are most typically met in hardware based tokens.

8.2.5 Confirmations

In many cases it is required that certain conditions must be met in order to perform certain steps with the certificate and its processing inside the trust center. An example for this is the certificate activation. A certificate may not become known to the OCSP, published in a public directory and be accessible before its user decides so. This has as consequence, that a complete certification request has to be suspended for a certain period of time. The term complete certification request denotes the full tasks that are planned to be performed with a certification request inside a trust center.

The CMA supports this kind of behaviour. In case of activation it publishes the certificate only if this is allowed to. Whether a certificate is allowed to be published is communicated from the RA or the CA within the request. If a certificate is activated, it is not automatically published on any directory or the OCSP server. A special trigger mechanism activates the certificate. This may be performed by a human administrator for example. When the certificate is activated, the CMA processes the request for this certificate further. Usually, the only tasks that remain to be done is the certificate publication and the update of the OCSP backend.

These confirmation services are required when end-users must interact with the trust center. This is often needed for example in order to acknowledge the receipt of a PSE. In the installation of the RBG CA at the TUD, the end-users authenticate against a web page in order to notify that they have received their software PSE. The certificate will be published only after a successful authentication.

8.2.6 CRL Notifications

When a CRL or delta-CRL is about to expire, a notification is needed to the CA in order to produce a newer revocation list. This is required if the CA is not already undertaking this task itself. This can be done by sending a mail notification to the trust center administrator for initiating a CRL renewal request. This is a request for a newer CRL without including any new entry. If the CA is offline and it produces the new CRLs itself (without notification), this notification mechanism can be used for informing the administrator to fetch the newer available CRL from the offline system. Alternatively, if the RA is responsible for issuing revocation requests, this notification could be sent to the RA system.

Moreover CRLs can be sent to the clients. The CRL push scheme described in [MR00] is realised by the CMA. The CMA operates as the CRL Push server. The CMA pushes the CRLs of the issuer that it is operating for, to different clients registered with the CRL push services.

8.2.7 PKI Publishing

Various PKI data must be publicly available for different clients. This is important for the proper functioning of the infrastructure. For example there is the need to locate a certificate in order to verify a signature or to encrypt a message. In other cases revocation information must be available for determining whether a certificate is valid or not. In [KLW04a] the PKCS#12 files are stored in a public directory for delivery or key on demand purposes.

Plenty of mechanisms exist for publishing PKI information. In Chapter 6 we have already seen these mechanisms. In most cases the LDAP based mechanism is used. The CMA publishes certificates, CRLs, and other information like the email address (for easier searching of certificates, see also 6.2.1) in an LDAP directory using the LDAP protocol.

Among the various PKI installations different considerations regarding the publication of certificates and CRLs exist. These are influenced by various factors. One factor is whether the end-entity already has an entry on the LDAP directory or not. Another is whether information that is not directly related to PKI products should also be published. In some installations the communication must be secured with TLS. LDAP directories may be used for supporting other services like OCSP. The CMA deals with these factors. A general-purpose library is needed. We see what are the features of this library and implementation details in Section 8.7. This library has been designed and implemented and is used by various PKI components for supporting PKI functions in LDAP.

The PKI publishing should also consider the public availability of some certificates. According to SigG some certificates may never be published on a public directory. There is the need therefore to distinguish among certificates that are public and those that are not.

CMA supports only LDAP based publishing mechanisms since the other mechanisms are rarely used. At the moment of this writing a new protocol is defined in the PKIX group of the IETF that allows retrieving and searching for certificates over HTTP [Gut06]. In this scheme the backend used for the information can be a database, or an LDAP directory, or any other persistent store. Therefore CMA supports the new protocol, if the backend that is used is an LDAP directory. We plan to implement this HTTP interface using LDAP and a relational database as a backend.

8.2.8 OCSP Backend Support

CMA supports OCSP services by providing the necessary store from which an OCSP server will get the necessary information it needs. This information is about whether a certificate exists and whether it is revoked or not. The mechanisms for providing this backend differ.

The store that the CMA is using is an LDAP directory. The OCSP server is querying the LDAP directory for a certificate or a CRL in order to see whether this certificate exists and is revoked. This can be done on demand when the OCSP client sends a request about a certificate to the server. It can also be done as soon as the certificate is published on the LDAP backend by notifying the OCSP server.

Another store is a relational database. Certificates and CRLs are stored in the database in two tables for example. The first table regards certificates and the second CRLs. The difference of this approach to the LDAP based one, is that the information on the database does not have a standard naming mechanism like in LDAP (schema) and therefore the OCSP server needs to be reconfigured for every database.

A third mechanism is by sending direct notifications that contain certificates, CRLs, or other information to the OCSP. Such notifications are special queries to the server. For example when a certificate is issued, a special signed query that contains the newly issued certificate is sent to the OCSP server. Another possibility is to place such information in an ITP message which is then sent to the OCSP server. The server must then store the new certificate for future queries. This could be done by using its own database. Additionally revocations may also be directly communicated to the server without any CRLs being involved. The serial number and the issuer DN of the revoked certificate is sent to the OCSP server which stores this information for example in a locally constructed CRL. Alternatively it may store the revocation information in its database in a special field on the certificate table denoting whether this certificate is revoked or not. In this case security considerations exist since this information is placed unsigned in the database and it must be protected. Similar considerations exist if the requests to the OCSP are also sent unprotected.

8.2.9 Other Services

The CMA may perform other tasks than the ones already mentioned. In some environments it may be responsible for issuing indirect CRLs and function therefore as a revocation authority. It may be responsible for validation services. Furthermore it may be used for supporting the "Key Information Service" mode of the XKMS protocol that locates public keys (and other services) on behalf of clients [W3C01, Sec. 1.4]. It may also be responsible for organising services required by the trust center itself. For example this is the error handling for requests that could not be processed (from the CMA or other trust center components).

8.3 Certificate Management Plugins

There are many tasks that the CMA performs. New tasks will occur. Current tasks will be removed or substituted by others. The different types of request need processing according to their nature. The various virtual hosted issuers require different handling of their products. The CMA addresses this situation by defining the certificate management plugins (CerMaP).

A certificate management plugin is responsible for realising exactly one CMA task. For example the ArchiveCertificates CerMaP is the plugin that is used for archiving certificates issued from the CA. A collection of these plugins is needed for performing all necessary tasks for a CA product.

Every request that reaches the CMA is processed according to the desired workflow of each virtual host. The possible requests that can appear have been presented in Section 8.1.3. For the P11Request for example the set of the CMA tasks is:

1. Archive certificate.
2. Send a notification to the user.
3. Update the OCSP backend and notify the OCSP server.
4. Publish the certificate on the LDAP.

For each of these tasks a CerMaP exists. A CerMaP is a component that delivers a certain service and realises a special functionality upon an input request. This input request is an ITP message. This ITP message contains certificates, CRLs, and other products. Each plugin extracts the necessary data from the ITP message and performs its tasks. The ArchiveCertificate plugin extracts a certificate from the ITP message, writes it on the filesystem, and stores it additionally in the database. A plugin may optionally add data to the input file.

If new functionality is needed, for example for realising a new protocol, only a new plugin is implemented, tested, and integrated into the CMA.

The CerMaPs are designed according to the template design pattern described in [GHJV95, Chap. 5] and [Met02, Chap. 21]. This allows the subclasses to realise the tasks of each CerMaP while the superclass provides the implementation of tasks that do not relate to the concrete services of each CerMaP. In Figure 8.2 the class diagram for the design of two CerMaPs can be found. The first is the UpdateLDAP plugin that is responsible for publishing certificates and CRLs to an LDAP directory. The second is the ArchiveCertificate plugin that archives certificates.

Each plugin implements the abstract methods *process* and *verify*. The process method is the realisation of the services of the CerMaP. Its input is an ITP message (encoded as a binary object) or a Java Wrapper for this message called the Product. In this method all tasks that a plugin performs are realised. The verify method checks whether this plugin has successfully performed its tasks. This is a very important function in some environments. In Section 6.3 we saw that when a certificate is published on the LDAP directory it must be checked

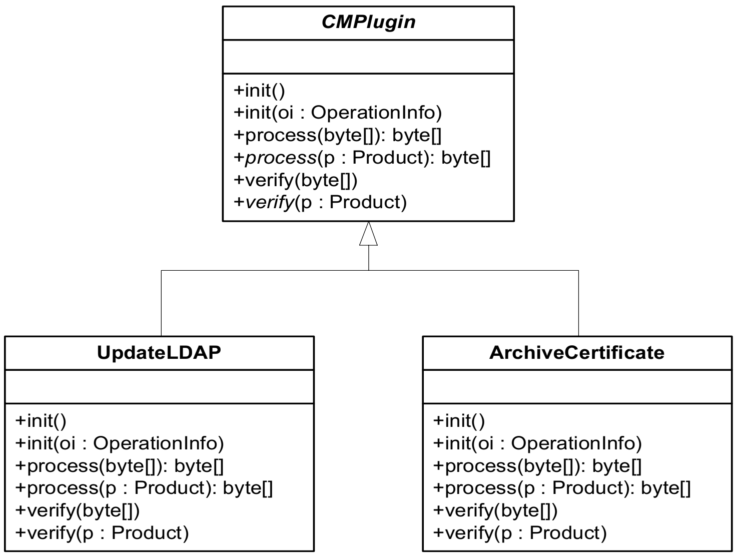

Figure 8.2: Class diagram for the certificate management plugins

whether the certificate has been published on the directory or not. While the actual publication takes place in the process method, the checking whether the publication was successful or not takes place in the verify method. The init methods offers a way to initialise each plugin. If initialisation is needed, then each plugin overwrites this method according to its needs.

All services of the CMA, like archiving, delivery, or publication have been implemented in Java as CerMaPs.

Planz has introduced the operators in [Pla02]. Operators function like the CerMaPs. The differences are that the operators are not able to process ITP messages and they do not specify the verify method. Moreover they are processing products that contain information about how they are processed. The daemon reads this information in order to steer the products.

8.3.1 Advantages

The use of ITP as the input of the certificate management plugins offers certain advantages. Since the input format is known to all applications, PKI developers can build arbitrary plugins. The type of each plugin and the services that it offers may vary according to the concrete scenario. Plugins that have already been implemented can be reused if they offer services that are needed in an installation. A library of interoperable plugins can be created that offers certificate management functionalities. Moreover, these plugins can be written in any programming language.

The format of the ITP messages also has several advantages compared to other representations. For example one possible representation is by using Java objects. The input data in this case is a serializable Java object. This object is used as a container. All necessary data is placed in a Java object through set methods, and it is accessible through get methods. An internal representation is used for storing the data. Such inputs cannot be processed from another component that is not aware of this representation. Other components, written in C for example, are not able to process such representations and a converter must be written. Furthermore deserialization problems may occur among different Java versions.

ITP messages can be exchanged among different trust centers or installations. This allows that the CerMaP inputs messages can be easily transferred to another instance and be further processed. Therefore some of the remaining processing steps for a request may be delegated or "outsourced". This format also enables the distribution of those requests among different instances.

The ITP messages can be signed and encrypted. This will support signature verification of the message from a CerMaP. According to the policy, if the verification fails the CerMaP may not process this message. Additionally, if a CerMaP inserts sensitive data in a message it may encrypt this part of the message. Moreover the messages exchanged among the CerMaPs are human readable. This eases debugging, understanding errors that may happen, or the manual processing of a product from a human administrator.

8.3.2 Workflow

Once a request reaches the CMA (in the PKCS#7 or ITP format) the importer extracts the data and creates an ITP message that contains this data. The ITP message roams from one CerMaP to another. Each CerMaP delivers its services for this message. After a message has been processed by all necessary CerMaPs it is stored in the database as a finished request. An activity diagram of this behaviour is shown in Figure 8.3 for a P11Request. The ITP message is unsigned.

Once a request reaches the CMA, the importer places every product in a database. At this point the importer has completed its services. The workflow for the product is controlled by the daemon. The first task of the daemon is to find products in the database that must be processed. It then extracts the product from the database. At this point it may additionally verify the signature on the product if this is signed. It checks which is the next plugin that this product should be inserted into and requests from this plugin to process this product. As soon as the plugin has finished processing a product the daemon may require that it also verifies the result of the processing. That is to check whether the task has been successfully performed or not. After that the product is placed once again in the database and is ready to be processed again in the next step. When all steps for this product are finished, this product is archived in the database (see also Figure 8.3).

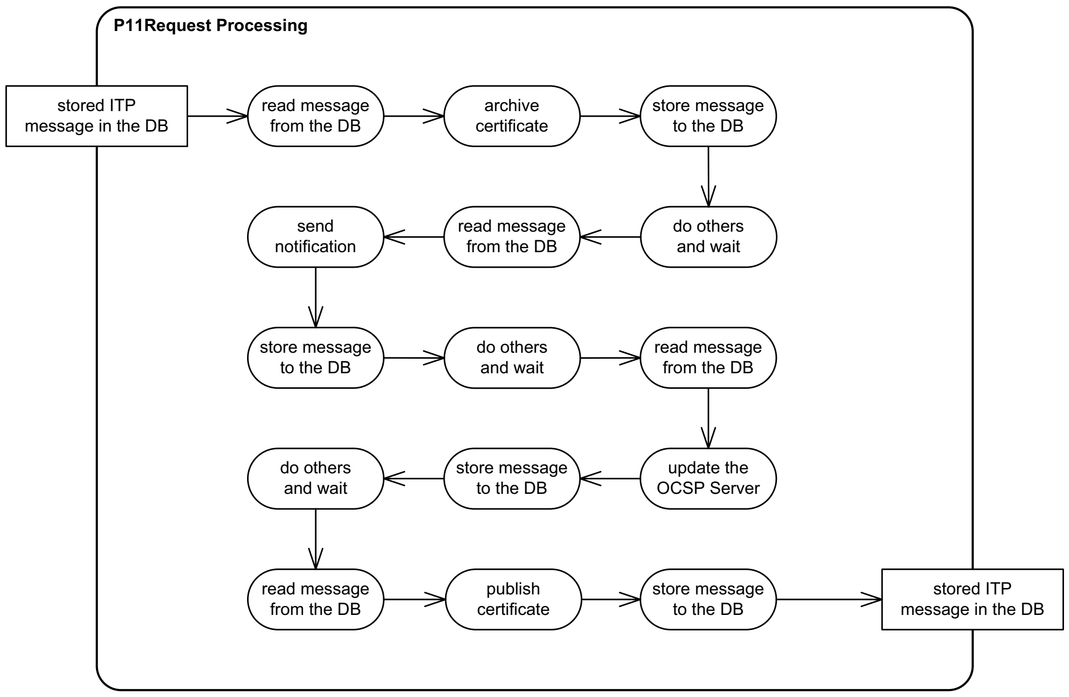

Figure 8.3: Activity diagram of the processing of a P11Request

The ITP messages that serve the communication among the CerMaPs contain only PKI related data. The daemon organises the workflow. This has the advantage that these products can be processed in several ways. Their processing can be parallelised for example. The products may be distributed among different computers. In addition, other applications do not need to be aware of the processing representation contained in the products. Therefore the new type of products can be used in general certificate management frameworks. If the messages among the CerMaPs contain information about how they must be processed these advantages do not exist anymore. Such information is for example the next operation for this product or the time that it must be processed.

This process allows to store the intermediate states of a request. Therefore if the CMA application is stopped, an error takes place, or anything unusual that will influence the application happens it is possible to continue the request from this point of failure.

8.4 Virtual Hosting

The various virtual hosted issuers in a trust center installation have different requirements on how their products are processed. As soon as an issuer has created a product, this is delivered to the CMA. Then the CMA processes this product according to the needs of each host. For addressing this situation the CMA is able to process products for the various virtual hosts and realise the desired certificate management workflow for each of them.

The realisation of the certificate management workflow of a product and the fulfillment of the CMA tasks is done by a sequence of CerMaPs. This regards one virtual host. This process is repeated for each host in order to achieve the virtual hosting. Therefore two goals are accomplished. First to be able to realise different certificate management workflow processes and second that this is done for every virtual host.

In Listing 8.1 an example of a configuration file is found. For implementing this scheme such a configuration file is created for every host. This configuration file contains the types of request that every host is able to process, the CerMaPs that are needed for each request, and configuration parameters for each CerMaP.

```
<host>

  <profile name="P12Request">
    <operation type="ArchiveCertificates">
      <param key="outputDirectory">filesystem path</param>
      <param key="useDB">true</param>
      <!-- other parameters -->
    </operation>
    <operation type="UpdateOCSPBackend">
      <param key="host">LDAP URL</param>
      <param key="port">389</param>
      <!-- other parameters -->
    </operation>
    <operation type="KeyBackup">
      <param key="useDB">true</param>
      <!-- other parameters -->
    </operation>
    <operation type="UpdateLDAP">
      <!-- parameters -->
    </operation>
    <operation type="DeliverPKCS12">
      <!-- parameters -->
    </operation>
  </profile>

  <profile name="P11Request">
    <operation type="ArchiveCertificates">
      <!-- parameters -->
    </operation>
    <operation type="MailNotification">
      <!-- parameters -->
    </operation>
    <operation type="UpdateOCSPBackend">
      <!-- parameters -->
    </operation>
    <operation type="UpdateLDAP">
      <param key="host">LDAP URL</param>
      <param key="usePoP">true</param>
      <!-- other parameters -->
    </operation>
  </profile>

</host>
```

Listing 8.1: Virtual host configuration file

For example there are five CerMaPs for the P12Request. These will realise the remaining tasks after the issuance of a certificate and a corresponding PKCS#12 file from a CA. Every operation, that denotes a CerMaP, has a set of parameters for configuring this CerMaP. In the case of UpdateLDAP CerMaP the host and the port where the LDAP server is running can be configured. In the case of P11Request the same CerMaP is used, but another parameter (the usePoP) configures this CerMaP to use the PoP scheme described in [KLW04a].

Therefore, for one virtual host the type of requests that are supported, the CerMaPs for each of them as well as specific parameters for each CerMaP can be configured. By replications of these files many virtual hosts can be supported. These files are configured and adjusted according to the certificate management processes of each host.

Similar format for such configuration files is used in [Pla02]. These configuration files are issuer dependent instead of host dependent. Since they are issuer dependent every instance that issues certificates or CRLs possesses such a configuration file. But one virtual host may have more than one issuer. This could be for example one organisation with a CA that issues certificates and a revocation authority that issues CRLs. In this case both configurations are identical providing redundancy and difficult management and maintenance of the installation. These drawbacks are not met in this design.

The situation becomes even more difficult to manage if an organisation issues cross certificates among internal CAs (intra-domain cross certification). As soon as the new CA starts to operate, a new configuration file is created and placed in the CMA configuration for the certificate management purposes of this new CA. Therefore, first a unilateral cross certification must be performed (the older CA certifies the new one). Afterwards the CMA is configured for the new host. After this step the CMA can start operating for the new CA. This means that the new CA cannot issue a cross certificate to the older one (bilateral cross certification) before the CMA is configured with the new CA. In addition as the number of cross certified CAs increases, the number of virtual CAs increases, too. The administration of such a procedure is time and resources consuming and contradicts the idea of flexible certificate management.

In the design every host has its own configuration file. This design does not exclude the possibility of every issuer having its own configuration file. If every issuer inside an organisation has different operational requirements, then the CMA can be configured to operate for this issuer by assuming that it is a virtual host. In this case a configuration file for every issuer is present.

8.5 Importers

The input formats that trigger the CMA vary. Different certificate management formats exist. We have discussed them in Section 5.2. CMA supports the PKCS#7 format and the ITP messages. When a PKCS#7 message enters the CMA, it is imported from the importer responsible for the PKCS#7 format. We will call this the PKCS7Importer. The case of ITP (ITPImporter) is similar.

The mentioned importers differ. The PKCS7Importer is able to verify signatures based on the PKCS#7 format, understand ASN.1 structures, and deserialize Java objects. The ITPImporter is able to verify XML signatures, understand and parse XML, and extract XML data. However they both implement the same process. That is to import a certificate management message, check whether it is valid, and export an ITP message that is used as the input to the CerMaPs.

In addition the importing mechanism is able to be extended. This is because other formats exist (like CMP, which is rarely used however), existing formats may be updated (for example ITP v2.0), and new protocols may appear.

In order to meet all these requirement we use the strategy design pattern for the importers of the CMA. For more on this design pattern see [GHJV95, Chap. 5] and [Met02, Chap. 23]. In Figure 8.4 the class diagram of the importing part of the CMA can be found. This design allows the CMAImporter interface to declare which methods must be implemented from each importer (since they must implement the same process). The subordinate classes offer the concrete implementation. For every mechanism a class is needed. When new mechanisms appear only a new class needs to be implemented.

One alternative is that every virtual host has its own importer. But then the number of importers is increasing with the number of hosts. Every importer is working independently trying to import a message headed for the corresponding host. The drawback of this approach is in the case of a message containing requests for more than one host. In order to import such requests complicated logic is implemented. Moreover, the importing of requests is a process of the CMA rather than of the various hosts. The approach of one importer for each message format, that imports all requests for all hosts, does not have these drawbacks.

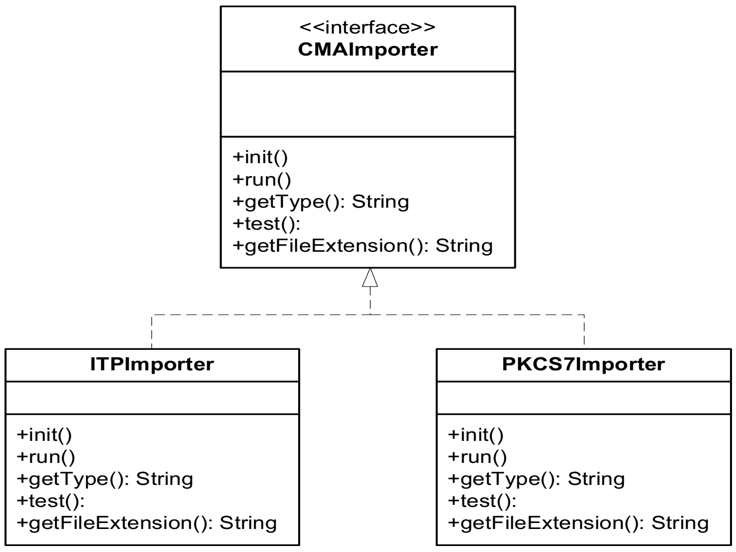

Figure 8.4: Class diagram for the importers

8.6 Low Level Description

In this Section we describe the functioning of the CMA in a lower level. We discuss the design and the classes that implement the functions of the CMA.

The CMA starts as a daemon. The class *CMA* is responsible for this. This class possesses the *init* method for realising the initialisation tasks of the CMA. In the initialisation phase the configuration of the CMA is read. A helper class for this purpose is the *XMLConfigurationParser*. This class reads the configuration file of the CMA. This file is in XML format. Also logging is initiated. The class *Logger* resolves the logging. When initialisation is done the CMA starts by calling the *start* method. From this moment the CMA is ready to deliver its services.

While the CMA runs it listens for incoming requests in order to further process them. This is done by two importers. The first one is the *ITPImporter*. It is responsible for importing ITP messages. The second one is the *PKCS7Importer*. It is assigned the task of importing PKCS#7 encoded requests. Both classes implement the *CMAImporter* interface. When a request is imported by one of the importers it is verified for its validity.

All requests that reach the CMA are signed from the PKI component that has created the request. In order to verify the signature the *CertificateValidator* class is used. There are two ways to create objects of this class. Either by providing a filesystem directory on which all trusted certificates are found, or by setting one *KeyStore* that contains all trusted certificates. Verification of the incoming requests is always performed. The requests are additionally archived by using the *archive* method.

In order to demonstrate the functioning of the CMA we suppose that a P11Request (see Section 8.1.3) reaches the CMA. It is encoded as an ITP message. In this case the ITPImporter imports the request by using the private method *importXMLFile*. All data is extracted from the ITP encoded message and is stored in a Java object of the type *Product*. The Product is the Java representation of the ITP message that roams from one plugin to the next one. It contains useful getters and setters method for the typical PKI products like certificates, CRLs, software PSEs, or entity related data. The relevant data for this request is: the certificate, the virtual host's name that has produced the request, the end user's data like the email address, and the DN of the entry on the LDAP server.

The product is written in the database. The responsible class for this task is the *DB*. It contains methods for storing and retrieving the ITP message that is represented as a Product. It further contains methods for storing and retrieving other PKI objects like certificates, CRLs, or PSEs (*X509Certificate*, *X509CRL*, and *PFX* respectively) from a database.

After all incoming requests have been imported by the CMA the products are read from the database. Then they are processed according to the workflow configured for the virtual host that has created the request. The class that holds information about each virtual host is the *Client*.[3] Each product, depending on its virtual host and type of request, will be processed differently. The P11Request goes through four plugins. The activity diagram of this behaviour for this request is shown in Figure 8.3.

There are four CerMaPs involved in this request. Each implements the abstract class *CMPlugin*. The *ArchiveCertificate* plugin processes the request. It extracts the certificate from the ITP message and stores it in the database by using the *DB* class. It further archives it in the filesystem. For this the standard Java class *FileOutputStream* is used.

The *MailNotification* plugin extracts the certificate from the ITP message. It further extracts the email address of the certificate user. This parameter is set from the RA into the initial request. It then sends an email to the user with the certificate attached. The *javax.mail* Java classes are used for this purpose.[4]

The *UpdateOCSPBackend* plugin extracts the certificate and sends it to the OCSP server for notifying it for the new certificate. The class *OCSPClient* is used for realising the connection to the OCSP.

The *UpdateLDAP* plugin extracts the certificate and publishes it to the entry of its user on the LDAP server. This parameter is set from the RA. The value of this parameter is the DN of the entry. The *LDAPConnection* class is responsible for realising the publication of the certificate. The functionality of the class is

[3]The name Client depicts each virtual host as a client, whom the CMA serves by delivering certain services.

[4]Available at `http://java.sun.com/products/javamail/` (date of access 14.01.2007).

discussed in Section 8.7.

8.7 LDAP API for Supporting PKI

The publishing of PKI information is one of the CMA tasks. Depending on the installation the requirements related to PKI publishing change significantly. Compared to the other tasks of the CMA, the publishing tasks are more demanding because they have to be customised in every installation. This is because different organisations use the directory for different reasons. For example in [LKWB05] the LDAP directories are used for registration and supporting PoP. Moreover, the LDAP directories can be used for supporting other services like OCSP.

In order to address this situation, we have designed and implemented a library that realises diverse publishing functions. It is implemented in Java using the JNDI technology [SUNa]. This library is used by the CMA for publishing certificates, CRLs, and other data. It is possible to use different authentication mechanisms and connections secured with SSL. It has been tested with diverse LDAP servers like OpenLDAP [Ope04], Novell NDS, or the SUN directory server. It can be used as a general purpose API from different trust centers for functions that are related to LDAP publishing. It can also be used at the side of PKI clients for retrieving CRLs and certificates. Therefore it covers almost all aspects of LDAP based publishing of PKI information. In the following we see what are the features of this library:

- Authentication:
  - Simple Authentication.
  - Simple Authentication over SSL/TLS.
  - SASL based authentication.
- Secure communication over SSL/TLS.
- Virtual hosting.
- Configurable schemata.
- Support for organisational procedures that are not related to PKI information.
- Support for the PoP scheme and the PSE software delivery scheme. These schemes are described in [KLW04a].
- Support for registration of PKI users according to the scheme described in [LKWB05].

- Supports publication of:
  - user and CA certificates.
  - CRLs.
  - delta-CRLs.
  - ARLs.
  - cross certificates.
- Different publication possibilities based on existent entries, creating new entries based on the certificate's DN or according to the serial number (each certificate is published in its own entry).
- User administration.
- Verification of publication.
- Support for an LDAP based OCSP backend.
- Certificate removal on revocation.
- Client API for searching, locating, and retrieving certificates and CRLs.

8.8 Technologies Used

The CMA needs some tools for addressing certain technical requirements. These requirements are for example the platform independency, the connection to databases or LDAP directories, the logging, or the cryptographic algorithms. We discuss the technologies that we have used for implementing the CMA.

8.8.1 Java

The CMA is implemented in Java. Java as a programming language offers great flexibility due to its platform independency. The developed applications can be run on different operating systems without requiring that the code or parts of it must be changed or be platform specific. This offers an easy development of applications. The CMA has been successfully installed and operated on Windows, Linux, and Solaris operating systems.

Moreover security is a design goal of Java. Security issues are addressed in Java in various ways. First on the platform itself. There are features like bytecode verification which checks for example type correctness or that no stack overflows or underflows occur. Another feature is the secure class loading needed for example in Java applets. But security in Java has other aspects, too. This is the support of cryptographic operations.

There is a framework in Java for supporting cryptographic functions called the JCA/JCE (Java Cryptography Architecture/Extensions). Within this framework various cryptographic algorithms can be implemented and be used by different applications. It offers an API for using these algorithms as well as an SPI (service provider interface) for implementing them. It is flexible enough for allowing an easy exchange of the cryptographic algorithms. The FlexiProvider implements various cryptographic functions using the JCA/JCE framework.[5] The CMA employs the FlexiProvider mostly for calculating hash functions and signature verifications.

Java supports also data and functions related to PKI. It offers classes for representing X.509 certificates, X.509 CRLs, software PSEs in the PKCS#12 format or in the JKS/JCEKS format. It enables verification of certificates, construction of certification and validation paths, or parsing of CRLs and certificates for extracting data among other functions. These features are constantly used by the CMA.

Moreover, security is addressed in Java on the network security layer by implementing SSL/TLS. This implementation can be used with various protocols like HTTP or LDAP. In addition Java offers the JAAS (Java Authentication and Authorization Service) framework for authentication and authorisation purposes.

Apart from the strong support for security services Java supports other important functions needed by the CMA. One is for example the access to relational databases. The technology in Java for realising this is the JDBC (Java Database Connectivity). It offers a flexible method for communicating with different databases. Moreover in Section 8.7 we saw that CMA is using the JNDI technology of Java for accessing LDAP directories. Lastly, since version 1.4 of Java a logging mechanism is also available that we employ for the logging purposes of the CMA.

Java is also XML aware. It has lots of different frameworks for accessing and parsing XML structures. This is needed for parsing XML files and being able to process ITP messages.

8.8.2 XML

The use of XML in the CMA is needed for configuration purposes. All configuration files of the CMA are in the XML format. This enables a human readable format of the configuration which can also be structured in a hierarchical representation. The parameters regarding the database access like the host IP, or the port where the database is running are configuration parameters that are configured in such a file.

XML files can be processed with comfortable and easy to use editors and therefore it is easy to create or update such a file. In addition the XML files can

[5]For more information see `www.flexiprovider.de` (date of access 30.11.2005). This software is open source and available for download.

be checked for syntactic correctness using DTD or XML schema. Moreover, the configuration files can be signed for providing authenticity and for identifying changes in the configuration, which may have security consequences. Furthermore, parameters like passwords or other sensitive data can be encrypted using XML encryption.

The configuration files for enabling virtual hosting are also in an XML format. Such a file can be seen in Listing 8.1.

XML is also used by the CMA for importing ITP messages. The ITP messages must be parsed, verified, and processed according to XML. The input requests of the certificate management plugins are also ITP messages and therefore XML is needed in this case, too. XML offers a platform independent portable data format. This is needed for the communication among various components of a trust center and the CerMaPs.

8.9 Analysis

We see that the design and implementation of the CMA has met the requirements listed in Section 4.2.2 as well as that it completes the core tasks assigned to it that are listed in Section 4.2.3.

Tasks

The tasks of the CMA are realised as certificate management plugins. Detailed description of those tasks can be found in Section 8.2. The minimum set of tasks that the CMA performs has been realised.

Virtual Hosting

This requirement is addressed by the presence of a special configuration file for each of the virtual hosts. This file configures, independently for each host, which plugins will be used for realising the certificate management functions of a host. Lots of different virtual hosts can be operated at the same time. New virtual hosts can be added or older ones can be removed by this mechanism. In the Technical University of Darmstadt a trust center with two virtual hosts has been installed. For more on this installation see Section 10.3.1.

Configurability

Many of the CMA operating parameters can be configured. The number of virtual hosts that the CMA is operating for, the types of request, as well as the necessary tasks regarding certificate management are defined as configuration parameters rather than components integrated in software.

The number of virtual hosts is defined by a configuration file for each of them. Inside this file the different types of request that the CMA is able to process is also determined. Moreover for each request it is possible to declare the required steps for its processing. This is done by configuring which certificate management plugins are associated with this type of request.

Configurability of all these properties is important in case the CMA is evaluated (according to CC for example). If the functionality of the CMA is configured as described above and if changes are required, then the CMA avoids a new evaluation. For example, if a new virtual host is added, then there is no need to re-evaluate the CMA. In addition if a new evaluation cannot be avoided, the CMA core programm (the daemon and the organisation of the workflow) can be evaluated once and only the new or updated plugins have to be (re)evaluated.

Flexibility

The CMA can be installed in many operating systems. This is due to its Java based implementation. Java offers platform independency and therefore flexible integration of the CMA in current operating systems. Moreover, Java provides other advantages, too, like the JCA/JCE framework. This framework offers cryptographic services in a flexible way since the same API can be used for different cryptographic algorithms.

The tasks and services of the CMA can be extended using the flexible mechanism of the certificate management plugins. These plugins can be added, removed, or updated for realising new functionality. This functionality will be provided in the appropriate methods of every subclass, enabling an easy and modular implementation of the plugins.

Moreover the plugin mechanism allows easy and flexible integration of the CMA to existent processes. The plugins should be implemented in a way that fits the current practice in an organisation. The benefit of this is that the CMA and the PKI will be not considered a new process but just an extension to the current practice in an organisation. A good example of such integration is the PKI installed at the Technical University of Darmstadt. We provide more details in Section 10.3.1.

Scalability

The CMA design addresses the scalability problem by supporting distribution of the tasks. This is due to the plugin design and the ITP messages. In order to fully process a request, this is processed by various certificate management plugins. These can be distributed among various resources. Diverse techniques can be used for distributing the tasks like multi-threading, and network based distributing techniques like RMI, CORBA, and Jini.

The messages exchanged among the distributed services are ITP messages.

This format is portable among platforms and programming languages. Applications that are aware of this format can process such messages. This allows the distributed services to be implemented independent of any specific technology providing therefore a variety of possible implementations.

Availability

The availability of the CMA is achieved through data persistence and distributability. As soon as a request is imported it is stored in the database. Additionally, it is archived in the filesystem. Therefore requests will be always available. After being stored in the database the request is forwarded to the first certificate management plugin and after that it is stored again in the database. In case of failure (for example the CMA application is stopped), the request remains in the database and it can be further processed when the application starts over. Alternatively, a second application may process the request if the services of the CMA are distributed.

Standards Conformity

The CMA conforms to the PKIX specifications as far as certificates and CRLs concerning [HPFS02]. It can only process such types of product on behalf of an issuer. It further processes known formats of messages like the PKCS#7 for certificate management purposes or PKCS#12 as a personal security environment. It further conforms to other specifications like the ISIS-MTT [TT] which provides a profile for certificates, CRLs, and PKI services. This specification is used mostly in Germany. Moreover, it uses standard techniques for publishing certificates and CRLs on LDAP directories. The LDAP based publishing has been already described in Section 6.2.

Security

The security of the CMA as a design goal of its specification will be discussed in Chapter 9.

Chapter 9

Security of the CMA

The CMA is part of a trust center software which has services like registration, certification, publication, and OCSP among others. It is important to evaluate the security of the CMA since it is a component of the PKI. This is because for evaluating the security of the infrastructure both the whole infrastructure and its components are evaluated.

Security is a design goal for the CMA. If the CMA is secure, then its services and tasks that support PKI functions are performed as desired. We see which are the threats to the CMA. We describe its security requirements and security functions. Further, we provide the design for realising these security functions.

9.1 CMA Security Functions

We present a security analysis for the CMA. In this analysis we describe the security functions of the CMA. These security functions are properly addressed and implemented. In one case we briefly show the association of a security function to the security functional components defined in the second document of the Common Criteria [Cri05a].

The CMA has different tasks in different environments. These tasks also have different security requirements. For example, if the CMA is responsible for issuing indirect CRLs for various issuers, then the creation of the signature on the CRL sets new security requirements that do not appear when the CMA is only managing certificates. In order to evaluate the security of the CMA one concrete environment and context is chosen. We choose the SigG context because a security evaluation of the technical components that are used is mandatory in this context [Leg01b]. Moreover this context has strict requirements. Due to this choice issues like the activation of certificates are addressed. Other issues, like key backup, do not need to be addressed since they never appear in the SigG context.

CMA has the lowest security requirements of all other trust center components. This is due to the CMA nature, that it is to process already signed products. Therefore, attacks like existential forgery or unauthorised use of signature private keys do not threaten it and they are not addressed at all at the security analysis.

We present the attackers and the threats of the CMA. We see its security functions that protect it against these threats. We discuss the CMA modular design for realising these functions. Parts of the description of the threats and security functions are extracted from [Kar03] and [LLR$^+$03].

9.1.1 Attackers and Threats

There are various attackers for the CMA. The first group of attackers belongs to the authorised users of the CMA. These are the CMA operators, the CMA administrators, and the system administrators. They CMA operators are allowed to start and stop the CMA. The CMA administrators are responsible for administrative tasks regarding the CMA (for example for evaluating the log files). The system administrators are responsible for administrating the machines on which the CMA runs or for configuring its network connections. The second group of attackers is the unauthorised users of the CMA. These are all the users that are in no way authorised to interact with the CMA. The attackers perform attacks intentionally or unintentionally.

The attackers pose different threats to the CMA. These threats are listed below. Threats related to the environment of the CMA (for example infection by a virus) cannot be addressed by the CMA itself and therefore they are not listed.

Data Manipulation and Disclosure

The relevant to the CMA data is the data that reach the CMA and the data that it processes. Such data is for example a certificate or a certificate revocation list.

This data can be manipulated. In this case the CMA processes invalid data. For example it can publish an older CRL or notify the OCSP server with a wrong certificate.

The data can be disclosed. Such data for example is the activation password of a user. If this password is disclosed, then it is possible to activate the user's certificate. The passwords that the CMA uses for publishing the certificates and the CRLs are also such data.

The security functions DP, IP, TCC, PV, OBP, OIBP, and TCI (see next Sections) protect the CMA against this family of threats. The security functions SL, OI, LMF, and LE support the above mentioned security functions.

Revocation Threats

The revocation of one certificate may never reach the correct OCSP server. If this happens then the correct OCSP server will provide wrong answers regarding the status of a certificate. Likewise a CRL may never be published on the correct public directory. Revocation information is in this case wrong.

The security functions DP, IP, TCC, PV, CA, OBP, OIBP, and TCI (see next Sections) protect the CMA against this family of threats. The security functions SL, OI, LMF, and LE support the above mentioned security functions.

Activation Threats

One threat for the CMA is to fail to notify the correct OCSP server for a new certificate. If this happens the certificate will not be verifiable. Therefore, correct information about its status cannot be obtained. If the CMA does not publish a certificate on the correct public directory then a certificate will not be available and cannot be used for verifying signatures.

An additional threat is to publish a certificate on the public directory and make it available, although this certificate is only verifiable.

The security functions DP, IP, TCC, PV, CA, OBP, OIBP, and TCI (see next Sections) protect the CMA against this family of threats. The security functions SL, OI, LMF, and LE support the above mentioned security functions.

9.1.2 Data Protection (DP)

The data that is found in external resources like a database or a directory are protected against manipulation. This will protect the data itself as well as other operations or processes that rely on their integrity. These will prevent attacks like the deleting of a CRL or its replacement for an older one in the directory.

For achieving this, only authorised users are able to establish a connection to the database used in the system as a backend. Likewise, only authorised users are able to modify and write data on the directory.[1] Therefore, some data is protected for confidentiality. This is especially the passwords used for authentication purposes. These could be the password for a directory user that has write access to the directory or the password of a database user who is allowed to connect to the database.

9.1.3 Input Protection (IP)

The data that is transported to the CMA are protected for integrity and authenticity. This data can be the certificates and the CRLs that come from another

[1]The PKI clients on the contrary can search or read data. This is important for locating certificates or CRLs and getting status information. This data will be used during a signature verification.

trust center component, usually the CA. If this data is not protected or their protection fails validation, then the CMA does not process them. This will prevent that the CMA processes invalid data.

9.1.4 Trusted Communication Channel to the Directory (TCC)

A trusted channel is established between the CMA and the external publishing directories like LDAP. This is a requirement for the secure transport of data like certificates and CRLs. These could not then, for example, be substituted during transport. The integrity and authenticity of this communication is important and if this fails to be established the communication is aborted.

9.1.5 Publishing Verification (PV)

The publishing of information to a directory external to the CMA, like the LDAP directories used for publishing PKI information, is verified for success. This means that when for example a certificate is published to the directory, the CMA checks whether the published certificate is identical to the one that the CMA tried to publish. This is performed also for other data like certificate revocation lists. If an unsuccessful operation is detected, then appropriate messages are created that will signalise this.

For this security function we provide a further analysis based on the original CC security functional components. This security function addresses and fulfils the FDP_UIT.1.1 and FDP_UIT.1.2 [Cri05a, Sec. 11.13]. The FDP_UIT.1.1 suggests that user data during transmission or receival is protected from modification, deletion, insertion, or replay. If nevertheless such an action takes place, FDP_UIT.1.2 suggests that this has to be detected. These components have also dependencies to other components. These are the FDP_ACC.1 or FDP_IFC.1 and the FTP_ITC.1 or FTP_TRP.1. These are also addressed for a complete evaluation. FDP_ACC.1 is fulfilled by the OBP, OIBP, and TCI (see Sections 9.1.7, 9.1.8, and 9.1.9). FTP_TRP.1 is fulfilled from CA (Certificate Activation, see Section 9.1.6).

9.1.6 Certificate Activation (CA)

When a certificate is activated, this becomes immediately verifiable (see Section 6.3). For making a certificate verifiable an OCSP server is used and therefore it is notified about the new certificate. The notification is done via a trusted channel and the correctness of this operation is examined.

A certificate is published for becoming available (see Section 6.3) only after an activation and only if it is already verifiable. Then, it is published on a directory like LDAP. This is performed over a trusted channel. It is checked whether the publication was successful or not. That is, it is examined whether the correct certificate has been published and is available for standard PKI clients.

9.1.7 OCSP Backend Protection (OBP)

The backend of the OCSP is a directory. From this trusted directory the OCSP gets all the information it needs in order to give signed answers to certificate status queries. Therefore this information is protected. Otherwise, the answers of the OCSP server may be wrong (if certificates or CRLs are deleted for example).

The connection to the directory is performed over a trusted channel. For providing answers to client queries, the OCSP server checks the directory for determining whether a certificate has been issued or not. In addition, it checks the directory for deciding whether a certificate is revoked or not from the most recent CRL. The status of the data is continuous, therefore only new information may be added to the directory. For example new certificates can be stored but it is not allowed to delete any certificates. Likewise, newer CRLs that substitute the older ones contain all previously revoked certificates.

9.1.8 Other Instances Backend Protection (OIBP)

Other instances may need data located in a directory in order to perform their services and tasks. Such instance for example is a revocation authority. In this case the access of data from the directory is performed over a trusted channel. The status of the data is continuous. The correct and secure functioning of the various instances will be achieved by protecting the data on the directory.

9.1.9 Trusted Channel Input (TCI)

The input data that trigger the CMA is delivered over a trusted channel. Only if such a channel exists the CMA accepts this data and processes them further. Thus, the CMA never processes data from a source that is not trusted. Such a source could compromise the CMA. Depending on the kind of data different actions are performed.

CRLs are published as soon as they become available to the CMA without any delay. It is checked whether their publication was successful or not. This includes both the updating of the OCSP backend and that of the external publishing directories. Therefore, clients are provided with the most recent revocation information. All of the above is performed over a trusted channel. The activation of certificates is done over a trusted channel, too. After activation all certificates become verifiable and some of them available. The information regarding the availability of a certificate is also protected.

9.1.10 Start of Logging (SL)

When CMA starts up, logging is initiated. It is not possible to operate without the logging mechanism. This means that if the CMA is unable to log it terminates and if the CMA runs it is able to log. This will ensure the monitoring and auditing of the application.

9.1.11 Operators Identification (OI)

The identities of the operators that start the CMA are logged. In addition, both the successful and the unsuccessful trials to authenticate as an operator and start the system are logged. The CMA verifies the identities of the operators and checks their authorisation. Only authorised operators are allowed to start the CMA. Therefore, the CMA cannot be started and provide services if unauthorised persons want to use it.

9.1.12 Log Messages Format (LMF)

For achieving auditing and monitoring, all log messages have a special format. A special process identity (PID) is given to the CMA. This is logged with every message. Other information that is logged is the date and time of an event, and the identity of the operators. In addition, the type of the event is logged. This means that a statement whether the message is for example informational, warning, or fatal is present. Lastly, the success or failure of this event is also logged.

9.1.13 Loggable Events (LE)

There is a family of events that are logged. All these events are logged with the appropriate format. For example in the case of publishing to the directory it is logged that the CMA tries to publish a certificate to the directory as well as whether this publication was successful or not. Thereby monitoring and auditing of the application is enabled.

9.2 CMA Modules

In order to effectively evaluate the security of the CMA, it is divided into modular parts. This is also a requirement from the Common Criteria methodology, for the low level description of a component [Cri05b, Sec. 15.5]. These modular parts may or may not be used for enforcing security functions. The CMA is divided into eight modules. These are the:

1. *Daemon* The Daemon is the software daemon of the CMA. It starts and stops the CMA. It also organises the workflow of the requests that the CMA processes.

2. *Importer* The Importer imports the input files that trigger the CMA. These are created by the CA. It reads them from the filesystem, verifies the signature calculated over this data and extracts all necessary information.

3. *Product Delivery* This module delivers the products of a CA. Such products are the certificates and the revocation lists. In case other products exist this module is also responsible for delivering them.

4. *Directory* This module resolves the necessary communication with a directory which is an LDAP server. This module can be used by various trust center components for realising their required functions that are related to LDAP.

5. *OCSP Client* This client sends special queries to an OCSP server in order to update its internal store with new certificates or new revocation information.

6. *RA Communicator* This module resolves the communication with an RA. There are different possibilities for realising this communication depending on the installation. These could be direct messages over a network like HTTP requests, or communication over a database, an LDAP directory, or the filesystem.

7. *Pass Sharing* This module is responsible for retrieving shared secrets. These secrets are shared among two or more participants. In a trust center installation these could be the administrators or the operators of the software. A secret could be a password for accessing certain resources (like an LDAP server or a relational database). Since a secret is shared, it is more difficult to extract it because at least two persons must collude. Therefore dual control is enabled.

8. *Logger* The Logger is responsible for logging all events and necessary data to the filesystem.

9.3 Security Design

9.3.1 Daemon

While the Daemon runs the logging is enabled. This is to meet SL. As soon as the daemon starts up it tries to write some information into the appropriate log file. If this is not possible the Java virtual machine (JVM) exits. Otherwise the

Daemon continues with its start up phase. The Logger supports this function and the Daemon is using this module.

Immediately after logging is enabled the Daemon requires that two operators decrypt the secrets that are needed for the correct functioning of the application. These are the passwords for connecting to the directory or the database. These secrets are stored encrypted and they cannot be disclosed without possession of the private keys involved in the secret sharing process. These passwords are known only to the CMA which can connect to the external sources that require the password based authentication. This meets DP. The decryption based on secret sharing is supported by the Pass Sharing module.

If the connection to the database is possible, this means that the password has been successfully decrypted. From this we can derive that the operators have successfully provided their own secrets (like password, or PIN, or biometric characteristics) and the logging of their identities is possible. If the connection to the database is impossible due to bad credentials, this means that decryption was wrong and therefore no trusted operators are found. Thereby OI is supported.

Furthermore, as long as the Daemon runs logging is enabled. When the application starts, the logging starts also. During the time the application runs, all loggable events can be logged. If this is impossible the application exits. Thus, LE is supported. This module is using the Logger module for realising these functions.

All messages have a special format as this is required by LMF. This will therefore meet LMF. The special format is supported by the Logger. The identities of the operators are provided by the Pass Sharing module and are logged, too.

9.3.2 Importer

The Importer reads files that are placed in a filesystem directory. These files conform either to the PKCS#7 or ITP format. All data from the CA is placed in a PKCS#7 container or an ITP message and is protected with a digital signature. The certificate that is used to verify the signature is a known trusted certificate to the CMA. If the signature is invalid this data is discarded, it is not further processed and a special alarm message is given. The CMA supports therefore the TCI and IP.

If the signature over the data is valid, the Importer extracts it. This data is serializable Java objects or XML data. From these objects other data is extracted like certificates or CRLs.

In the following cases the CMA does not process the files: a correct file is placed in a wrong directory. A wrong file, which does not conform to one of the expected formats, is placed in the correct directory. A correct file is placed in the correct directory but the signature is invalid.

For the importing procedures logging is enabled for supporting LE.

9.3.3 Product Delivery

Product Delivery writes the CA products on the disc. These are the certificates, CRLs, and other products that the CA produces. The attempt to write a product on the disc and the result of this attempt are logged. LE is supported.

9.3.4 Directory

The Directory module realises all necessary interfaces and connections to a directory. It is divided into two submodules. The Authorised and the Anonymous. The Authorised module is responsible for connecting to the directory with permission to alter its content. This means that this module has write access. The Anonymous module is used for connections that are not allowed to change the directory's content, namely it has only read access. In both modules all connections are secured with SSL. Thereby CA, OBP, OIBP, TCI, and TCC is supported. Both the server and the client require that an SSL secure connection is established between them. Failing to establish such a connection results to special log messages (LE) and termination of the CMA operation. At a minimum the directory server authenticates itself using public key techniques.

The Authorised Directory is used exclusively by the CMA since it is the only component allowed to alter data on the directory. In this module a directory user is required to authenticate to the directory. There are several authentication mechanisms to achieve this like password based or SSL client authentication. All these mechanisms rely on secrets like the password itself or the password protecting the keys for the client authentication. These secrets are protected as described in the Pass Sharing module.

The certificates and CRLs are published on the directory to support TCI. Publishing of a certificate takes place only after its activation. The Directory module publishes the activated certificate immediately on the directory to make it verifiable (since the directory is used as the OCSP backend). This is to support CA. The correctness of this action is examined through a special request to the OCSP server. This is described in the OCSP Client module. OBP and TCI is supported. The RA Communicator provides information about whether the certificate will be available or not. If the certificate must be available, it is published immediately on a directory that PKI clients can query for information. This is done only after a successful update of the OCSP server. This supports TCI and CA. The procedure for the CRL is similar. The difference is that the CRL is immediately published without any activation and without distinguishing between available CRLs or not. All CRLs are available.

The Anonymous Directory is the submodule that is responsible for reading data from the directory. It is used for checking whether a publication from the Authorised module has been successful or not. It fetches the newly published certificate or CRL from the directory and compares it with the one that the

Authorised module has published. If they do not match, it provides special notifications. Thereby PV is supported. This submodule can be used from any other instance that uses the same operations. An example for this is an OCSP server that uses a directory as its backend. It can use the existing functions of this submodule to get data from the directory.

9.3.5 OCSP Client

The OCSP Client sends special requests to a standard OCSP server for informing it for new certificates or revocations. After a certificate is activated this becomes immediately verifiable. An OCSP server is used for this purpose. For providing the server with the new information, the OCSP Client sends a query to the OCSP server that contains the activated certificate. Upon this request the server searches the backend for this certificate. If the certificate is found, it becomes a known certificate to the server. It then answers back to the OCSP Client. If the server's answer is "good" it denotes that the OCSP server has been successfully updated with the new information. In the case of CRL the query is about the last revoked certificate. The expected answer is "revoked". The result of these actions is logged by the Daemon (LE). With the above OBP and TCI is met.

9.3.6 RA Communicator

The RA Communicator is responsible for providing the activation of certificates. A special message is sent from the RA signalising which certificate is activated. The RA Communicator accepts this message and initiates the remaining tasks that are performed with the certificate. These are to enable the certificate to be verifiable and if it is desired also available. If the initiation does not take place this is logged (LE). This behaviour meets the CA. The messages from the RA are sent in a PKCS#7 or ITP format and they are protected for authenticity and integrity in order to meet TCI.

9.3.7 Pass Sharing

The Pass Sharing module uses the secret sharing technique proposed by Shamir [Sha79]. Particularly it uses a (k, n) threshold scheme. In this scheme at least k participants are required in order to reveal a secret. These participants are the operators.

The passwords of the directory and the database that the CMA is using are shared. The operators decrypt their shared secrets, in order to retrieve the passwords. The private keys and the corresponding certificates are located in a smart card. From these certificates the identities of the operators will be extracted. The passwords and identities are forwarded to the CMA. Thereby DP, OI, and LMF is met.

The Pass Sharing module is based on another module that is provided as an external library. The security properties of this library is also addressed in a similar approach as this of the CMA.

9.3.8 Logger

The Logger is logging events in a log file. These events are logged sequentially. If at any point logging is impossible the Logger stops the application that uses it. This also happens if the log file is deleted. Thus, SL is met. This module is required for logging an event, in a special format, and with the identity of the operators present. Therefore it supports OI, LMF, and LE.

9.3.9 Overview

In Table 9.1 we see an overview of the security functions of the CMA and which of its modules enforces them.

Security Function	**Enforcing Module(s)**
DP	Daemon, Pass Sharing
IP	Importer
TCC	Directory
PV	Directory
CA	Directory, RA Communicator
OBP	Directory, OCSP Client
OIBP	Directory
TCI	Directory, Importer, RA Communicator, OCSP Client
SL	Daemon, Logger
OI	Daemon, Logger, Pass Sharing
LMF	Daemon, Logger, Pass Sharing
LE	Daemon, Logger, Product Delivery, Importer, RA Communicator, Directory, OCSP Client

Table 9.1: SF enforcing modules

Chapter 10

CMA in a PKI Environment

The CMA offers solutions to the certificate management problems in a PKI. This authority is only a part of the whole PKI. We see its interaction with the other trust center components and its role inside the PKI. This Chapter shows the use of CMA in the PKI environment and examines its integration into it.

The CMA has already been used in many PKI installations. It is part of a trust center software called FlexiTrust. We introduce the FlexiTrust software. We see which are the goals of this software and present its architecture. We allocate the tasks of a trust center among the various components. We visit existent projects in which the software has been installed and see what are the services of the CMA in the PKI installed at the University of Darmstadt

10.1 FlexiTrust

FlexiTrust is a flexible trust center software developed at the Technical University of Darmstadt (TUD). Up to now there are several installations of FlexiTrust. Most prominent among them is the one at the German national root CA. Using the FlexiTrust software the root CA certificates as well as accredited certificates for certification service providers (CSP) are issued. FlexiTrust has been successfully evaluated in November 2003 according to CC. This was the version 3.0 of the software. It fulfils the SigG and SigV requirements [Zer03]. For more on this evaluation process see [WLK+05].

The goal of FlexiTrust is to achieve flexibility. This has two aspects. The first one is the flexibility of the cryptographic primitives. The underlying algorithms that are used can be easily exchanged following the newest developments in cryptography and cryptanalysis. The second one is flexibility in the installation. This regards the software itself which is not difficult to install across different platforms. Moreover the integration in existent and already established workflows in an organisation is possible. In addition, if new developments are necessary, then their integration in FlexiTrust is easily achieved. The major effort therefore is to implement the new features and not their integration in the software. Lastly, FlexiTrust is easy to operate. A rather complex installation at the TUD required less than a day training for the operators of the trust center. In Section 10.3.1 we will see in more details what are the tasks of the CMA in this installation.

10.1.1 The RA-KA-CMA model

FlexiTrust consists of three entities. The RA, the Key Authority (KA), and the CMA. A forth PKI entity is the end-entity. This does not resemble the design of a typical PKI as we saw it in Section 3.2.7. Reasoning for the presence of a KA is found in [WLK04]. The design and technical realisation of this authority is discussed in [WKLB05]. We have already seen which considerations and needs are present for the CMA.

This design has been proven in practice to have very good properties. Firstly, it addresses the three main different tasks of a trust center. That is the registration, certification with optional key generation, and certificate workflow management. Secondly, it is modular enough to enable an easy installation. For example the RA can be installed in a different computer than the KA. Thirdly, it provides security enhancements by enabling the KA to be offline.

In Figure 10.1 the design of the whole FlexiTrust infrastructure can be seen. The RA, KA, and CMA are inside the trust center. Messages can be exchanged between them (for example with ITP or PKCS#7). Under these components the cryptographic provider is found. This is the FlexiProvider. It is based on the JCA/JCE technology of Java and provides implementations of different cryptosystems. This is the basis for the flexibility of FlexiTrust in exchanging the cryptographic primitives.

10.2 Task Allocation

10.2.1 Registration Authority

The registration is located at the RA. It is the entrance of the end-entities to the trust center. The RA is responsible for verifying the data of the entity and initiating a request to the KA.

This request to the KA is digitally signed for providing authenticity of the data. The format that is used for this purpose is the PKCS#7 and ITP. Additionally, some data may be encrypted. In other cases it may just be obfuscated. An obfuscation example is the hash of a password. The password cannot be retrieved from the hash value, but it can be checked against it. This is done by hashing the password and comparing the two values.

There are various ways to collect registration data in the RA. One is by the physical presence of a human entity or a representative for technical entities (for

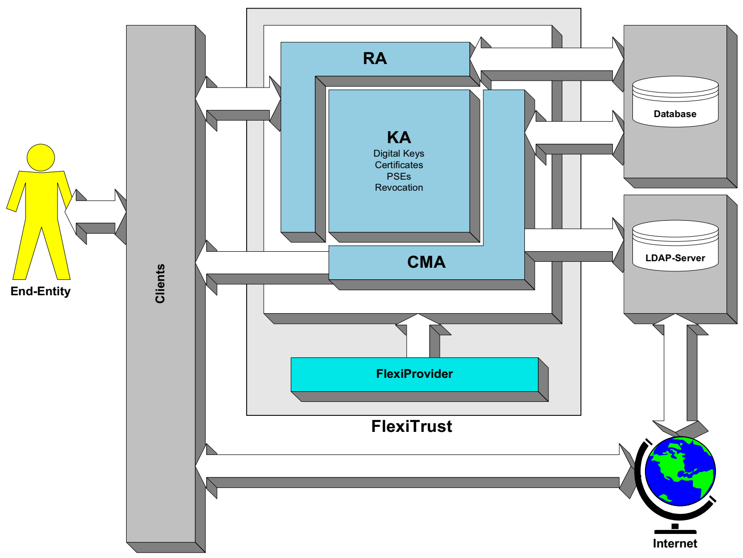

Figure 10.1: The FlexiTrust architecture

example the machine administrator). The collected data is checked for validity. After that a form is filled with this data. This is a web based form in the FlexiTrust RA. Another way is to use existent data from a secure store. Such a registration procedure, that reads data from an LDAP directory for registering the entities, is described in [LKWB05].

The set of collected data in the RA is important for the further functioning of a certification or revocation request. The DN of the owner of the certificate is determined in the RA. This is not done in the KA because it is accessible only to the trust center operators. Also, in case the end-entity already possesses a key pair, the public key of the end-entity is delivered to the RA that further delivers this to the KA. Other data that needs to be collected is the issuer of the certificate. The KA does not know which issuer should issue a certificate. Therefore, the RA defines the issuer or virtual host for whom the certification or revocation request is addressed. In case of revocation the RA also sends the serial number of the certificate that must be revoked from the KA.

The RA also makes some recommendations for parts of the data. Such data is special extensions in the certificate or CRL. Depending on the KA policy and certificate profile the KA may choose to accept or reject the recommendations. For example the RA may propose to set the keyUsage extension for the public key to encryption and digital signature. But the KA may decide to remove the encryption option if it is not possible to perform any key backup or realise another mechanism (like encryption of the certificate).

Another example is the validity period of the certificate. While the RA may propose that the certificate should be valid from the moment the data is collected, the KA may decide to set the notBefore date of the certificate exactly at the time of the signature creation. In another case, if the RA proposes that a certificate should be valid for a certain period of time, the KA may alter this time. If a certification request reaches the KA with a proposal for a validity period of one year, but the CA certificate is valid only for the next three months, the KA may decide to limit the notAfter date of the certificate to the notAfter date of the CA certificate. That is to ignore the one year proposal and alter it to three months. The rules about the processing of the products are described in the policy of the KA.

10.2.2 Key Authority

The KA is responsible for the certification, revocation, and generation of key pairs. It is triggered by a signed request coming from the RA. It first verifies the signature on the request. If the request is authentic (it comes from a known RA) it extracts the necessary data in order to perform a certification or a revocation. This depends on the type of request.

The KA then chooses the correct issuer for signing the certificates or CRLs. This is specified by the RA. Depending on the policy of the KA the contents of the certificate may differ from the ones proposed by the RA. The main task of the KA is to sign certificates and CRLs. In addition it produces key pairs and stores them in a PSE. The security of those processes is very important. For this reason the KA is usually offline.

The keys of the issuer are carefully protected in the KA. If these keys are compromised then an attacker may produce certificates in the name of this issuer. The keys may be found in hardware based PSEs. These are usually either HSMs or smart cards. In other cases they may also be stored in software for more flexibility and less costs. In both cases the passwords or PINs used for protecting the token may be shared among users. This is necessary if dual control is a requirement. In many cases the KA is also physically offline. This means that there is no network connection to the KA computer. The consequence of this is that the requests are transferred to and from the KA with an offline transport medium (like a CD, or a floppy disc, or a USB device).

When the KA has processed a request, it stores all information in a new request that is headed towards the CMA. This request contains new certificates, CRLs, PSEs, and data related to them.

10.2.3 Certificate Management Authority

The CMA receives requests from issuers like the KA. When a known issuer sends a request, this is processed according to the workflow defined for this issuer. First, the CMA verifies the authenticity of the request. It then extracts the information about the issuer. Afterwards it performs all necessary and predefined processing steps. This is the delivery of certificates and PSEs, publishing of the information on a public server, mail notifications to the users, and other tasks. The details of the CMA tasks have been discussed in Chapter 8. In Section 10.3.1 we will see the concrete tasks of the CMA in a PKI installation at the TUD.

10.2.4 Overview

In Table 10.1 we see a list of tasks in a trust center and the component that realises them. This list is illustrative rather than exhaustive. Some of them are implemented in the one or the other component. In other cases one task is performed by more than one component.

Task	**RA**	**KA**	**CMA**
Registration	●		
Data Collection	●		
Identification	●		
Set Issuer	●		
Set Subject DN	●		
Set Validity	●	●	
Set Extensions	●	●	
Sign Certificate		●	
Sign CRL		●	
Publish PKI Information			●
Mail Notifications			●
Deliver PSEs			●
Archiving		●	●
OCSP Backend			●
CRL Renewal Notification		●	●

Table 10.1: Task allocation overview

This trust center configuration enables a clear separation of the tasks that every component has to perform. Therefore, these tasks have been optimised for efficiency and security in each component. Moreover, the modular design has enabled us to effectively evaluate the software according to CC. Modularity is desired for such an evaluation.

This design addresses the central KA configuration. In most cases this KA is also offline. Therefore identification, registration, and data collection are done in a different component that in addition can be easily distributed. This is the RA or different LRAs. They cannot operate as KAs since this is too dangerous. The keys must be well protected. Storing and protecting the keys may be very expensive. In addition, the issuer is not usually distributed but it is just one instance because one key pair exists. Therefore, the tasks associated with signing and key generation are assigned to the KA.

The KA has extra security mechanisms enabled. It is one authority with strictly defined tasks that are not subject to frequent changes. This allows a careful design and auditing as far as the security properties concerning. Moreover, it is easier to enforce security mechanisms in one central instance. This applies to the implementation and organisational issues related to its installation.

The role of CMA in the trust center is to manage and administer the PKI products like certificates and CRLs for different issuers. It has different mechanisms for supporting core PKI services like dissemination of the information, delivery, or archiving. It enables different business logic and PKI information workflows. It can operate for more than one issuer and thus it can globally deliver its services to many PKI environments.

The virtual hosting design is found in all FlexiTrust components. In each of them it is realised with a different mechanism. This is due to the different requirements and design criteria of every component. It is a very flexible design principle, since it enables one installation to function for many issuers. This has great organisational and financial advantages. New issuers can be easily plugged in, with tasks that can be customised according to their business logic.

10.3 Current CMA Installations

FlexiTrust has already been installed as a trust center for supporting PKI in many different organisations and companies. Especially the CMA has delivered special services for each installation in order to support different business logic and information workflow. We will see some of those installations and the solutions that the CMA provides in the TUD PKI.

10.3.1 TU Darmstadt PKI

In the Technical University of Darmstadt a PKI has been installed. The purpose of this PKI is to provide all students of the university with a smart card containing a certificate and a key pair in order to use different applications. Some of them are: signing up for the examinations, courses enrolment, secure download of the lecture notes, secure connection through VPN, Windows Logon, secure email, and other. Parallel to this, the servers (web, email, etc.) of the university also receive a certificate in order to be able to use TLS. For realising the above mentioned tasks two virtual CAs are hosted in the computer center of the university.

Figure 10.2 shows the CA hierarchy. The root CA is the DFN-CA.[1] This is operated from the German research net, a well known and acknowledged organisation in Germany. It is responsible for connecting different universities in

[1]DFN stands for Deutsche Forschungsnetz.

Germany. This CA has cross-certified the TUD CA. This is a unilateral, inter-domain cross certificate. The TUD CA is responsible for issuing cross certificates for other CAs in the university. There are two subordinate CAs. The first one is the CCA (Chipcard CA) and the second one is the SCA (Server CA). These certificates are unilateral, intra-domain cross certificates.

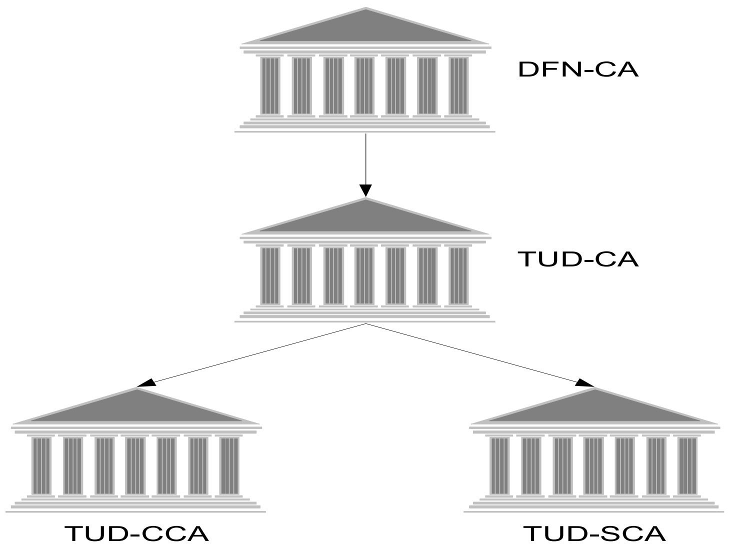

Figure 10.2: The TUD CA hierarchy

The CCA is the responsible CA for issuing the certificates to the students. All corresponding keys are located inside a smart card and they are created by the card distributor. The SCA is responsible for issuing the certificates to the university's servers. The keys can be located in software and they are created from the administrators of the servers.

We see which challenges the CMA has met in this project, its specification, and how it has realised the special business logic and workflow required.

One requirement is the virtual hosting. Two different CAs coexist in the system. These are the CCA and the SCA. The system is able to accept more CAs.[2] The two CAs have different processes and therefore configurations. We see the details of each configuration. Another requirement is the integration of the software in existent workflows. Since the entire user management of the university is based on an LDAP directory, the CMA supports the already established processes.

From the server CA only servers of the university are certified. The machine administrator creates keys for a server, sends a signed PKCS#10 request to the RA office, and receives the certificate per email or physical presence. The delivery

[2]For example a CA that certifies keys that the students create themselves or a CA for the university employees.

does not need to be done automatically since the numbers of the certificates issued is relatively small (30-40 certificates per year). The trust center administrator extracts the certificate and delivers it to the server administrator. For the SCA issuer two types of request exist. One is a certification request (without key creation) and revocation. The tasks of the CMA for this virtual CA are listed below with a small description where this is necessary.

1. Extract and archive certificates.
2. Notify the trust center administrator in case of failure.
3. Send mail notifications in case of revocation.

   Sends an email to the responsible administrator of the server whose certificate has been successfully revoked. A variety of information is contained in the email like the name of the server and the serial number of the certificate.
4. Publish LDAP information.
   (a) Secure communication with SSL.
   (b) Publish CRLs in a defined DN on the LDAP. This is part of the CMA configuration.
   (c) Remove the complete entry from the directory in case of revocation. That is to meet special administrative procedures in the computer center of the university.
   (d) Publish the certificate in a defined DN on the LDAP. This information is communicated to the CMA by the RA.
   (e) Remove any older information on the entry before publishing a certificate.
   (f) Publish the certificate with some meta-data like the serial number, the validity period, etc.
   (g) Organise the data on the directory similar to the scheme described in [LKWB05] for supporting revocation and other administrative tasks.

The Chipcard CA (CCA) issues certificates only to the students of the university. The key pair is located in a smart card. This smart card is sent to the students by post. A key pair is found in the card when it reaches the students. This has been created by the card distributor. The card distributor delivers all necessary data of the cards to the university computer center. Specifically a mapping between the serial number of the card and the public key located in the card is needed. The computer center knows which students received which card and therefore it can assign the correct public key to the correct student. This data is stored in an LDAP directory. The scheme is described in details in [LKWB05].

Moreover, a proof-of-possession is required when the students try to download their certificate, after receiving their card. This scheme is described in [KLW04a]. The application that helps the students to download their certificates is called the Card Manager and it has been developed in the TUD.[3]

The number of certificates issued is about 20.000 every year. During the initial registration of all students more than 4.000 requests were processed by the trust center in a working day. For this virtually hosted CA, the duties of the CMA are the following.

1. Extract and archive certificates.

2. Notify the trust center administrator in case of failure.

3. Send mail notifications for a successful certification.

   Sends an email to the students (at their university email address). This notifies them that the certificate is ready to be downloaded. After that notification, the students can use the Card Manager in order to download their certificate in the card.

4. Send mail notifications in case of revocation.

   Sends an email to the students (at their university email address), notifying them that their certificate has been successfully revoked. A variety of other information is contained in the email like the subject and serial number of the certificate.

5. Renew certificates automatically and send mail notifications.

   Searches for certificates that expire soon. Then it initiates a certificate renewal request that is sent to the CA. It then notifies the students (at their university email address), that their certificate expires soon.

6. Publish LDAP information.

   (a) Secure communication with SSL.

   (b) Publish CRLs in a defined DN on the LDAP. This is part of the CMA configuration.

   (c) Remove the encrypted certificate from the student's entry in the directory in case of revocation. This will ensure that the user will not try to re-write his old certificate in the card.

[3]It is a WebStart application. It can be downloaded from `http://www.tu-darmstadt.de/hrz/chipkarte/CardManager/` (date of access 10.12.2005). In order to fully use the application someone must be registered as a student in the university and in addition possess a TUD card.

(d) Initiate a new certification request in case of revocation. This is performed in order to enforce the scheme described in [LKWB05]. This is an automatic re-certification.

(e) Examine whether the public key on the certificate and the public key in the student card (it is also stored in LDAP) match. If they match proceed, otherwise do not.

(f) Encrypt the certificate and publish its encrypted value in a defined DN on the LDAP. This information is communicated to the CMA by the RA. This will enforce PoP according to [KLW04a]. This also ensures that the students can only use their correct cards since these are sent with the post and mistakes may take place.

(g) Organise the data on the directory similar to the scheme described in [LKWB05] for supporting revocation and other administrative tasks.

10.3.2 Other Installations

There are several other installations of the CMA at companies, organisations, or institutions. We will briefly list them.

One of them is at the German Root CA system. The issuer that the CMA is working for is the root CA of Germany. This CA issues qualified certificates according to the SigG for the root certificates or other certification service providers. An evaluation of the software based on CC was necessary in order to realise the trust center.

The computer center of the computer science department of the TUD operates a PKI. This PKI is a software based infrastructure that uses PKCS#12 files. It distinguishes between certificates for students and scientific staff. The certificates for the scientific staff are the basis of many functions. Two of these functions are room reservations and online administration of the courses. Tasks of the CMA in this installation are the mail notifications, printing of the personalisation data (PKCS#12 password), publishing, and waiting for acceptance acknowledgements.

At the Justus Liebig University of Giessen (Justus Liebig Universität Gießen) a smart card based PKI is installed. The card is used for online enrolment for courses and signing up for examinations among other applications. It is also used as a student identity card having printed data on the card body.

A variety of other current CMA installations exist. These are either in small or big companies and institutions. The goal of the CMA is to support a useful and secure PKI.

The continuous feedback from real life needs is integrated in the CMA, which grows according to those needs. It is a living part of a flexible trust center software. More developments and enhancements will be applied to the CMA in order to meet state of the art requirements.

Chapter 11

Future Work

This work proposed the certificate management authority or CMA as a new trust center component. We designed and implemented this authority and employed it in existing PKI projects with different requirements. This new authority is responsible for the certificate management tasks within a PKI. It specifies these tasks, how they are performed, and examines their requirements. The decentralised RAs and the offline issuers pose special challenges. The virtual hosting enables a flexible and efficient certificate management. Revocation and certificate status information is supported. Moreover, different workflows and business logic are realised.

The PKI installed at the TUD has special certificate management processes. A PoP is needed in order to prove that the correct student holds the correct card. The new scheme developed in the context of this work was integrated for providing these services. Further, a new software PSE delivery mechanism was proposed that is secure and easy to use.

The directory services inside a PKI were rigorously investigated. A library was implemented in Java that allows various schemes and cases to be realised. Especially, the special requirements of the SigG were addressed and their technical realisation was provided. The communication of the CMA with other trust center components was addressed. The properties and requirements for this type of communication were discussed. A new protocol based on XML was designed and implemented in order to meet the requirements of intra-trustcenter communication.

The security properties of the CMA were examined. We showed which are the security requirements for such an authority and how these can be fulfilled and implemented. In order to ease the evaluation process of the CMA according to Common Criteria, a protection profile (PP) can be established. This will lead to shorter evaluation time as well as lower costs. A future work is to create a protection profile for the CMA for a CC based evaluation.

The needs of a PKI change constantly. New protocols will appear that address various problems. New processes and services will become available. Various organisations operate a PKI for different reasons. Algorithms, schemes, and workflows may become insecure. These will be redesigned or refactored in order to overcome problems. The CMA is prepared to meet these new developments. These will be implemented in the CMA. Its flexible design enables an easy integration of the new developments.

A few new protocol proposals exist at the PKIX group [IET]. Some of them fall into the CMA's area of responsibility. These are mostly the ones related to repository protocols. These new features will be addressed and where this is necessary implemented or extended by the CMA. In specific environments, in which new requirements regarding certificate management occur, new solutions will be designed and integrated in the CMA.

Further, it can be investigated what are the tasks and services that the CMA can offer in several other infrastructures. Such an infrastructure is a PMI based on attribute certificates. It can be examined whether the presence of a CMA is meaningful in a PMI or not and what are the tasks related to attribute certificate management. The design and implementation of a CMA for a PMI is still open.

Practical identity based cryptosystems were proposed in [BF01]. Very few identity based infrastructures exist at the moment. The applications based on these infrastructures are still limited. The specifications of such an infrastructure is currently work in progress. Although most variants of those infrastructures do not employ certificates (but some do), it is still an open question whether a CMA is needed or not. In this case an issuer exists, that creates private keys for the infrastructure participants. The administration of those products may be delegated to the CMA. The certificate based variants, may profit even more from the presence of a CMA.

Bibliography

[AF99] C. Adams and S. Farell. Internet X.509 Public Key Infrastructure Certificate Management Protocols. *IETF Request For Comments*, 2510, March 1999.

[AFKM05] C. Adams, S. Farell, T. Kause, and T. Mononen. Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). *IETF Request For Comments*, 4210, September 2005.

[AL99] C. Adams and S. Lloyd. *Understanding Public-Key Infrastructure.* New Riders Publishing, 1999.

[And93] R. Anderson. The Classification of Hash Functions. In *Proceedings of 4th IMA Conference on Cryptography and Coding*, pages 83–93, 1993.

[ANL03] N. Asokan, V. Niemi, and P. Laitinen. On the Usefulness of Proof-of-Possession. In *Proceedings of the 2nd Annual PKI Research Workshop*, pages 122–127, April 2003.

[Aus01] T. Austin. *PKI.* John Wiley & Sons, Inc., 2001.

[Bao00] F. Bao. Introducing Decryption Authority into PKI. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, pages 288–296, December 2000.

[Bau99] M. Baum. Gültigkeitsmodell des SigG. *Datenschutz und Datensicherheit*, 23(4):199–205, 1999.

[BB04] K. Bicakci and N. Baykal. A New Design of Privilege Management Infrastructure with Binding Signature Semantics. In *Proceedings of Public Key Infrastructure: First European PKI Workshop: Research and Applications, EuroPKI 2004*, volume 3093 of *Lecture Notes in Computer Science*, pages 306–313, June 2004.

[BF01] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology: Proceedings of CRYPTO*

2001 - 21th Annual International Cryptology Conference, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, 2001.

[BHK$^+$99] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and Secure Message Authentication. In *Advances in Cryptology: Proceedings of CRYPTO '99 - 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233, 1999.

[BHR99] S. Boeyen, T. Howes, and P. Richard. Internet X.509 Public Key Infrastructure LDAPv2 Schema. *IETF Request For Comments*, 2587, June 1999.

[BPRS02] R. Brandner, U. Pordesch, A. Roßnagel, and J. Schachermayer. Langzeitsicherung qualifizierter elektronischen Signaturen. *Datenschutz und Datensicherheit*, 26(2):97–103, 2002.

[Buc01] J. A. Buchmann. *Introduction to Cryptography.* Springer-Verlag New York, Inc., 2001.

[BvdHH$^+$02] R. Brandner, M. van der Haak, M. Hartmann, R. Haux, and Schmücker P. Electronic Signature for Medical Documents – Integration and Evaluation of a Public Key Infrastructure in Hospitals. *Methods of Information in Medicine*, 41(4):321–330, 2002.

[BWKKW05] S. Blake-Wilson, G. Karlinger, T. Kobayashi, and Y. Wang. Using the Elliptic Curve Signature Algorithm (ECDSA) for XML Digital Signatures. *IETF Request For Comments*, 4050, April 2005.

[Car02] D. R. Carlton. *IPSec Tunneling im Internet.* mitp-Verlag, 2002.

[Cha82] D. Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203, 1982.

[Cha99] D. Chadwick. Smart Cards Aren't Always the Smart Choice. *IEEE Computer*, 32(12):142–143, December 1999.

[Cha00] D. W. Chadwick. Secure Directories. In *Proceedings of the NATO Advanced Networking Workshop on Advanced Security Technologies in Networking 2000*, June 2000.

[Cha03] D. Chadwick. The X.509 Privilege Management Infrastructure. In *Proceedings of the NATO Advanced Networkign Workshop on Advanced Security Technologies in Networking 2003*, pages 15–25, September 2003.

[CL] D. W. Chadwick and S. Legg. Internet X.509 Public Key Infrastructure LDAP Schema and Syntaxes for PKIs. Available at `http://sec.isi.salford.ac.uk/download/pkix-ldap-pki-schema-00.txt` (12.03.2006).

[CM04] D. Chadwick and S. Mullan. Returning Matched Values with the Lightweight Directory Access Protocol version 3 (LDAPv3). *IETF Request For Comments*, 3876, September 2004.

[Cri05a] Common Criteria. Common Criteria for Information Technology Security Evaluation - Part 2: Security functional requirements - Version 2.3. Available at `http://www.commoncriteriaportal.org/public/files/ccpart2v2.3.pdf` (24.11.2005), August 2005.

[Cri05b] Common Criteria. Common Criteria for Information Technology Security Evaluation - Part 3: Security assurance requirements - Version 2.3. Available at `http://www.commoncriteriaportal.org/public/files/ccpart3v2.3.pdf` (24.11.2005), August 2005.

[CWP04] D. De Cock, K. Wouters, and B. Preneel. Introduction to the Belgian EID Card BELPIC. In *Proceedings of Public Key Infrastructure: First European PKI Workshop: Research and Applications, EuroPKI 2004*, volume 3093 of *Lecture Notes in Computer Science*, pages 1–13, June 2004.

[DA99] T. Dierks and C. Allen. The TLS Protocol Version 1.0. *IETF Request For Comments*, 2246, January 1999.

[DBP96] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. In *Fast Software Encryption, Third International Workshop, Proceedings of FSE '96*, volume 1039 of *Lecture Notes in Computer Science*, pages 71–82, 1996.

[DFTT05] DT, SIT FHG, TÜV, and TELETRUST. CT-API 1.1, Application Independent Card Terminal Application Programming Interface for ICC Applications. Available at `https://www.secure.trusted-site.de/Download/CTAPI/CTAPI11EN.PDF` (03.10. 2005), 2005.

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[DH98] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. *IETF Request For Comments*, 2460, December 1998.

[DHIR05] L. Dietze, B. Holznagel, L. Lo Iacono, and C. Ruland. Qualifizierte Signatur im elektronischen Messdatenaustausch. In *Proceedings of SICHERHEIT 2005*, volume P-62 of *Lecture Notes in Informatics*, pages 335–348, April 2005.

[Eck04] C. Eckert. *IT-Sicherheit, Konzepte – Verfahren – Protokolle*. Oldenbourg Wissenschaftsverlag GmbH, 3 edition, 2004.

[Ell02] C. Ellison. Improvements on Conventional PKI Wisdom. In *Online Proceedings of the 1st Annual PKI Research Workshop*, April 2002. Available at `http://www.cs.dartmouth.edu/~pki02/` (23.04.2005), published also as Special Publication by NIST.

[ErJ01] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). *IETF Request For Comments*, 3174, September 2001.

[FFW99] J. Feghhi, J. Feghhi, and P. Williams. *Digital Certificates.* Addison-Wesley, 1999.

[FH02] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. *IETF Request For Comments*, 3281, April 2002.

[FHK95] D. Fox, P. Horster, and P. Kraaibeek. Grundüberlegungen zu Trust Centern. In *Proceedings of Trust Center 95, Trust Center – Grundlagen, Rechtliche Aspekte, Standardisierung und Realisierung*, pages 1–10, 1995.

[FKL$^+$06] S. Fritsch, V. Karatsiolis, M. Lippert, A. Wiesmaier, and J. Buchmann. Towards Secure Electronic Workflows. In *Proceedings of Public Key Infrastructure: Third European PKI Workshop: theory and practice, EuroPKI 2006*, volume 4043 of *Lecture Notes in Computer Science*, pages 154–168, June 2006.

[FMS$^+$05] M. Franklin, K. Mitcham, S. Smith, J. Stabiner, and O. Wild. CA-in-a-Box. In *Proceedings of Public Key Infrastructure: Second European PKI Workshop: Research and Applications, EuroPKI 2005*, volume 3545 of *Lecture Notes in Computer Science*, pages 180–190, June 2005.

[FNG03] T. Fickert, M. Nau, and R. W. Gerling. Encrypting File System unter Windows. *Datenschutz und Datensicherheit*, 27(4):223–227, 2003.

[FS03] N. Ferguson and B. Schneier. *Practical Cryptography.* Wiley Publishing, Inc., 2003.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[GHSW98] A. Grimstad, R. Huber, S. Sataluri, and M. Wahl. Naming Plan for Internet Directory-Enabled Applications. *IETF Request For Comments*, 2377, September 1998.

[Goo00] G. Good. The LDAP Data Interchange Format (LDIF) – Technical Specification. *IETF Request For Comments*, 2849, June 2000.

[Gre02] B. Greenblatt. *Building LDAP-Enabled Applications with Microsoft's Active Directory and Novell's NDS*. Prentice Hall PTR, 2002.

[GSB$^+$04] R. Guida, R. Stahl, T. Bunt, G. Secrest, and J. Moorcones. Deploying and Using Public Key Technology: Lessons Learned in Real Life. *IEEE Security & Privacy*, 2(4):67–71, July-August 2004.

[Gut00] P. Gutmann. A Reliable, Scalable General-purpose Certificate Store. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, pages 278–287, December 2000.

[Gut03] P. Gutmann. Plug-and-Play PKI: A PKI your Mother Can Use. In *Proceedings of the 12th USENIX Security Symposium*, pages 45–58, August 2003.

[Gut06] P. Gutmann. Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP. *IETF Request For Comments*, 4387, February 2006.

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure certificate and CRL Profile. *IETF Request For Comments*, 2459, January 1999.

[HH99] R. Housley and P. Hoffman. Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP. *IETF Request For Comments*, 2585, May 1999.

[HM02] J. Hodges and R. Morgan. Lightweight Directory Access Protocol (v3): Technical Specification. *IETF Request For Comments*, 3377, September 2002.

[HMW00] J. Hodges, R. Morgan, and M. Wahl. Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security. *IETF Request For Comments*, 2830, May 2000.

[Hou99] R. Housley. Cryptographic Message Syntax. *IETF Request For Comments*, 2630, June 1999.

[HP01] R. Housley and T. Polk. *Planning for PKI.* John Wiley & Sons, Inc., 2001.

[HPFS02] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *IETF Request For Comments*, 3280, April 2002.

[HS97] T. Howes and T. Smith. The LDAP URL Format. *IETF Request For Comments*, 2255, December 1997.

[IET] PKIX Working Group IETF. Public-Key Infrastructure IETF Working Group. Available at `http://www.ietf.org/html.charters/pkix-charter.html` (12.03.2006). Internet drafts are updated often and on this site the newest status can be obtained.

[IT97] Recommendation X.509 ITU-T. Information Technology – Open Systems Interconnection – The Directory: Authentication Framework. August 1997.

[IT01] Recommendation X.500 ITU-T. Information Technology – Open Systems Interconnection – The Directory: Overview of Concepts, Models and Service. Februar 2001.

[IT02] Recommendation X.680 ITU-T. Information Technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation, July 2002.

[JCB03] H. Jianzhong, X. Changsheng, and C. Bin. Research and Implement of an Encrypted File System Used to NAS. In *Second IEEE International Security in Storage Workshop, (SISW 2003)*, pages 73–77, October 2003.

[JMV01] D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, August 2001.

[Jos01] S. Josefsson. Network Application Security Using The Domain Name System. Master thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, 2001.

[Jos06] S. Josefsson. Storing Certificates in the Domain Name System (DNS). *IETF Request For Comments*, 4398, March 2006.

[KA98] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *IETF Request For Comments*, 2401, November 1998.

[Kar03] V. Karatsiolis. Low Level Design - Infrastructure Services, Version 4.0, October 2003.

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. *IETF Request For Comments*, 2104, February 1997.

[Kli05a] V. Klima. Finding MD5 Collisions - a Toy For a Notebook. *IACR Cryptology ePrint Archive*, 2005(75), March 2005. Available at `http://eprint.iacr.org/` (05.11.2005).

[Kli05b] V. Klima. Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications. *IACR Cryptology ePrint Archive*, 2005(102), March 2005. Available at `http://eprint.iacr.org/` (05.11.2005).

[KLW04a] V. Karatsiolis, M. Lippert, and A. Wiesmaier. Using LDAP Directories for Management of PKI Processes. In *Proceedings of Public Key Infrastructure: First European PKI Workshop: Research and Applications, EuroPKI 2004*, volume 3093 of *Lecture Notes in Computer Science*, pages 126–134, June 2004.

[KLW$^+$04b] V. Karatsiolis, M. Lippert, A. Wiesmaier, A. Pitaev, M. Ruppert, and J. Buchmann. Towards a Flexible Intra-Trustcenter Management Protocol. In *The Third International Workshop for Applied PKI, IWAP 2004*, October 2004.

[KLW05] V. Karatsiolis, M. Lippert, and A. Wiesmaier. Planning for Directory Services in Public Key Infrastructures. In *Proceedings of SICHERHEIT 2005*, volume P-62 of *Lecture Notes in Informatics*, pages 349–360, April 2005.

[Koh78] L. M. Kohnfelder. Towards a Practical Public Key Cryptosystem. Bachelor thesis, Dept. of Electrical Engineering, MIT, May 1978.

[KWG$^+$98] S. Kille, M. Wahl, A. Grimstad, R. Huber, and S. Sataluri. Using Domains in LDAP/X.500 Distinguished Names. *IETF Request For Comments*, 2247, January 1998.

[Lab05] OpenCA Labs. OpenCA Labs. Available at `http://www.openca.org/` (08.11.2005), 2005.

[LCZ05] S. S. Lim, J. H. Choi, and K. D. Zeilenga. The Design and Implementation of LDAP Component Matching for Flexible and Secure Certificate Access in PKI. In *Online Proceedings of the 4th Annual PKI R&D Workshop*, April 2005. Available at `http://middleware.internet2.edu/pki05/proceedings/` (03.03.2006).

[Leg01a] The German Legislator. Gesetz über Rahmenbedingungen für elektronische Signaturen und zur Änderung weiterer Vorschriften. *Bundesgesetzblatt Jahrgang 2001 Teil I*, Nr. 22:876–884, 21. Mai 2001.

[Leg01b] The German Legislator. Verordnung zur elektronischen Signatur (Signaturverordnung – SigV). *Bundesgesetzblatt Jahrgang 2001 Teil I*, Nr. 59:3074–3084, 21. November 2001.

[Leg04] S. Legg. Lightweight Directory Access Protocol (LDAP) and X.500 Component Matching Rules. *IETF Request For Comments*, 3687, February 2004.

[LKWB05] M. Lippert, E. Karatsiolis, A. Wiesmaier, and J. Buchmann. Directory Based Registration in Public Key Infrastructures. In *The 4th International Workshop for Applied PKI, IWAP 2005*, pages 17–32, September 2005.

[LKWB06] M. Lippert, V. Karatsiolis, A. Wiesmaier, and J. Buchmann. Life-cycle management of X.509 certificates based on LDAP directories. *Journal of Computer Security*, 14(5):419–439, 2006.

[LLR+03] M. Lippert, S. Lange, G. Raptis, M. Ruppert, C. Valentin, R. Vogt, and A. Wiesmaier. Sicherheitsvorgaben für FlexiTrust 3.0, Version 1.1.1, November 2003.

[MAM+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP. *IETF Request For Comments*, 2560, June 1999.

[Met02] S. J. Metsker. *Design Patterns Java Workbook.* Addison-Wesley, 2002.

[MLSW00] M. Myers, X. Liu, J. Schaad, and J. Weinstein. Certificate Management Messages over CMS. *IETF Request For Comments*, 2797, April 2000.

[MM05] C. Mrugalla and S. Maseberg. IT-Grundschutz-basierendes Sicherheitskonzept für die Virtuelle Poststelle des Bundes. In *Proceedings of SICHERHEIT 2005*, volume P-62 of *Lecture Notes in Informatics*, pages 11–14, April 2005.

[MR00] P. McDaniel and A. Rubin. A Response to "Can We Eliminate Certificate Revocation Lists?". In *Proceedings of the 4th International Conference on Financial Cryptography '00 (FC00)*, volume 1962 of *Lecture Notes in Computer Science*, pages 245–258, February 2000.

[MSNP04] R. Megginson, M. Smith, O. Natkovich, and J. Parham. Lightweight Directory Access Protocol (LDAP) Client Update Protocol (LCUP). *IETF Request For Comments*, 3928, October 2004.

[MvOV97] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Inc., 1997.

[Mye97] J. Myers. Simple Authentication and Security Layer (SASL). *IETF Request For Comments*, 2222, October 1997.

[NM96] D. Naccache and D. M'Raïhi. Cryptographic Smart Cards. *IEEE Micro*, 16(3):14–24, June 1996.

[NT94] National Institute of Standards NIST and Technology. FIPS 186 – Digital Signature Standard (DSS). Available at `http://www.itl.nist.gov/fipspubs/fip186.htm` (24.04.2005), May 1994.

[NT02] National Institute of Standards NIST and Technology. FIPS 180-2 – Secure Hash Standard. Available at `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf` (05.11.2005), August 2002.

[Ope04] OpenLDAP. OpenLDAP Project. Available at `http://www.openldap.org` (07.02.2004), 2004.

[Ope05] OpenSSL. OpenSSL Project. Available at `http://www.openssl.org` (08.11.2005), 2005.

[Pla02] T. Planz. Entwurf und Implementierung einer Infrastrukturkomponente für ein Java-basiertes Trustcenter. Master thesis, Dept. of Computer Science, Technische Universität Darmstadt, 2002.

[Ram04a] B. Ramsdell. Secure / Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling. *IETF Request For Comments*, 3850, July 2004.

[Ram04b] B. Ramsdell. Secure / Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. *IETF Request For Comments*, 3851, July 2004.

[Res00] E. Rescorla. HTTP Over TLS. *IETF Request For Comments*, 2818, May 2000.

[Res01] E. Rescorla. *SSL and TLS, Designing and Buidling Secure Systems*. Addison-Wesley, 2001.

[Riv92] R. Rivest. The MD5 Message-Digest Algorithm. *IETF Request For Comments*, 1321, April 1992.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[RSA93] Laboratories RSA. PKCS#7 v1.5: Cryptographic Message Syntax Standard. Available at `http://www.rsasecurity.com/rsalabs/pkcs/` (23.04.2005), November 1993.

[RSA99] Laboratories RSA. PKCS#12 v1.0: Personal Information Exchange Syntax. Available at `http://www.rsasecurity.com/rsalabs/pkcs/` (23.04.2005), June 1999.

[RSA00] Laboratories RSA. PKCS#10 v1.7: Certification Request Syntax Standard. Available at `http://www.rsasecurity.com/rsalabs/pkcs/` (23.04.2005), May 2000.

[RSA02] Laboratories RSA. PKCS #1 v2.1: RSA Cryptography Standard. Available at `http://www.rsasecurity.com/rsalabs/pkcs/` (06.11.2005), June 2002.

[RSA04] Laboratories RSA. PKCS#11 v2.20: Cryptographic Token Interface Standard. Available at `http://www.rsasecurity.com/rsalabs/pkcs/` (23.04.2005), June 2004.

[Sch05] J. Schaad. Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF). *IETF Request For Comments*, 4211, September 2005.

[Sha79] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[Sho94] P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings of the 35th IEEE Annual Symposium on Foundations of Computer Science*, pages 124–134, November 1994.

[Sma03] N. Smart. *Cryptography: An Introduction*. McGraw-Hill, 2003.

[Smi00] M. Smith. Definition of the inetOrgPerson LDAP Object Class. *IETF Request For Comments*, 2798, April 2000.

[SUNa] SUN. Java Naming and Directory Interface. Available at `http://java.sun.com/products/jndi/` (08.02.2004).

[SUNb] SUN. Java Web Start Technology. `http://java.sun.com/products/javawebstart/` (03.10.2005).

[TT] T7 and TeleTrust. Common ISIS-MTT Specifications for Interoperable PKI Applications - version 1.1. Available at `http://www.isis-mtt.t7-isis.org/` (12.03.2006).

[W3C01] World Wide Web Consortium W3C. XML Key Management Specification (XKMS). Available at `http://www.w3.org/TR/xkms/` (10.04.2005), March 2001.

[W3C02a] World Wide Web Consortium W3C. XML Encryption Syntax and Processing. Available at `http://www.w3.org/TR/xmlenc-core/` (10.04.2005), December 2002.

[W3C02b] World Wide Web Consortium W3C. XML-Signature Syntax and Processing. Available at `http://www.w3.org/TR/xmldsig-core/` (10.04.2005), February 2002.

[Wah97] M. Wahl. A Summary of the X.500(96) User Schema for use with LDAPv3. *IETF Request For Comments*, 2256, December 1997.

[WAHM00] M. Wahl, H. Alvestrand, J. Hodges, and R. Morgan. Authentication Methods for LDAP. *IETF Request For Comments*, 2829, May 2000.

[WCHK97] M. Wahl, A. Coulbeck, T. Howes, and S. Kille. Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions. *IETF Request For Comments*, 2252, December 1997.

[WDZ03] C. P. Wright, J. Dave, and E. Zadok. Cryptographic File Systems Performance: What You Dont Know Can Hurt You. In *Second IEEE International Security in Storage Workshop, (SISW 2003)*, pages 47–61, October 2003.

[WFLY04] X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. *IACR Cryptology ePrint Archive*, 2004(199), August 2004. Available at `http://eprint.iacr.org/` (05.11.2005).

[WHK97] M. Wahl, T. Howes, and S. Kille. Lightweight Directory Access Protocol (v3). *IETF Request For Comments*, 2251, December 1997.

[WKLB05] A. Wiesmaier, V. Karatsiolis, M. Lippert, and J. Buchmann. The Workshop - Implementing Well Structured Enterprise Applications. In *Proceedings of the 2005 International Conference on Software Engineering Research and Practice, SERP'05*, pages 947–953, June 2005.

[WLK04] A. Wiesmaier, M. Lippert, and V. Karatsiolis. The Key Authority – Secure Key Management in Hierarchical Public Key Infrastructures. In *Proceedings of the International Conference on Security and Management, SAM '04*, pages 89–93, June 2004.

[WLK$^+$05] A. Wiesmaier, M. Lippert, V. Karatsiolis, G. Raptis, and J. Buchmann. An Evaluated Certification Services System for the German National Root CA – Legally Binding and Trustworthy Transactions in E-Business and E-Government. In *Proceedings of the 2005 International Conference on e-Business, Enterprise Information Systems, e-Government, and Outsourcing, EEE'05*, pages 103–108, June 2005.

[Wor] PCSC Workgroup. PC/SC Specification. Available at `http://www.pcscworkgroup.com/` (06.10.2005).

[WRL$^+$06] A. Wiesmaier, U. Rauchschwalbe, C. Ludwig, B. Henhapl, M. Ruppert, and J. Buchmann. Intrinsically Legal-For-Trade Objects by Digital Signatures. In *Proceedings of SICHERHEIT 2006*, Lecture Notes in Informatics, pages 218–221, Februar 2006.

[WY05] X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In *Advances in Cryptology: Proceedings of EUROCRYPT '2005 - 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35, May 2005.

[WYY05] X. Wang, Y. L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In *Advances in Cryptology: Proceedings of CRYPTO 2005 - 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36, 2005.

[Zer03] T-Systems Zertifizierungsstelle. Bestätigung von Produkten für qualifizierte elektronische Signaturen. Available at `http://www.bundesnetzagentur.de/media/archive/1699.pdf` (24.01.2006), December 2003.

Index