SRI INTERNATIONAL

SRI Technical Report

# PKI and Revocation Survey

Grit Denker and Jonathan Millen
Computer Science Laboratory

Yutaka Miyake
KDD R&D Laboratories, Inc.

SRI-CSL-2000-01, August 2000

Supported by KDD R&D Laboratories, Inc.

333 Ravenswood Avenue • Menlo Park, CA 94025-3493 • (650) 859-2000

# PKI and Revocation Survey[1]

Grit Denker and Jonathan Millen
Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA
{denker,millen}@csl.sri.com

Yutaka Miyake
KDD R&D Laboratories, Inc., Kamifukuoka-shi, 356-8502, JAPAN
miyake@kddlabs.co.jp

**Abstract**

This survey covers basic information about public key infrastructures and summarizes the predominant technology and standards. Special attention is given to mechanisms for certificate revocation. Methods for CRL distribution and validity checking are compared.

[1]Supported by KDD R&D Laboratories, Inc.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Public key cryptography is a triumph of the imagination. It is outstanding among modern inventions as a practical application of a mathematical theory once thought to have none. It would not have the impact that it does without desktop computers and the Internet, but its power comes from the intricate interplay of logic, computer science, and number theory. Public key techniques have liberated cryptography from the exclusive domain of governments and large organizations, and made it a tool accessible to everyone on the Internet, for their own purposes as well as for communication with financial and other institutions.

The logic of public key applications depends on the association of public keys with names identifying individuals, organizations, and the privileges and authorities they hold. The binding of a public key to a name is expressed in a certificate, digitally signed by some authority. Who are the authorities? How are certificates created and distributed, and how can they be revoked if necessary? These are the questions answered by a public key infrastructure (PKI).

There is no single universal PKI, but rather a collection of schemes supported by private research, commercial vendors, government agencies, and standards committees. They are not all interoperable, but there does appear to be a convergence of support for the certificate format in the ISO-ITU X.509 standard. An important advantage of the X.509 certificate standard is its ability to define extensions that can be customized to meet the needs of applications.

This survey covers basic information about public key infrastructures and summarizes the predominant technology and standards. Special attention is given to mechanisms for certificate revocation. The fundamental cryptographic techniques and terminology are summarized in Section 2. Concepts of alternative public key infrastructures and approaches to certificate revocation are presented in Section 3. Section 4 focuses on performance issues for revocation and validity checking. Section 5 gives an overview of PKI standards activities, and Section 6 outlines the services and products available from a few representative commercial vendors.

# Chapter 2

# Cryptographic Basics

The reader of this survey is assumed to have the usual computer professional's knowledge of the basic facts about modern data encryption. The purpose of this section is primarily to highlight those facts that are most relevant to the survey, and to familiarize the reader with the terminology used here.

A cryptosystem has algorithms for encrypting plaintext messages into ciphertext and for decrypting ciphertext into the original plaintext. Both encryption and decryption require the use of a key. In a *symmetric key* cryptosystem, a message encrypted with a key must be decrypted using the same key. A *public key* cryptosystem uses pairs of mathematically related keys. One key of a pair is made public and used to encrypt messages, and the other is kept secret and used to decrypt ciphertext messages that were encrypted using its mate.

Public key algorithms are much slower than symmetric key algorithms, so public key cryptosystems are used primarily for digital signatures or distributing symmetric keys, and symmetric key cryptosystems are used to encrypt longer messages or data files.

## 2.1 Symmetric Key Encryption

Good symmetric key cryptosystems have been publicly available for about twenty years, since the introduction of DES (the Data Encryption Standard). DES is a 64-bit block cipher with a 56-bit key (extended with 8 bits of parity to 64 bits). A software implementation on a 450MHz machine encrypts data at about seven megabytes per second. Export restrictions on DES from the U.S. stimulated the development of other algorithms in Europe and elsewhere, such as IDEA (International Data Encryption Algorithm). Also a 64-bit block cipher, IDEA uses a 128-bit key and runs at about six megabytes per second on the same hardware. (This and other performance figures are taken from [4].)

In 1997, NIST (U.S. National Institute for Standards and Technology) announced the initiation of a competition for the AES (Advanced Encryption Standard). The call stipulated that the AES would specify an unclassified, publicly disclosed symmetric-key encryption algorithm. It must support a block size of 128 bits and key sizes of 128, 192, and 256 bits.

The AES finalist candidate algorithms are MARS, RC6, Rijndael, Serpent, and Twofish. They run at speeds of 6 to 20 megabytes per second in software on a 450 MHz machine. As of mid-August, 2000, the choice among them is expected soon.

## 2.2 Public Key Encryption

Public key cryptography was introduced by the Diffie-Hellman paper [5], which gave an algorithm by which two parties could combine secret keys to form a new shared key. If $g$ is a primitive element of a finite field, and $x$ is a secret key chosen from that field, $g^x$ is effectively a public key that the other party can combine with his own secret $y$ to form the shared key $g^{xy}$ (in the finite field). (This scheme was due in part to Ralph Merkle, according to Hellman.) Some independent early work along these lines, unpublished but recently released, has come to light [6].

The best known public key cryptosystem is RSA (Rivest-Shamir-Adleman), which uses finite-field arithmetic with a modulus of $n = pq$, where $p$ and $q$ are large secret primes. A message $m$ represented as a number modulo $n$ is encrypted as $m^e$, and decrypted as $(m^e)^d \equiv m$ mod $n$ for public exponent $e$ and private exponent $d$ such that $ed \equiv 1$ mod $(p-1)(q-1)$. The size of the modulus $n$ determines the key and block size; 1024 bits is currently considered safe. This relatively large key size is based on current capabilities for factoring the public modulus $n$ into its secret factors $p$ and $q$. RSA encryption can be done at about 10 kilobytes per second, in software on a 450 MHz machine. Decryption can often be done faster because the public exponent can be chosen to be relatively small, although very small exponents (like 3 or 5) have been shown to be unsafe in the context of certain protocols.

There are a few other public key cryptosystems. Some of the more promising recent ones use elliptic curve groups over finite fields. These are not really new algorithms, but rather the implementation of existing algorithms using a different system to do the arithmetic. The advantage of elliptic curve cryptosystems is that they do not have the factoring vulnerability of RSA, and a modulus of 160 bits is believed to be about as safe as 1024 bits for RSA.

The standards document *IEEE P1363: Standard Specifications For Public Key Cryptography* has recently been adopted as a standard by the IEEE [24]. It includes algorithms and other reference data for several public key cryptosystems, including those based on the discrete logarithm, integer factorization, and elliptic curves.

## 2.3 One-way Hash Functions

*Hash functions* are used to produce a fixed-length digest of an arbitrarily long message. A hash function is *one-way* if it is cryptographically hard to invert, i.e., to find any message that hashes to a given digest. If the message is combined with a secret key before hashing, or the digest is encrypted with a secret key, the result is a *message authentication code* (MAC).

Two popular hash algorithms are SHA (Secure Hash Algorithm) from NIST, which produces a 160-bit digest and runs at about 25 megabytes per second on a 450 MHz machine; and MD5, which produces a 128-bit digest and runs at about 57 megabytes per second on the same machine.

## 2.4 Digital Signature

A *digital signature* is essentially a MAC that can be checked with public information. A digital signature can be produced using any public key cryptosystem: the signer encrypts a digest of the message with her private key. The verifier checks authenticity of the message by decrypting the MAC with the known public key, and comparing it to the computed digest.

## 2.5 Certificates

A public key *certificate* is a message asserting a binding between a public key and associated security identification and authorization data. The certificate is digitally signed by some authority. A certificate typically contains at least a serial number, the name of the holder of the associated private key, the name of the signing authority, and a validity period (from the time at which the certificate is effective to the time at which the certificate expires). Certificate validity periods vary considerably according to their use. A validity period can be very long – a year or more – for the certificate of an established institution, or only hours or minutes for a certificate giving access to a server resource for a particular session.

The two principal certificate design issues are the nature of the names of key holders and the choice of other security identity and authorization data. The ISO-ITU standard X.509 defines a certificate format that has been nearly universally adopted. It uses so-called "distinguished names" with a hierarchical structure analogous to that of Internet domain names that supports global unique naming. It provides a useful set of other fields for security-relevant and implementation data. It also permits customized extensions to be defined for application-dependent fields. More detail on the X.509 format and extensions are given in section 5.1 and section 5.2.2.

## 2.6 ISO Standard Terminology

The following terms are defined in [17].

**base CRL:** A CRL that is used as the foundation in the generation of a dCRL.

**certificate policy:** A named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. For example, a particular certificate policy might indicate applicability of a type of certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range.

**Certificate Revocation List (CRL):** A signed list indicating a set of certificates that are no longer considered valid by the certificate issuer. In addition to the generic term CRL, some specific CRL types are defined for CRLs that cover particular scopes.

**certificate serial number:** An integer value, unique within the issuing authority, which is unambiguously associated with a certificate issued by that CA.

**certificate validation:** The process of ensuring that a certificate was valid at a given time, including possibly the construction and processing of a certification path, and ensuring that all certificates in that path were valid (i.e. were not expired or revoked) at that given time.

**Certification Authority (CA):** An authority trusted by one or more users to create and assign public-key certificates. Optionally the certification authority may create the users' keys.

**certification path:** An ordered sequence of certificates of objects in the DIT which, together with the public key of the initial object in the path, can be processed to obtain that of the final object in the path.

**CRL distribution point:** A directory entry or other distribution source for CRLs; a CRL distributed through a CRL distribution point may contain revocation entries for only a subset of the full set of certificates issued by one CA or may contain revocation entries for multiple CAs.

**delta-CRL (dCRL):** A partial revocation list that only contains entries for certificates that have had their revocation status changed since the issuance of the referenced base CRL.

**end entity:** A certificate subject that uses its private key for purposes other than signing certificates or an entity that is a relying party.

**hash function:** A (mathematical) function which maps values from a large (possibly very large) domain into a smaller range. A "good" hash function is such that the results of applying the function to a (large) set of values in the domain will be evenly distributed (and apparently at random) over the range.

**one-way function:** A (mathematical) function f which is easy to compute, but which for a general value y in the range, it is computationally difficult to find a value x in the domain such that f(x)=y. There may be a few values y for which finding x is not computationally difficult.

**private key:** (in a public key cryptosystem) that key of a user's key pair which is known only by that user.

**public key:** (In a public key cryptosystem) that key of a user's key pair which is publicly known.

**public-key certificate:** The public key of a user, together with some other information, rendered unforgeable by encipherment with the private key of the certification authority which issued it.

**Public Key Infrastructure (PKI):** The infrastructure able to support the management of public keys able to support authentication, encryption, integrity or non-repudiation services.

**relying party:** A user or agent that relies on the data in a certificate in making decisions.

**security policy:** The set of rules laid down by the security authority governing the use and provision of security services and facilities.

**trust:** Generally, an entity can be said to "trust" a second entity when it (the first entity) makes the assumption that the second entity will behave exactly as the first entity expects. This trust may apply only for some specific function. The key role of trust in this framework is to describe the relationship between an authenticating entity and an authority; an entity shall be certain that it can trust the authority to create only valid and reliable certificates.

## 2.7 Infrastructure Concepts

A certification authority (CA) digitally signs a certificate containing a public key to authenticate the identifying information in the certificate.

A public key infrastructure (PKI) is defined by answers to the following questions:

- What entities are CAs?
- What trust does a CA signature imply?
- How is that trust awarded?
- How are certificates revoked?

There have been two main approaches to PKI: certification hierarchies and certification networks. In a certification hierarchy, there is a root CA whose authority is derived from an organization that defines its own membership and policies, such as a company, government, or other institution. Subordinate CAs in the hierarchy derive their authority by delegation from higher-level ones. The collection of CAs in a hierarchy below a root CA is called a *domain*. Although at one time it was imagined there could be single global certification hierarchy, at present there are many instances of such hierarchies set up by different organizations.

In a certification network, there is no ordering among authorities. Any key can be used to sign any certificate, with the possibility of loops and multiple signatures. The best known certification network in the Internet is the PGP (Pretty Good Privacy) Web of Trust. Any user with a public key can sign a certificate and thus act as a certification authority. A user who signs a certificate is called an *introducer* in this context. A relying party trusts an introducer by virtue of personal knowledge, and the introducer is trusted only to check that the user named in the certificate is in possession of the secret key corresponding to the public key in the certificate.

The two main approaches can be combined at a high level into a network of certification hierarchies. In effect, the root CAs of a set of organizations form a web of trust, where some root CAs sign *cross certificates* for other root CAs. Cross certificates can also be introduced between non-root CAs, for efficiency reasons.

### 2.7.1 Certification Hierarchies

As a practical matter, a certification hierarchy is supported by software and hardware providing various functions:

- Registration Authority (RA) for identifying new users and creating (or revoking) certificates, associated with a CA;
- User Client software to generate key pairs, communicate with the RA and other servers, encrypt, decrypt, or sign messages, and maintain a local cache of certificates;
- Repository for certificates or revocation notices for users in a domain;
- Validation Authority to reply to requests for status of individual certificates, by checking the current revocation list.

The registration authority for a CA provides an interface by which a new user can request a certificate. It usually also has an administrator interface to maintain a database containing information relevant to the certificate issuance policy.

An example of client software is a Web browser, using HTTP to communicate with the RA, and containing a cryptographic module for generating keys and performing cryptographic message operations. The client must be initialized with the public key of some CA.

Repositories and validation authorities can maintain information about more than one domain.

#### Certification Paths

In order to accept a certificate signed by a CA, a relying party must check the signature using the public key known to belong to the signer, and the relying party must also trust the signer to act as a CA. In a certification hierarchy, the public key of a subordinate CA, and the trust in that CA, can be conveyed by a certificate signed by a higher-level CA, delegating CA authority. If the public key and authority of the higher-level CA is not already known, that can be conveyed by another certificate, and so on. The sequence of certificates called a certification path, and in this report we refer to it also as a *certificate chain*. A certificate chain must stop at some previously known CA, referred to as the *anchor* of the chain.

Once a CA certificate has been received and validated, it can be cached so that the entire chain is not needed again, until the subject certificate expires.

### 2.7.2 Web of Trust, PGP

The Web of Trust in the Internet grew out of the use of the freely available PGP (Pretty Good Privacy) software package [31]. This software allows users to generate RSA key pairs, sign public key certificates, and use these keys to encrypt, decrypt, sign, and authenticate messages. Temporary symmetric keys are also generated so that messages can be encrypted efficiently using IDEA; the temporary key is sent encrypted using the public key.

PGP certificates support multiple signatures. There are no certificate chains, because there is no authority hierarchy; a relying party must make an individual decision whether to trust a particular signer as an introducer (certifying authority). Hence, a certificate is accepted if any one of the signatures belongs to a trusted introducer. PGP also supports partial trust, and a certificate may also be accepted if it is signed by more than one partially trusted introducer.

There are public servers acting as repositories for PGP certificates. In particular, there are eight "clones" of one at MIT, with a database of over 57,000 certificates.

### 2.7.3 Certificate Revocation

Normally, a certificate includes an expiration date, beyond which it is no longer valid. Sometimes there are reasons for revoking a certificate before the expiration date [9, 10]. The two main reasons are compromise of the private key and change of status. If a private key falls into the hands of a malicious party, that party could read confidential messages sent to the key owner, or digitally sign documents attributed to the key owner. If a certificate conveys privileges due to the position of the subject in some organization, the certificate may be revoked if that subject is removed from employment or from the privileged position.

A revocation notice is a kind of certificate stating that a public key certificate with a given serial number is no longer valid. The revocation notice should be signed by either the revoked key or a certificate authority, to avoid malicious denial of service to the key owner. Revocation notices are submitted to the RA by the key owner or domain administrator.

A certificate revocation list (CRL) is simply a list of revocation notices issued by a CA. In a certification hierarchy, the CRL is created and maintained at a repository under the control of a domain root CA, and it may be distributed to other repositories. Notices in the CRL are not individually signed. If the CRL is distributed over the Internet, the CRL as a whole can be dated and signed, so that the relying party will know that it is recent and authentic.

There are several performance issues relating to optimal distribution of CRLs. Among these are:

- Should a repository maintain an entire domain CRL or part?
- How often should updated CRLs be sent?
- Can CRL updates include only the changed items ("delta CRL")?
- Should updates be sent periodically or on request?

Current guidance on these issues is summarized in section 3.1.

In [26] Rivest proposes new certificate formats and architectures that allow to eliminate CRLs. The are able to issue suggested certificate formate allows to specify two validation periods for a certificate. One of the periods defines the time during which the certificate is guaranteed to be valid by the signer of the certificate. The other one expresses that the certificate is expected to be valid, but an acceptor of the certificate might want to check the revocation status of it in applications that involve high risks. Rivest also discusses architectures in which the client has the burden to prove that the certificate is not revoked. This could be done by intermediate servers that issue new certificates for the client that fit with the server's recency policy for certificates.

### Online Verification

User workstations cannot be expected to store a complete CRL. Instead, when the revocation status of a certificate is required, it sends a status request to a CRL repository, containing the serial number of the certificate in question. The repository sends a signed reply stating that the certificate is or is not valid, as of the latest CRL date. There is a proposed Internet standard protocol for this purpose, OCSP (Online Certificate Status Protocol) (see Section 3.2).

When individual status responses are signed by the repository, the repository must be trusted to report the CRL contents accurately, and its key must be known. It is possible for status responses to be distributed from an untrusted repository, if the CRL is reconstructed as a CRT (Certificate Revocation Tree) signed by a trusted authority. Individual certificate status responses can be extracted from the CRT and authenticated with the original signature, without the need to trust the repository. The CRT concept is discussed in section 4.

### Fault-tolerant Distribution of Revocations, Renewals

In the Web of Trust, or in a network of cross-certified root CAs, revocation notices or certificate renewals may need to be distributed by forwarding them from node to node. If the forwarding action is unreliable due to failures or delays, the distribution protocol should incorporate redundancy to reduce risk. Some research on how to design this kind of distribution protocol using a "depender graph" structure was done by Wright, Lincoln, and Millen [27].

# Chapter 3

# Performance Issues

Each certificate has a validity period. However, the certificate may be revoked before it expires. Therefore, a relying party should validate the status of a certificate to determine whether it has been revoked or not. Normally, CRLs (Certificate Revocation Lists) are used to check the revocation status of certificates [17]. A CRL is a list of the serial numbers of all unexpired certificates that a CA has revoked. The CA updates the CRL periodically, and each relying party retrieves the CRL from the repository of the CA or other distribution points. The relying party can then validate the status of individual certificates from its own copy of the CRL.

CRL distribution gives rise to some problems. The CRL has a validity period indicating how long the relying party may use its CRL before it is required to obtain the updated version from the CA. If the validity period is long, the relying parties can use the cached CRL for a long time. However, it can take a long time before the serial number of a revoked certificate is listed on the relying party's CRL. In order to reduce the difference between the current revoked certificate information and the contents of the relying party CRL, the validity period of the CRL should be short. In this case, the relying parties have to access the repository of the CA or some other validation authority frequently. This may cause a heavy processing load for the repository. Another problem is the size of CRLs. If a CA has revoked many certificates, the size of the CRL for that CA may become large, and transmission of it to relying parties consumes significant bandwidth on the network. Moreover, each relying party may use only a part of the information, and the burden of sending the rest of the large CRL may be wasted.

This section describes the analysis of CRL distribution, and suggests efficient mechanisms to solve the above problems. The OCSP protocol for obtaining the validity of individual certificates, rather than an entire CRL, is also described.

## 3.1 CRL Updates

David Cooper of NIST analyzed the nature of CRL updates using a mathematical model. In his papers[2, 3], he compares several alternative CRL distribution mechanisms, such as segmented CRLs, over-issued CRLs, delta-CRLs, and sliding window delta-CRLs. created models. This section reviews Cooper's approach and results. His models focus on distribution methods that minimize peak loads on the repositories, or the total bandwidth used for transferring CRLs to the requesting relying parties.

In his modeling, the following assumptions are adopted.

- Relying parties request CRLs only when needed to perform a validation. (no-precaching of CRLs)
- Relying parties have perfect caches. (no CRLs are deleted from the cache until they have expired)
- An exponential interarrival probability density is used to model the timing of validation attempts. In this model, the probability that a relying party's first validation attempt will occur in the interval $[t...t + dt]$, in the limit as $dt \to 0$, is

  $$ve^{-vt}dt$$

  where $v$ is the validation rate (i.e., average number of certificates per unit time that a relying party attempts to validate).

### 3.1.1 Traditional Method

This method is specified in X.509[17], and most basic method to distribute the information of revocation using CRLs. The procedures of this method is following.

1. CA issues a CRL periodically, and posts it to a repository.
2. The CRL includes all unexpired certificates issued by the CA that have been revoked.

3. Each CRL includes a nextUpdate field that specifies the time of next CRL issuance.

4. Any relying party requiring certificate status information, that does not already have an unexpired CRL, retrieves the current CRL from the repository. (The paper assumes that all relying parties obtain CRLs from the same repository.)

5. Over the period of time in which a CRL is valid, each relying party will make at most one request to the repository for a CRL.

**Performance**

In Cooper's paper[2], the overall CRL request rate is computed. If a CA issues a new CRL at time 0, the request rate for CRLs from the repository at time $t$ is expressed by the following equation:

$$R(t) = Nve^{-vt}$$

In this equation, $N$ is the number of relying parties, and $v$ is the validation rate.

To obtain illustrative performance values for this method and others in this section, we will define the request load by assuming that there are 300,000 relying parties, each validating an average of 10 certificates per day.

Figure 3.1 shows the request rate for a CRL, issued using the traditional method, over the course of 24 hours, assuming that the CRL was issued at time 0 and no other CRLs were issued during the subsequent 24 hours.

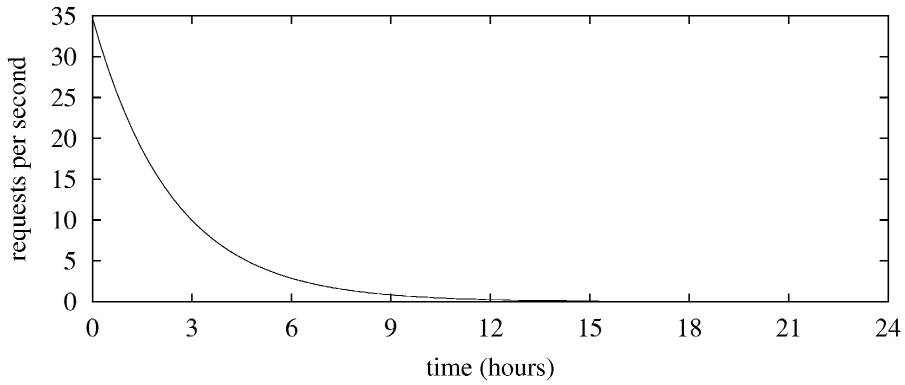



Figure 3.1: CRL request rate, traditional method (reproduced from [2])

As Cooper points out, the problem with the traditional method is that the CRLs cached by every relying party expire at the same time. Immediately after the CRLs expire, and a new CRL is issued, every relying party will need to obtain a CRL from the repository in order to perform a validation. As a result, there is a relatively high request rate when a new CRL is issued, followed by an exponential decline in the request rate.

### 3.1.2 Over-issued CRLs

The idea behind over-issued CRLs is that the CA issues a new CRL before the previous one expires (i.e., before the nextUpdate time of the previous CRL has been reached). Figure 3.2 shows an example of over-issued CRLs. In this figure, each CRL is valid for 24 four hours, but a new CRL is issued every 6 hours.

| | | |
|---|---|---|
| cRLNumber = 1<br>thisUpdate = Mon. 12:00am<br>nextUpdate = Tues. 12:00am | cRLNumber = 2<br>thisUpdate = Mon. 6:00am<br>nextUpdate = Tues. 6:00am | cRLNumber = 3<br>thisUpdate = Mon. 12:00pm<br>nextUpdate = Tues. 12:00pm |
| cRLNumber = 4<br>thisUpdate = Mon. 6:00pm<br>nextUpdate = Tues. 6:00pm | cRLNumber = 5<br>thisUpdate = Tues. 12:00am<br>nextUpdate = Wed. 12:00am | cRLNumber = 6<br>thisUpdate = Tues. 6:00pm<br>nextUpdate = Wed. 6:00pm |

Figure 3.2: Issuance timing of over-issued CRLs (reproduced from [2])

#### Performance

The request rate of over-issued CRLs at time $t$ is

$$R_I(t) = \frac{Nve^{-vt}}{(O-1)(1-e^{-vl/O})+1}$$

In this equation, $N$ is the number of relying parties, $v$ is the validation rate, $O$ is the number of CRLs that are valid at any given time ($O = 4$ in Figure 3.2), and $l$ the length of time that a CRL is valid. This equation applies to each interval of length $l/O$.

Figure 3.3 shows the request rate for over-issued CRLs over the course of 24 hours, assuming that CRLs are issued as in Figure 3.2, using the same load assumptions. In Figure 3.1, the peak request rate was 34.72 requests/second, but the method of over-issued CRLs reduces this rate to 9.25 requests/second.

The only disadvantage of this approach is that the CA must actually bring the distributed CRL up to date at the beginning of each interval. This does not change

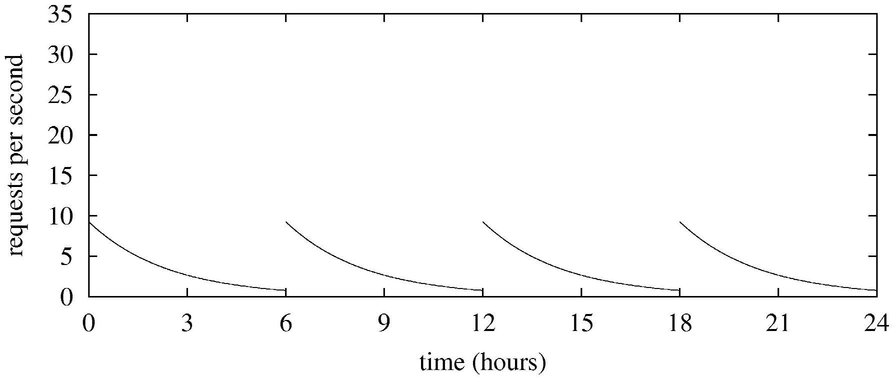


Figure 3.3: Request rate for over-issued CRLs (reproduced from [2])

the overall number of revocations that must be entered, but it does imply that the CA must go online $O$ times more frequently.

### 3.1.3 Segmented CRLs

Another way to improve performance over the traditional method of distributing certificate status information is to segment CRLs. The full CRL is divided into segments, and a request for the status of a certificate results in sending only the segment that would contain that certificate serial number if it were revoked. While segmenting CRLs does not reduce the peak request rate for CRLs, it will reduce the size of each CRL. This allows a repository to service the same number of requests for CRLs with a fraction of the bandwidth.

**Performance**

The request rate of segmented at time $t$ is

$$R_S(t) = Nve^{-vt/s}$$

In this equation, $N$ is the number of relying parties, $v$ is the validation rate, and $s$ is the number of CRL segments. This equation assumes that certificates are allocated to CRL segments at random. The maximum request rate is

$$R_S(0) = Nv$$

as before.

The advantage of the segmented CRLs is to reduce the size of each CRL segment, so that when a bandwidth (bit rate) of $B$ is required the traditional way, only $B/t$ is required for segmented CRLs. However, because the segmented CRL does not have all revoked certificate information, the cached CRL segments may not have the for other certificate validation. In this case, the relying party has to access the repository again to retrieve another segment of the CRL. For this reason, the request rate decreases more gradually compared with Figure 3.1. This also means that the average age of a CRL segment increases; the revocation information is less current.

Multiple CRL repositories and use of the CRL distribution points extension in X.509[17] may reduce the request rate per server in this method. If each segmented CRL was distributed from different servers (or repositories), the request rate for each server would be decreased proportionately. The relying parties can recognize which server has the CRL required for validation of target certificate by referring to the CRL distribution points extension in that certificate.

### 3.1.4 Over-issued segmented CRLs

This is a combination of over-issued CRLs and segmented CRLs. There are two basic ways that over-issuing can be combined with segmentation. The first (case 1) is to issue all CRL segments at the same time, but issue the CRL segments more often than is required by the CRLs' validity period. Figure 3.4 shows the issuing timing of this case. The other way (case 2) is to issue each segment only as often as necessary, but to stagger the issuance of each segment so that the peak request rates for the different segments occur at different times. Figure 3.5 shows the issuing time of this case.

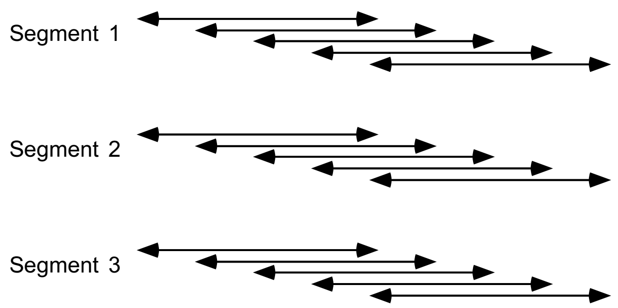



Figure 3.4: Combination of over-issued CRLs and segmented CRLs (case 1)

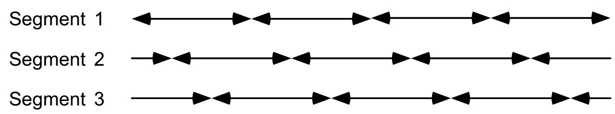



Figure 3.5: Combination of over-issued CRLs and segmented CRLs (case 2)

**Performance**

In order to evaluate the over-issued segmented CRLs, the same scenario as for segmented CRLs is used as an example.

In case 1, if three CRL segments are used and each segment is issued once every 8 hours, then the peak request rate will be only 14.83 requests/second. Furthermore, the more frequently the segments are issued, the more the peak request rate can be reduced.

In case 2, if the issuance of the three CRL segments were staggered by 8 hours, the peak request rate would be only 16.64 requests/second (see Figure 3.6) as opposed to a peak rate of 34.72 requests/second if all three segments were issued at the same time. Unfortunately, the peak request rate does not continue to decline with increasing numbers of segments. With 4 CRL segments issued at 6 hour intervals, the peak request rate increases slightly to 17.15 requests/second. As the number of CRL segments approaches infinity, the peak request rate approaches the peak rate for an unsegmented CRL.

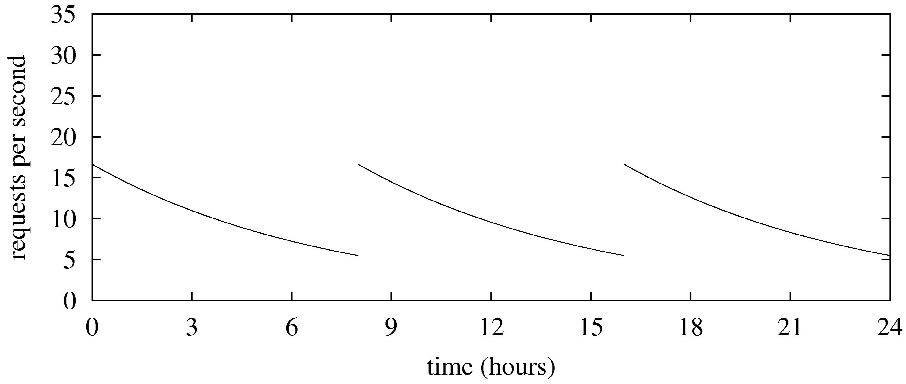



Figure 3.6: Request rate for three CRL segments with staggered issuance (reproduced from [2])

### 3.1.5 Traditional delta-CRLs

Delta-CRLs were introduced in the recent version of X.509[17]. The main purpose of the delta-CRLs is to reduce amount of data transferred to the relying parties. The delta-CRL lists all of the certificates whose status changed since the last base CRL was issued. If the relying party has the base CRL in its cache, the relying party only retrieves the delta-CRL to get the most recent revocation status. Therefore the total bandwidth required to send the revocation information would be decreased, or frequent update of the CRL information would be possible.

With the traditional method for issuing delta-CRLs, a base (or full) CRL is issued periodically and each delta-CRL lists all of the certificates whose status has changed since the last base CRL was issued. Whenever a new base CRL is issued, a final delta-CRL referencing the previously issued base CRL is also issued. Figure 3.7 shows an example of delta-CRLs issued in the traditional manner. In this example, relying parties download base CRLs at most once every 4 hours. Delta-CRLs are then obtained to ensure that validations are based on certificate status information that is at most 10 minutes old.

| 0:00 | 0:10 | ... | 3:50 | 4:00 | 4:10 |
|---|---|---|---|---|---|
| base | | | | | |
| base | delta | | | | |
| ⋮ | | | | | |
| base | | | delta | | |
| base | | | | delta<br>(new base) | |
| | | | | base | delta |

Figure 3.7: Traditional delta-CRLs

#### Performance

The request rate for base CRLs will be determined separately for two types of intervals: intervals corresponding to delta-CRLs issued at the same time as a new base CRL (a “synch” interval) and intervals during which no base CRL is issued (a “non-synch” interval).

If base CRLs are issued $L$ time units apart ($L = 4$ hours in Figure 3.7) then the request rate for base CRLs during a “synch” interval at time $t$ will be

$$R_s(t) = Nve^{-v(t+L)}$$

In this equation, $N$ is the number of relying parties, and $v$ is the relying party's validation rate.

The request rate for base CRLs during "non-synch" intervals is the same as the request rate for CRLs issued in the traditional manner:

$$R_{ns}(t) = Nve^{-vt}$$

where $t$ is the amount of time since the most recent base CRL was issued.

With this approach, a new base CRL must be requested when the relying party has missed the last delta-CRL in the previous synch interval, which happens so often that the peak bandwidth is not reduced significantly. The advantage of delta-CRLs is primarily that the CRL is no more than the short (10-minute) non-synch interval out of date when it is used.

If it is assumed that the base CRLs and delta-CRLs are issued as in Figure 3.7, and there are 300,000 relying parties each validating an average of 10 certificate per day, the paper[3] says that issuing delta-CRLs in the traditional manner only reduces the peak bandwidth by 6.7% over the traditional method of issuing CRLs under the same conditions.

### 3.1.6 Sliding window delta-CRLs

The delta-CRL provides information about all status changes that occurred during a certain "window" of time. The problem with the traditional method of issuing delta-CRLs is that the window sizes of the delta-CRLs vary. If a relying party last obtained fresh certificate status information at time $t$ and obtains a delta-CRL that references a base CRL that was issued at time $t' \geq t$, then the relying party cannot use the delta-CRL to update its local cache without obtaining a new base CRL. But if the delta-CRL includes all updates over a larger window, the relying party will not need a new base CRL until that window is passed. So, the larger the window sizes of the delta-CRLs, the lower the request rate will be for base CRLs. The idea behind sliding window delta-CRLs, then, is for each delta-CRL to have the same, large window size instead of using variable size windows as with the traditional method.

Figure 3.8 shows an example of sliding window delta-CRLs. In this figure, each delta-CRL is valid for 10 minutes but has a window size of 4 hours, relative to the delta-base indicated to its left.

| 0:00 | 0:10 | ... | 4:00 | 4:10 | 4:20 |
|---|---|---|---|---|---|
| base | | | delta<br>new base | | |
| | (delta<br>base) | | | delta | |
| | | (delta<br>base) | | | delta |
| ⋮ | | | | | |

Figure 3.8: Sliding window delta-CRLs

#### Performance

The request rate for delta-CRLs in a system that uses sliding window delta-CRLs is the same as the request rate for full CRLs in a system that issues CRLs in the traditional manner. However, the request rate for base CRLs at time $t$ is substantially reduced. That rate is:

$$R_{s\Delta}(t) = Nve^{-v(t+w)}$$

In this equation, $v$ is the relying party's validation rate, and $w$ is the window size of the current delta-CRL.

Figure 3.9 shows the request rate for base CRLs and delta-CRLs over four hours assuming that base CRLs and delta-CRLs are issued as shown in Figure 3.8. As in the previous graphs, it is assumed that there are 300,000 relying parties each validating an average of 10 certificates per day.

The paper[3] says that if delta-CRLs are issued as in Figure 3.8, the peak bandwidth is reduced to 79.7% over the traditional method of issuing delta-CRLs under some condition.

### 3.1.7 Summary of CRL Updates

At the beginning of this section, several problems for CRLs were described. Cooper [2, 3] made the behavior of repository for CRLs and its problems clear by using the

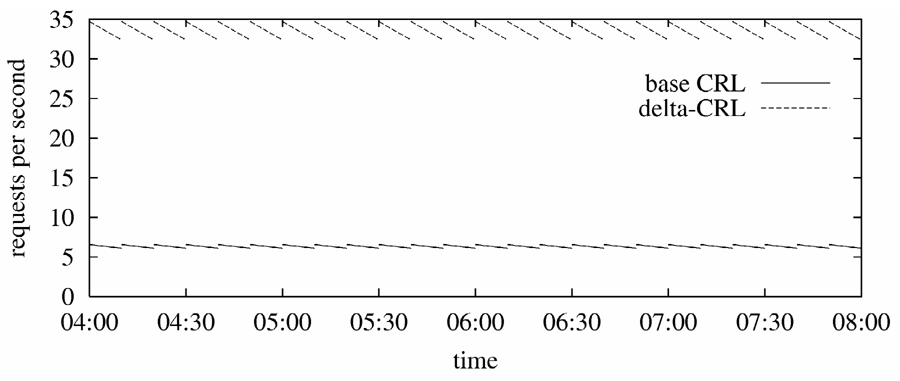



Figure 3.9: Request rate for base and delta-CRLs in Fig. 3.8 (reproduced from [3])

mathematical model. He also proposed several methods to improve the performance of revocation distribution by CRLs, and compared them with the traditional distribution method using the mathematical models.

Table 3.1 compares each method described in this section. *Communication Costs* of the traditional method and over-issued CRLs are almost equal, because over-issued CRLs merely distributes the request timing of the traditional method. Segmented CRLs and over-issued segmented CRLs may decrease the communication costs, because each relying party need not retrieve all CRL segments. Traditional delta-CRLs and sliding window delta-CRLs can decrease the communication cost, because the information included in the delta-CRLs is the difference from the last base-CRL only. However, it should be noted that improvement of the peak bandwidth for traditional delta-CRLs is small.

In terms of *peak request rate*, over-issuing CRLs is a most effective method. In case of sliding window delta-CRLs, this method can decrease the request rate of base-CRLs, but the request rate of delta-CRLs is still high, and revocation status information is older on the average.

In order to provide the latest certificate status information to the relying parties (*frequent update*), the CA should update the CRL often. In this case, traditional delta-CRLs and sliding window delta-CRLs work effectively. Though the peak request rate of these two methods is almost same as for other methods, these two methods can decrease the total bandwidth required. Especially, sliding window delta-CRLs can keep the total bandwidth low at every moment. On the other hand, if a long validity period for CRLs is acceptable (i.e. the CRL is not updated frequently), then over-issued CRLs should be selected to decrease the request rate. The combination of this method and segmentation can decrease the required bandwidth of the network.

Evaluation on the *Many Revoked Certs* line in Table 3.1 compares the performance when the CRL has a lot of revoked certificate information. In this case, a large bandwidth is consumed to send the CRLs. Therefore traditional delta-CRLs and sliding window delta-CRLs have an advantage compared with other methods.

| | TRA | OI | SEG | OI/SEG | ΔCRL | SWΔ |
|---|---|---|---|---|---|---|
| Communication Costs | - | - | + | + | ++ | +++ |
| Peak Request Rate | - | ++ | - | + | - | - |
| Frequent Update | - | - | - | - | + | ++ |
| Many Revoked Certs | - | + | + | + | ++ | +++ |

TRA: Traditional method　OI: Over-issued CRLs
SEG: Segmented CRLs　OI/SEG: Over-issued segmented CRLs
ΔCRL: Traditional delta-CRLs　SWΔ: Sliding window delta-CRLs

Table 3.1: Comparison of several CRL methods

## 3.2 CRLs vs. Online Validity Checking

The Online Certificate Status Protocol (OCSP)[22, 11] was developed and specified by the PKIX working group of IETF for online validity checking. This enables applications to determine the revocation status of an identified certificate. OCSP may be used to satisfy some of the operational requirements of providing more timely revocation information than is possible with CRLs and may also be used to obtain additional status information. In this section, an outline of OCSP and a comparison between CRLs and OCSP are given.

Figure 3.10 shows how a user (the relying party) validates the revocation status of a certificate using CRLs. A certificate authority (CA) issues a CRL to a repository periodically, and the user accesses the repository to retrieve the newest CRL. Figure 3.11 shows how a user validates the revocation status of a certificate using OCSP. The user issues a status request for a specified certificate to an OCSP responder, and receives a response from it. The OCSP responder must obtain the CRL periodically from the CA.

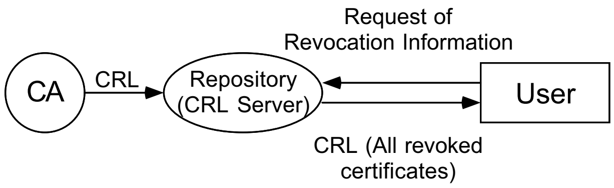



Figure 3.10: Inquiry of revocation status using CRLs

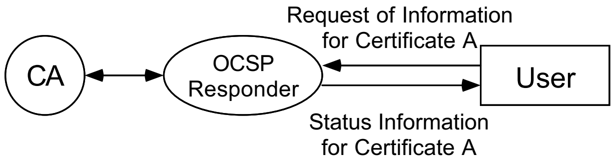



Figure 3.11: Inquiry of revocation status using OCSP

### 3.2.1 OCSP Messages

Basically, there are two kind of message formats for OCSP, *OCSP request* and *OCSP response*. The *OCSP request* message is composed by a user, and sent to an OCSP Responder. Table 3.2 shows contents of this message.

| | |
|---|---|
| Version | Version number of the request syntax. |
| Certificate ID | A list of certificate information that should be validated. |
| Signature | Signature of certificate for a requesting user. (Optional) |
| Extensions | Optional extensions which may be processed by the OCSP Responder. |

Table 3.2: OCSP request message

The *OCSP response* message is composed by the OCSP Responder, and sent to the requesting user. Table 3.3 shows the contents of this message.

All definitive response messages shall be digitally signed. The key used to sign the response must belong to one of the following:

- the CA who issued the certificate in question
- a Trusted Responder whose public key is trusted by the requester

| | |
|---|---|
| Version | Version number of the response syntax. |
| Responder ID | Name of the responder and its key hash. |
| Results | |
| Certificate ID | Identifier of target certificate. |
| Status | Status of target certificate (good, revoked, unknown). |
| Validity | Validity period of this status information. |
| Extensions | Optional extensions used for this status information. |
| Extensions | Optional extensions for this response message. |
| Signature | Signature algorithm information and signature computed across hash of this response message. |

Table 3.3: OCSP response message

- a CA Designated Responder (Authorized Responder) who holds a specially marked certificate issued directly by the CA, indicating that the responder may issue OCSP responses for that CA

### 3.2.2 Performance Comparison

In case of CRLs, users validate the CRL messages signed by a CA. Therefore the computation of the signature is only required when the CA issues the CRLs. However, in case of OCSP, a signature is computed for each response message. Because the computation of the signature requires significant processing time, it may cause a heavy load for an OCSP responder or CA.

Kikuchi *et al.* analyzed the difference in processing time between CRLs and OCSP in his paper[18]. This section summarizes his analysis.

#### Estimation of processing time for one request

In case of CRLs, processing for one request starts at the receipt of the request, and ends after sending the response message that includes a CRL. Therefore, it is assumed that the processing time for the CRL is dominated by the transmit time of the response message including the CRL.

The time required to transmit the response message including the CRL ($T_{CRL}$) is expressed by the following equation.

$$T_{CRL} = \frac{L(s)}{B}$$

In this equation, $L(s)$ is the bit length of the CRL when the number of revoked certificates is $s$, and $B$ is the bandwidth between the users and the repository. If the size for one revoked certificate is $a$ and the size for the fixed length part, such as a signature and serial number, is $b$, the bit length of CRL is

$$L(s) = as + b.$$

Therefore, the $T_{CRL}$ becomes

$$T_{CRL} = \frac{as + b}{B}.$$

In the case of OCSP, the processing time consists of receiving the request, retrieval of the target certificate status, composition of the response message, making a signature for the response, and transmitting the response message. The paper[18] assumes that the processing time for OCSP is dominated by the signature cost. Therefore, the processing time for OCSP ($T_{OCSP}$) used in this paper consists of the computation time of the signature only.

### Comparison of CRLs and OCSP

Figure 3.12 shows the relationship between the number of revoked certificates and the processing time for one request. The assumptions for this figure are the following.

- Computation time of signature for one response message is 0.126 sec. (This value is the computation time of RSA 1024 bit signature using SSLeay-0.9 on a 167 MHz UltraSPARC, and corresponds with $T_{OCSP}$.)
- Bandwidth between the users and repository ($B$) is 1.5 Mbps.
- Size for one revoked certificate in the CRL ($a$) is 224 bits.
- Size for fixed part in the CRL ($b$) is 3608 bits.

Figure 3.12 says that the response time by CRL is faster than the OCSP procedure if the number of revoked certificates is less than 828. However, this result depends on

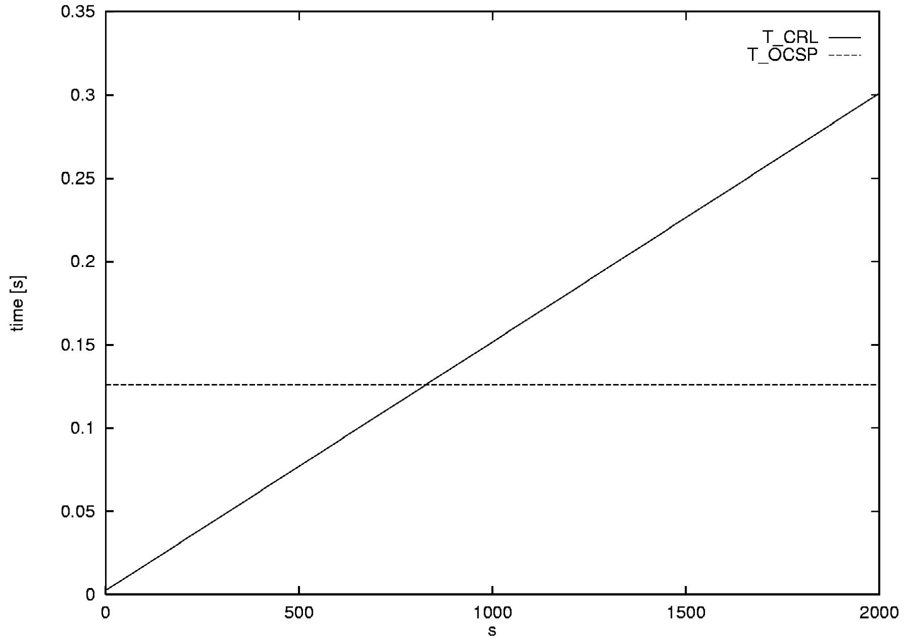



Figure 3.12: Processing time of certificate validation (reproduced from [18])

several variables, such as the bandwidth of the network and the time for signature computation. For example, if the bandwidth of network is increased, the processing time to transfer the CRL would be short. On the other hand, if the computer power is increased, the computation time to make a signature for the OCSP response message would be shorter. Moreover, in the case of CRLs, the retrieved CRL in the user's cache may be reused for another certificate validation until it expires. In this case, if the number of certificates that should be validated was same between CRLs and OCSP, the number of requests by the user would be different between these two methods (CRLs $\leq$ OCSP). Although Figure 3.12 may be useful to compare the nature of CRLs and OCSP, other elements should be considered to choose the validation method and design the PKI system.

### 3.2.3 Summary of OCSP Performance

OCSP provides users with online validation of target certificates. One of the advantages of this protocol is that the response message from the OCSP responder only contains the information for target certificates. In the case of CRLs, because information on all revoked certificate that are not expired is included, the size of the

CRL may be large. This puts puts a greater communications load on the network. Delta-CRLs may reduce the total bandwidth to send the revocation information, but users must have larger storage capacity to store the CRL. By comparison, the OCSP response message is small, and the users do not need much storage space to validate certificates.

However, OCSP has several disadvantages. One disadvantage is the signature cost as described above. The CA or other trusted responder has to sign all response messages. This may increase the processing load of the CA or responder. Another disadvantage is the number of requests. In the case of CRLs, the users may use the cached CRL to validate another certificate during the time that CRL is valid. Though the OCSP responder can provide the status of multiple certificates with one response for the users, another request is required to validate new certificates. Therefore the number of requests for OCSP is greater than for the CRL method if the validation rate is the same.

# Chapter 4

# CRTs and Improvements

In [20, 21] Kocher introduces the concept of Certification Revocation Trees (CRTs) as a means to answer certificate validity requests from a user in a compact way. CRTs are designed as an efficient and scalable structure to distribute information about revoked certificates. A system based on CRTs consists of three components: CAs who produce CRLs, CRT responders analogous to OCSP responders who answer queries about certificate status, and users who request and check the replies. A CRT responder receives CRL updates from a CRL distribution point and uses them to construct and maintain a CRT.

The CRT is digitally signed after it is constructed. The signer could be the CA or an independent validation authority (VA) trusted to construct the CRT correctly from the most recent CRL. The essential advantage of a CRT is that the responder can generate authenticated status responses without computing signatures; it takes advantage of the existing signature on the CRT. Hence it can create replies much more quickly. This also means that a copy of a CRT can be distributed to other directories, which can produce authenticated responses from it without themselves being trusted.

**CRT issuance:**
A CRT is built from a CRL. The serial numbers of revoked certificates are paired into *ranges* of certificates that constitute the leaves of the CRT. For example, if $5$, $12$, and $13$ are serial numbers of revoked certificates, then the following ranges constitute the leaves of the corresponding CRT: $(-\infty, 5), (5, 12), (12, 13), (13, \infty)$. The range $(5, 12)$ means that certificates with serial numbers 5 and 12 are revoked, but any certificate with serial number larger than $5$ and less than $12$ is good. Be-

sides two serial numbers, the range data structure may also include other CRL entry information such as the reason for revocation and date of revocation.

The range structures are hashed and the hash values are used as leaf nodes of a Merkle hash tree. This is a binary tree that is built by concatenating adjacent pairs of hashes at each level and computing the hash of the pair to form the parent node at the next level. Finally, the root hash value is augmented with additional information such as the signer and validity period of the CRT to form the root record, which is digitally signed. Figure 4.1 illustrates the CRT for the example certificate ranges above.

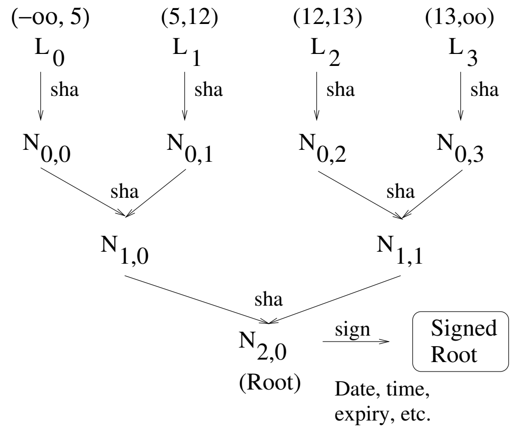



Figure 4.1: Sample CRT

**Confirmation issuance:**
The CRT and the digitally signed root record are distributed to the CRT responder. When the responder gets a validation request for a certificate serial number, it replies with the range containing that serial number, all the hashes that bind that leaf to the root, and the root signature. For instance, a query for the revocation status of certificate number 12 is answered by $L_2, N_{0,3}, N_{1,0}, N_{2,0}$ and the signature. This sequence of hashes is called a *hash chain*, and the full response including the root record is regarded as a *proof* of the certificate status.

**Confirmation verification:**
Upon receipt of the range, hash values, and signature, the user checks whether the range includes the certificate. Then, the proof is checked by the user. For

this purpose, the user computes the hash of the range and uses the other hashes to recompute the root hash, which is compared with the one in the reply. This is combined with the rest of the root record and the signature is checked.

It is possible for a VA to combine CRLs from several CAs into a single CRT. Each CRL determines a consecutive set of ranges. In this case, the range data structure also includes a hash of the public key of the CA that is responsible for certificates in that range.

## 4.1 Unbalanced and *K*-valued Hash Trees

Kikuchi, Abe and Nakanishi discuss in [19] the trade off between communication costs for responses and computation costs for updating the CRT. The computation cost increases with the number of revocations and the communication cost depends on the depth of tree. They investigate two main issues: balanced vs. random trees and binary vs. $k$-valued hash trees.

**Balanced tree vs. random trees:**
Updating a CRT in Kocher's approach always results in a balanced tree since the CRT is totally recomputed from a new CRL. (By definition, a binary tree is balanced if its depth is almost uniform; that is, the path lengths from leaves to root are all the same or differ by only one.)

If unbalanced trees are permitted, one can increase the efficiency of CRT updates when delta CRLs are used. At the time of receiving a new delta CRL, expired certificates may be removed from the tree by merging the corresponding two ranges. If a new certificate is revoked, the corresponding range is split. In both cases, only the hashes on the path(s) to the root are recomputed. Splitting a range extends the depth of the branch it was on by one, and merging two ranges reduces the depth of one branch. The trees that result from this may be unbalanced, and are referred to as *randomly built*.

The cost of updating either a randomly built or balanced tree updating a randomly built tree is linear in the number of modified certificates, but rebalancing takes much longer than random insertion.

The communication cost of a CRT response is proportional to its depth, which increases as the base-2 logarithm of the number of ranges. However, an unbalanced tree has greater average depth, by a factor of about 2 ln(2).

Thus, there is a tradeoff between balanced and randomly built trees. In order to decide which is the better solution, the authors of [19] estimate total processing costs as a weighted sum of update and response times, where the weighting depends on the ration of updates to verification requests. In general, their analysis shows that the preferable technique depends on the size of the CRL. There is a break-even point such that balanced CRTs are better for smaller CRLs and randomly built ones are better for larger CRLs.

In order to find representative results, the authors made some assumptions about processing and communication time. With their figures, a randomly built tree is better when the number of revoked serial numbers is more than about ten times the ratio of verification requests to updates. They stated that a reasonable value for that ratio is 100.

**Binary trees vs. $k$-valued hash trees:**
The second issue deals with the communication costs for binary trees as compared with $k$-valued balanced hash trees, in which the number of branches at each node is $k$. The average depth is the base-$k$ log of the number of leaves, which decreases with $k$. Figure 4.2 shows an example of a 3-valued hash tree.

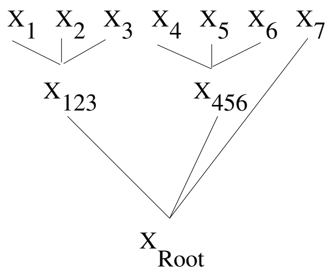



Figure 4.2: Sample 3-valued CRT

The advantage of $k$-valued hash trees is that they produce shorter hash chains as proof for validation queries. Though the average depth of a $k$-valued hash tree decreases with $k$, the number of hash values to be sent increases at the same time, since $k - 1$ hashes must be supplied at each level. The communication cost (in time) is a function of the number of revoked certificates and the value of $k$. In particular, it can be expressed as:

$$C = (\alpha(k-1)\log_k(s) + \beta)B,$$

where $\alpha$ is the length of the hash digest, $s$ is the number of revoked certificates,

$\beta$ is the length of the signed root record, and $B$ is the bandwidth (in bits per second). The authors show that for $\alpha = 64$ and $\beta = 3608$, $k = 2$ optimizes the communication cost.

On the other hand, fewer hashes are stored for a $k$-valued hash tree, and the processing time necessary to recompute the hash tree is less for larger $k$. The authors combine tree-recomputation and response times to arrive at optimal $k$ values, depending on the number of revoked certificates. For instance, they find that, in the case of $512$ certificates, $k = 85$ is the optimal value, and for a list of $1024$ certificates, a 120-valued hash tree shows the overall optimal performance.

## 4.2 Authenticated Search Data Structures

In [23] Naor and Nissim propose yet another data structure for revoked certificates. They propose *authenticated search data structures* as a representation structure for lists of revoked certificates. An authenticated search data structure is a data structure that allows one to efficiently verify the status of certificates and answer a validation request with a proof that confirms the membership or nonmembership of a certificate in a list of revoked certificates. Moreover, such a data structure allows one to efficiently update (insert and delete) revoked certificates. An authenticated search data structure is a special kind of authenticated dictionary (AD), that is, a dictionary that can reply to a validation request with authenticated answer. Therefore, we refer to authenticated search data structures also as AD data structures.

As is the case with CRTs, an AD data structure is constructed from a CRL supplied by a CA. The AD data structure is built by a CA or a trusted validation authority, and it may be distributed to untrusted directories that can respond to queries with authenticated replies.

The AD data structure is a perfectly balanced 2-3 Merkle hash tree with serial numbers of revoked certificates, in order, as leaf nodes. That means each interior node has 2 or 3 children and the paths from the root to the leaves have all the same length. The serial numbers at the leaves are sorted.

In order to prove that a certificate is revoked, the existence of a leaf in the AD data structure that corresponds to that certificate has to be proved with a hash chain. In order to prove that a certificate is not revoked, the existence of two certificates corresponding to two neighboring leaves in the tree with serial numbers surrounding the queried number is proved. This requires two hash chains from adjacent leaves and thus doubles the reply cost.

Figure 4.3 illustrates an AD data structure for the revoked certificates $2, 5, 7, 8$, and 9. The hash chain proof that certificate 6 is not revoked is established by the following values: $H_2, 5$, and $7, H_8, H_9$, in the order given, and the signature on $H_0$. The signature check by the user implies the adjacency of 5 and 7 in the CRL.

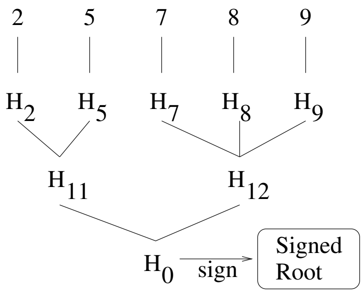



Figure 4.3: Sample AD data structure

If updates need to be distributed to a trusted directory, the CA updates all affected tree node values per insertion or deletion of certificates in the tree. The CA sends a difference list plus a signed new root value, tree height, and time stamp to the directory. (The tree height is necessary to resolve the ambiguity in the choice of 2-3 tree structure with a given number of leaves.) Due to the tree structure, updates are logarithmic in the number of revoked certificates.

Figure 4.4 shows the new 2-3 tree that has 6 as an added certificate. Three new hashes needed to be recomputed and one signature.

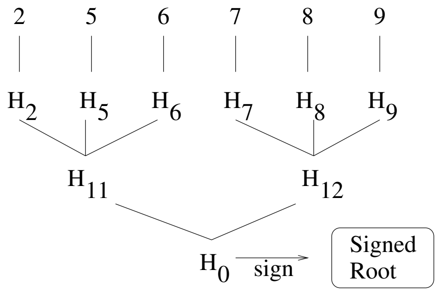



Figure 4.4: AD data structure with added leaf

The overall advantages of AD data structures are scalability, communication cost robustness with respect to parameter changes and the good update rate. Communication costs between the CA and the directory are optimal in this approach because

| | CRT | AD | OCSP |
|---|---|---|---|
| Structure Creation and Server Storage | + | + | ++ |
| Computational Costs (responding to a query) | ++ | ++ | + |
| Communication Costs | + | + | ++ |
| Server Update Processing | + | + | ++ |
| Client Response Processing | + | +/- | + |
| Untrusted Redistribution | + | + | - |

Table 4.1: Comparison of validity checking methods

they are proportional to the number of updates in the revocation list. This allows for high update rates. Another advantage is that the proofs are short and transferable. Moreover, the 2-3 tree approach never requires one to recompute the entire tree when certificate revocations are updated.

## 4.3 Comparison of Validity Checking Methods

We have outlined several certification validation methods: CRT-based approaches, authenticated data search structures, and the online method OCSP. Each of the methods has its advantages. In the following table we give an overview of how these approaches perform with respect to several measurements. We use three symbols for classification: $-, +$, and $++$. The $-$ symbol says that the approach does not do well in the given category in comparison with the others, a $+$ symbol stands for a good evaluation of the approach in the given category, and $++$ points out a clear winner for the given category.

In terms of computational costs for structure creation and server storage, CRTs and the AD data structure are obviously inferior to OCSP because OCSP does not maintain any data structure for revoked certificates other than the CRL, which they all need. Still, a CRT or an AD structure is computed only once from a CRL and only recomputed after CRL updates. For CRTs one has still the choice between balanced and random hash trees as well as binary and $k$-valued hash trees. For the initial creation of a CRT there is no gain for a balanced tree, but the $k$-valued hash trees require in general fewer hash operations.

Nevertheless, the storage overhead for CRTs and ADs pays off when it comes to computation costs for answering a query from a client. In OCSP, computationally expensive signatures have to be computed for every response message, whereas in

CRTs and ADs these are already stored. The same signature is used for responses during the validation period of a the CRT or AD. In the process of answering a certificate validation request, no cryptographic operations need to be computed (like hashing or signatures) since all intermediate nodes of the CRT or AD can be precomputed. Even if an untrusted third party is used for redistributing certification information, then only a few hashes have to be computed (and that is still far more efficient than signing messages as necessary in OCSP). But in the case of untrusted redistribution, CRT supersedes AD because the latter might need to compute two hash chains for a negative answer.

The communication costs in all three approaches are comparable. In the case of CRTs and ADs the query response length is $\log(|CRL|)$ whereas in OCSP it is a signed message with the reply and some further information.

With respect to server update processing OCSP turns out to be the clear winner, since it does not need to update any other structure than the CRL. CRL updates cause updates in the CRT and AD and therefore they are slower than OCSP. Nevertheless, AD data structures supersede CRTs with respect to insertion and deletion of certificates. In an AD data structure updates are logarithmic and, thus, they are even more efficient than updates in CRTs.

On the client side, using a CRT-based method or an OCSP does not have big impact on the client response processing performance. In both bases the client has to check a signature. In the case of CRTs she also has to compute a few hash values. Answers from an AD-based server are possibly twice as expensive to check because in the case of a negative answer two certification chains have to be validated.

Another advantage of CRTs and ADs over OCSP is that they allow for untrusted redistribution of certificate information. Certificate confirmations do not have to come from a trusted third party because they are self-verifying. This allows one to choose the server that issues the proofs in a way that optimizes on performance.

# Chapter 5

# PKI Standards Activities

This section summarizes the useful standards and active standards activities for PKI.

## 5.1 X.500

X.500 is a standard for the structure of an electronic directory in which names, locations and other information about people and organizations is stored using hierarchies of countries, regions, organizations, and individuals. The X.509 standard for certificates and revocation lists is closely tied into X.500. Also, X.509 makes use of the global object naming system used in X.500.

X.500 is an Open Systems Interconnection (OSI) Directory Standard that was first approved in 1988. It was enhanced in 1993 and agreed upon as a standard by ISO and ITU-T [16] (also known as ISO/IEC 9594: Information Technology - Open Systems Interconnection - The Directory). Since X.500 is based on a hierarchical model, the representation of the global X.500 directory is referred to as *Directory Information Tree* (DIT).

Each vertex in the tree is assigned a number. This number is a *relative distinguished name* (RDN) that is unique among its siblings. The concatenation of RDNs from the root to any node in the tree forms a *distinguished name* (DN) for that node. One way of defining a DN is to use an *object identifier (OID)* as specified in X.660 [30] (also known as ISO/IEC 9834-1). The X.660 standard defines valid OIDs that are registered by ANSI. An OID is the name of an object, and it is a value of

ASN.1 type OBJECT IDENTIFIER. An object identifier value is a globally unique ordered list of integers. Each integer in the list is an *object identifier component.* There must be at least two components to form a valid object identifier.

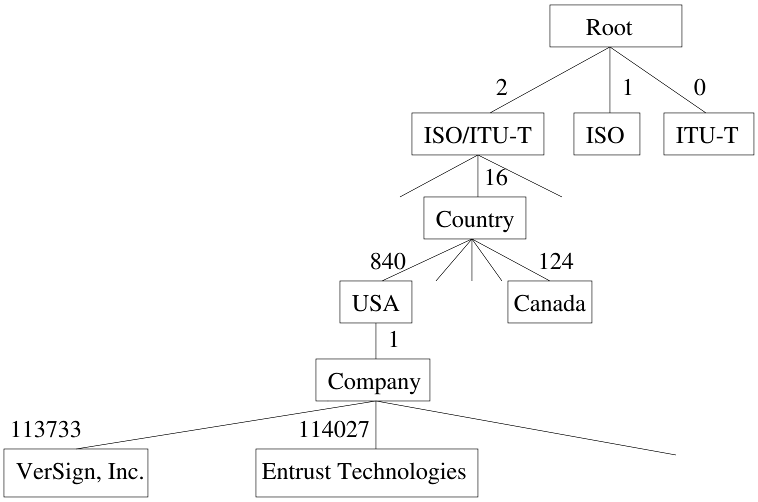
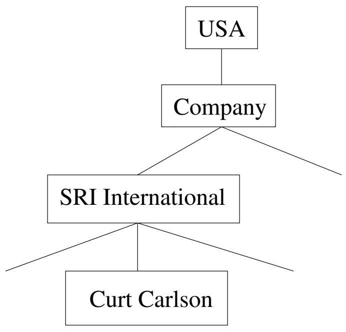



Figure 5.1: DN using Object Identifier

Figure 5.1 illustrates part of the OID structure currently registered with ANSI and defined in X.660. X.660 allows three different children for the root of all values of ASN.1 type OBJECT IDENTIFIER, namely, 0 for ITU-T, 1 for ISO, and 2 for joint ISO/ITU-T object identifier (formerly, joint ISO/CCITT). For instance, 2.16.840 (joint ISO/ITU-T OID) and 1.2.840 (ISO OID) are US object identifiers. The latter one is no longer in use; new organizations are registered under 2.16.840. The X.660 standard was approved jointly by the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC) and the International Telecommunication Union (ITU) in 1992.

An entry in a DIT can have different attributes. For instance, given the DIT in Figure 5.2, the entry for "Curt Carlson" may have the following set of (attribute type: attribute value)-tuples: Common Name: Curt Carlson, Telephone Number: 1 650 859-2878, Mail: curt.carlson@sri.com, Title: President/CEO.

In order to make such a global directory feasible for Internet applications it needs to be distributed by nature. In the X.500 approach distribution is achieved by so-called *Directory System Agents* (DSA). A DSA stores and maintains local information. Locality is a relative term in the sense that it can refer to information of




| | | | |
|---|---|---|---|
| Curt Carlson | 1 650 859-2878 | curt.carlson@sri.com | President/CEO |

Figure 5.2: A simplified entry in a DIT

only part of an organization as well as information of one organization or even several organizations. Users or organizations are responsible for updating their DSA. Each DSA is the database for local information stored using the hierarchical X.500 model. A DSA can exchange information with other DSAs through the use of the *Directory System Protocol* (DSP) of the X.500 recommendation set. This protocol enables DSAs to route requests for information to the appropriate DSA that is in charge of that information. This makes the distribution transparent to the user. The sum of all DSA database constitutes the overall DIT where each DSA holds only some fragment of the overall information base. Access to the Directory services is provided by the so-called *Directory User Agent* (DUA) that supports the functionality necessary for clients to search through the directory. The *Directory Access Protocol* (DAP) controls the communication between the DUA and one or more of the DSAs. The Lightweight Directory Access Protocol (LDAP) developed at the University of Michigan has been developed as a solution for use in the Internet. LDAP is based on TCP/IP.

## 5.2 X.509

X.509 [17, 12] is a certificate framework that supports the authentication of entries in an X.500 directory. X.509 specifies certificates for security key material and certificate revocation lists. Moreover, process methodologies are proposed for the use of certificate authorities (CAs) in managing, certifying, and revoking certificates. Since X.509 is closely tied to the X.500 directory, the CAs are usually arranged in a hierarchy.

### 5.2.1 X.509 Fields

An X.509 certificate is a ASN.1 data type that is a signed sequence of fields. Table 5.1 summarizes the X.509 certificate fields.

| Field Name | Optional | Explanation |
|---|---|---|
| version | | Version 1, 2, or 3 certificate |
| serialNumber | | Integer assigned by CA to the certificate (unique for issuing CA) |
| signature | | Identifies the algorithm used to sign the certificate |
| issuer | | DN of the CA that issued the certificate |
| validity | | time interval for which the information about the certificate is maintained by CA |
| subject | | DN that identifies the entity of the certificate |
| subjectPublicKeyInfo | | a tuple consisting of an algorithm identifier and a bit string representing the public key |
| issuerUniqueIdentifier | Yes | can be used as a unique identifier for the issuer in case of renaming (only version 2 and 3) |
| subjectUniqueIdentifier | Yes | can be used as a unique identifier for the issuer in case of renaming (only version 2 and 3) |
| extensions | Yes | can be used define additional fields (only for version 3) |

Table 5.1: X.509 Certificate Fields

The extension field allows to define new fields without modifying the ASN.1 data types definition. We will come back to these fields in more detail in the next section.

X.509 employs CRLs for certificate revocation. A revocation list is a signed sequence of several fields, including the version of the CRL (only stated if version 2

CRL), the identifier of the algorithm that has been used to sign the CRL, the issuer name, the date and time at which the revocation list was issued, (optionally) the date and time of the next update, an optional list of revoked certificates and revocation list extensions. For each revoked certificate, its serial number, its revocation date and (optionally) revocation extensions are stated. Figure 5.3 illustrates the CRL format used in X.509 Version 1 and 2.

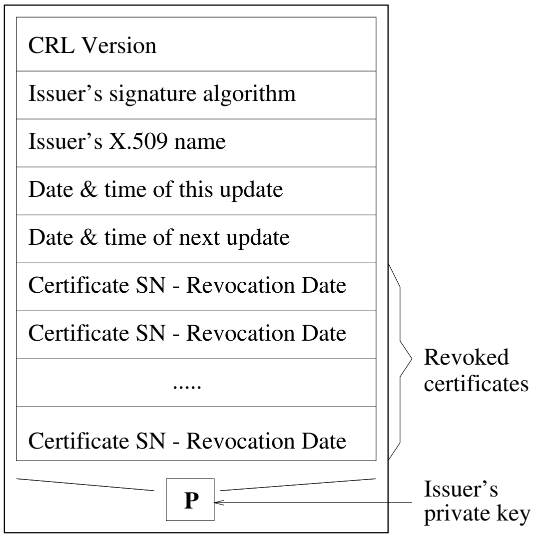



Figure 5.3: CRL version 1/2

As mentioned before X.509 describes a the general hierarchical model. In large scale, widely distributed applications it is not feasible to assume one hierarchy that spans organizations of different countries or even one country. Therefore, cross-certificates have been proposed as a means to establish trust between independent certificate authorities. A cross-certificate is a certificate in which subject and issuer are both CAs of independent domains, that is they are not related by subordinate relationship.

### 5.2.2 X.509 Version 3

X.509 and X.500 were designed to operate in an offline environment. Because of the strict hierarchical structure, versions 1 and 2 of X.509 are well suited for the use within a organization. An organizational CA issues certificates for employees of the enterprise and respects the hierarchical enterprise structure. X.509v3

[13] proposes several certificate extensions as well as CRL extensions to address revocation issues. In essence, the X.509 certificates and CRL formats are made extensible in Version 3.

The optional "extensions" field of an X.509v3 certificate leaves room for the designer to define certificate fields as needed. The extensions field is a sequence of extension fields, each one defined by an extension identifier, a boolean flag expressing whether the extension is critical or not, and an encoding of an extension value associated with the extension identifier. The criticality flag is used in order to decide what to do if an implementation does not recognize an extension. If the flag is set to FALSE, the implementation may ignore the extensions field, but if the extension is critical and not recognized by the processing software, then the certificate is considered invalid. Such an invalid certificate would cause a validation attempt of a signature to fail. The X.509v3 standard allows every organization that has a need for extensions to define them, with the restriction that the extension identifier is defined in accordance with X.660.

Several standard certificate extensions have been proposed in Version 3. Those include key and policy information, subject and issuer attributes, certification path constraints, and enhanced CRL functionality.

#### Version 3 Certificate Extensions

*Certificate policies and policy mapping.* With these extensions information can be conveyed about the intended use of a key and about the policies with which the key has been created. A policy is a document (usually in plain-language) that defines obligations and warranties. Policies indicate security procedures, legal disclaimers or provisions. Certificates may be issued in accordance with one or more policies. A policy might place restrictions on the use of its certificates and their appropriateness for specific purposes. For instance, a policy might express that a key is good enough for email but not secure enough for usage in money-intense financial transactions.

*Subject and issuer attributes.* These extensions support alternative alternative names of various forms (such as email addresses) for a certificate subject and a certificate issuer. Alternative names can also be specified to identify a CRL issuer. These extensions make X.509v3 independent from the X.500 directory.

Besides defining alternative names, these extensions can also be used to convey further information about an entity that assist the certificate user in deriving higher confidence about the authenticity of the entity.

*Certification path constraints.* These certificate extensions are useful for CA-certificates. They express constraints in the presence of multiple certificate policies that can be processed automatically during validation of a certificate path. The constraints may restrict the types of certificates that can be issued by the subject CA or the kinds of certification paths that can grow from the certificate.

The following example and figure 5.4 for a certification path constraint that restricts the type of certificates that occur subsequently in a certification path has been taken from [1].

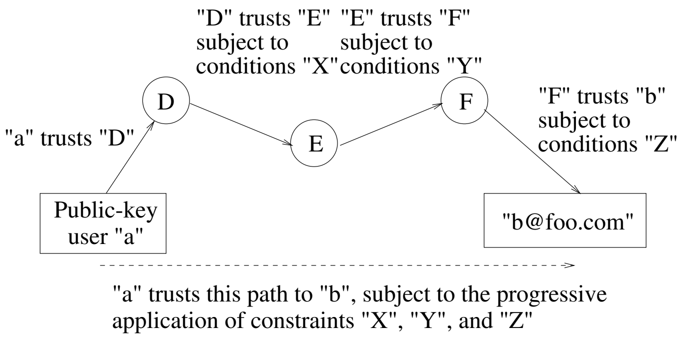



Figure 5.4: Commutativity

User $a$ has $D$ as her certification authority, and thus, completely trusts $D$. $D$ has certified another CA, $E$, only trusting $E$ to issue certificates for other CAs (for instance, $E$ performs some kind of national CA registration). Constraint $X$ would then state that $D$ only trusts $E$ to certify other CAs. $E$ has issued a certificate for CA $F$ stating that it only trusts $F$ to issue certificates for end users in the domain foo.com. So constraint $Y$ would state that $E$ trusts certificates issued by $F$ only if they certify an end user and that user's name is in the foo.com domain. Finally, $F$ issues a certificate for user $b$, but only trusts $b$ for casual email (as opposed to, say, making financial commitments on $F$'s behalf.) So constraint $Z$ state that the certificate issued for $b$ by $F$ should only be used for casual email.

In this way the unlimited trust that $a$ places in $D$ becomes increasingly constrained as the certification path grows. When $a$ obtains a certificate for $b$ she knows that she should only use it for casual email, and she has greater confidence in the strength of the authentication then with, say, PGP's web of trust because she can see how trust has been restricted along the certification path. Given these constraints, she would not accept a certificate issued by $E$ for $b$ (or any other user), nor would she accept any certificates from any certification authority certified by $F$. If CAs define the tightest practical conditions when they certify other CAs, then as a certification path grows it becomes progressively more constrained until it can grow no longer.

### Version 3 CRL Extensions

*CRL number and reason code.* This extensions indicate revocation reasons and assign to each CRL a monotonically increasing number. This allows users to determine if a CRL was missed.

*CRL distribution points.* These extensions combine revocation information from several CAs into one CRL. The overall aim is to reduce the size of CRLs processed by a CA's users. The CA can partition the CRL in some way and issue partitions from different distribution points. This way, the user does not have to accept full CRLs that contain information she is not interested in. For example, a corporate CA might issue a different CRL for each division of the company. A user who wants to verify a certificate for an employee of a particular division only needs to check the division's CRL. There are other ways of partitioning CRLs. For instance a CA might split revocation information due to revocation reason. Routine revocations like those that occur when employees change their name can be stored separately from revocations that are due to security compromises. Splitting CRLs has also the advantage of updating them in different intervals.

*Delta-CRLs.* Delta-CRLs are another means of reducing the size of CRLs. Rather than issuing a full CRL (or a full partition of a CRL) the CA only announces a list of changes that occurred since the last CRL issuance. Users update their own CRL database with the information in the delta-CRL in order to keep an up-to-date CRL. Downloading and processing delta-CRLs saves bandwidth and computing time compared to full-CRLs.

*Indirect CRLs.* This extension allows a CRL to be issued from an entity other than the CA that issued its certificates. The underlying idea is that a distribution CA gathers information from multiple CAs and provides revocation information for all of them.

## 5.3 Organizations and Their Goals

The objectives of relevant standards organizations are summarized here briefly.

### 5.3.1 ISO/ITU

The International Organization for Standardization (ISO) [14] is a non-governmental organization. Its a worldwide federation of national standardization committees. ISO's work results in international agreements which are published as International Standards.

The International Telecommunication Union (ITU) [15] is an international organization that enables coordination on global telecom networks and services. The ITU-T is a subgroup of ITU that is concerned with standardization of telecommunication technology.

### 5.3.2 PKIX

The IETF (Internet Engineering Task Force) Working Group PKIX [25] was established in 1995. PKIX adopted the X.509 and aims at developing an Internet standard that is based on X.509. Several informational documents and standards were produced by this working group. The standard RFC 2459 [12] suggest the X.509 version 3 certificates and version 2 CRLs for use in the Internet. PKIX also proposed the Online Certificate Status Protocol (OCSP) (RFC 2560) [22].

The working group is now embarking on additional standards work to develop protocols that are either integral to PKI management, or that are otherwise closely related to PKI use. Work is ongoing on alternative certificate revocation methods. There also is work defining conventions for certificate name forms and extension usage for "qualified certificates," certificates designed for use in (legally binding) non-repudiation contexts. Finally, work is underway on protocols for time stamping and data certification. These protocols are designed primarily to support non-repudiation, making use of certificates and CRLs, and are so tightly bound to PKI use that they warrant coverage under this working group.

### 5.3.3 SPKI/SDSI

The IETF Working group SPKI/SDSI develops Internet standards for public key technologies including certificate formats, signature formats, and key acquisition protocols.

SPKI stands for Simple Public Key Infrastructure [29, 7, 8] and SDSI stands for Simple Distributed Security Infrastructure [28]. Originally independent efforts, the groups recently joined forces. SPKI/SDSI addresses trust management issues. According to their philosophy, digital certificates that bind names to public keys are not appropriate to support different trust models. The name is only one attribute of a key holder. A key holder might have different names under which she is known to different groups. SPKI/SDSI certificates are therefore not based on global name spaces, but local name spaces which might be connected with each other. Besides greater flexibility in name-key bindings, it is also of great importance to be able to express the specific privileges a key holder has, credentials that are held, and which authorization have been granted to the key holder. SPKI/SDSI certificates also carry a validity period. Thus, if a SPKI/SDSI implementation employs CRLs for certificate revocation, the certificates need to have a reference to the location of the CRL. As of the date of this report no information was available as to which protocols will be used for on-line validations.

The SDSI 2.0 design represents the merger of SDSI and SPKI. It has a unified treatment of certificates, a coherent treatment of names (both for individuals and for groups), an algebra of ”tags” for describing permissions and attributes, and a flexible means of denoting cryptographic keys.

# Chapter 6

# Vendor Solutions

There are now several commercial vendors of public key certificate services and technology. The products supplied by these vendors enable an organization to generate and store public key certificates for its employees or other user community. These certificates can be assigned and customized according to the policies of the customer organization.

Some vendors operate their own domain with a root CA and subordinate hierarchy. Others merely supply software with which a customer can set up an on-site organizational CA and related services. These options can be combined by giving the customer an organizational root CA, but providing services such as

- certification of the customer root CA by the vendor root CA, and
- backup or publicly accessible directoory services.

Another role for a vendor is to act as an independent validation authority. It can collect certificate revocations from several CAs, and reply to customer queries regarding certificate validity.

## 6.1 Web Server and Browser Capabilities

Internet access from a desktop computer is usually accomplished using a browser such as Netscape Navigator or Microsoft Internet Explorer. A browser is a client that communicates with servers using HTTP as the application protocol over TCP/IP as the end-to-end protocol.

Recent versions of Netscape Navigator, for example, provide for secure HTTP sessions through support of X.509 certificates and SSL. SSL is a protocol that runs as a sublayer between TCP/IP and the application layer to distribute keys and use them to encrypt or authenticate data. The browser incorporates a module for cryptographic services from RSA Data Security implementing the proprietary PKCS #11 formats and services, including RSA public key encryption and DES symmetric key encryption. Some background on Netscape browser capability is available at `http://www.netscape.com/security`.

The Navigator browser stores certificates, checks signatures on received certificates, and generates public, private, and shared keys. Using SSL, a browser can authenticate a server certificate, generate a symmetric key, and use it to encrypt subsequent data communication with the server. The browser can securely send a generated public key to a registration authority to obtain a certificate for that key signed by the associated CA.

The browser also supports accss to X.500 directories using LDAP (Lightweight Directory Access Protocol). Users can enter LDAP URLs (URLs beginning with the "ldap://" prefix) in Navigator browser windows to search an LDAP directory. By default, Netscape Communicator uses standard attribute names (which are described in the X.520 standard and in the LDAP protocol) when searching a directory. To use a directory schema with different attribute names, the user can specify the customized attribute names in a preferences file.

While browsers are intended to interoperate with any server following the protocol standards, Netscape offers its own directory and server products, including a certificate server to create and store public key certificates. There is also a customizable product called "Network Security Services" (NSS) that was once a commercial Netscape product but is now available from mozilla.org as open source software. Their site is `http://www.mozilla.org/projects/security/pki/nss`.

## 6.2 VeriSign

VeriSign, Inc. provides Internet trust services. VeriSign acts as a root CA with a hierarchy of subordinate CAs that are either local to VeriSign or remotely located at user sites. Netscape browsers are shipped with pre-cached Verisign CA certificates.

VeriSign provides digital certificate services for companies that want to deploy digital certificates in their organization. These services include creation, renewal, recovery, and storage of certificates. VeriSign also manages the distribution of CRLs to enterprise user directories. Verisign supports X.509 certificates, and handles the customized versions of these certificates used by SSL, S/MIME, SET, and IPSec.

VeriSign also offers on-site support for companies. They enable customers to install a customized enterprise CA for issuing digital certificates, and help setting up registration authorities and directories and implementing administrative functions. The on-site PKI can be either self-contained or supported by Verisign directories and services for certificate backup and distribution.

VeriSign provides a variety of tools and software components for other related secure functions such as VPN deployment, various secure B2B web applications, and secure email for Lotus Notes R5 and Microsoft's Exchange.

Their web site is `http://www.verisign.com`. More detailed information is found in a sub-page, `http://www.verisign.com/repository`.

## 6.3 Entrust

Entrust Technologies provides security product solutions for PKI, secure email, web applications like e-commmerce portals, file encryption, wireless applications, and VPNs. Entrust Technologies operates in one of the two following modes:

- It installs proprietary software for an enterprise PKI or
- It manages security services for companies through web pages. In this case company employees can register themselves over the given interface web pages according to company policies.

Entrust and Valicert have a "strategic relationship" to augment Entrust's PKI products and services with Valicert's certificate validation services. The Entrust web page is `http://www.entrust.com`.

## 6.4 Xcert

Xcert sells the following products related to PKI, to allow customers to set up their own on-site root CA and associated directory services.

**Sentry CA** This is a certificate authority. This provides certificate issuance, repository, and key management services. It is designed to maintain performance when scaled, supporting massive demand for signing operations, PKI queries, and large-scale certificate storage and management. Sentry CA is designed for fault tolerance, load balancing, and component redundancy. It simultaneously supports PKI over multiple protocol interfaces such as LDAP, SSL-LDAP, HTTP, and HTTPS. Sentry CA provides an integrated directory and publishes certificates to any other standards-based directory.

**Sentry RA** Sentry RA (Registration Authority) works with the Sentry CA to verify the credentials of certificate requests and to provide cerftificates to clients.

**WebSentry** This is a plug-in module that works with Sentry CA to PKI-enable web servers.

**Xcert Development Kit** This provides an Application Programmer's Interface (API) that enables programmers to enroll public keys, retrieve certificates, retrieve CRLs (Certificate Revocation Lists), check the status of certificates, and verify certificates.

Xcert also offers PKI support and consulting ervices, such as installation of PKI products, integration of PKI products, training, development, and so on. Their web site is `http://www.xcert.com`.

## 6.5 MasterCard/Visa/SET

The SET Secure Electronic Transaction LLC is a company created to support the payment authorization protocol developed jointly by MasterCard and Visa. This protocol uses public key certificates to identify merchants and cardholders. There is a SET Root CA, and financial institutions act as subordinate CAs. Their web site is `http://www.setco.org` (not .com).

The SET architecture uses computer systems called *payment gateways* to authorize payment requests using the protocol and the supplied certificates, and to interact with a financial network to implement the payments. A payment gateway also has an associated CA and key pair.

SET uses X.509 certificates with a customized format for subject names, including such components as account number and "promotional name." There are also private extensions and conventions about how standard extension fields are used for different types of SET certificates.

SET certificates can be revoked or cancelled. Revoked certificates are placed on a CRL maintained by a payment CA. Cancelled certificates are recorded in various local databases.

## 6.6 Valicert

Valicert is a validation authority (VA), which performs revocation checking and certain other services. It is not a CA, but it must be regarded as trusted with respect to its validity responses, which it signs. Valicert has a “strategic relationship” with Entrust to support Entrust CAs and services, but the Valicert VA can support other CAs as well. Valicert can provide customers with their own enterprise or allow them to use the Valicert global VA service.

Valicert’s core technical approach is to maintain a certificate revocation tree (CRT) reflecting CRL information sent to it by those CAs whose revocations it handles. The Chief Scientist of Valicert is Paul Kocher, inventor of the CRT. While Valicert supports OCSP, it has a proprietary protocol it uses to convey CRT responses.

The VA public key used to check validity response signatures can be either preconfigured into user client software, or certified with the key of the CA that provided revocation information. The first case is called “direct trust,” the second “delegated trust.”

A Valicert VA can also provide other PKI-related services such as document archival, time stamping and notarization. It can validate or search for certificate chains linking a user certificate to a known CA. Their web site is `http://www.valicert.com`.

# Bibliography

[1] M. Branchaud. A Survey of Public Key Infrastructures. Master thesis, Dept. of Computer Science, McGill University, Montreal, 1997. http://home.xcert.com/~marcnarc/PKI/thesis.

[2] D. A. Cooper. A Model of Certificate Revocation. In *Proc. of the Fifteenth Annual Computer Security Applications Conference, ACSAC '99*, pages 256–264, December 1999. http://csrc.nist.gov/pki/PKImodels/welcome.html.

[3] D. A. Cooper. A More Efficient Use of Delta-CRLs. In *Proc. of the 2000 IEEE Symposium on Security and Privacy*, pages 190–202, May 2000. http://csrc.nist.gov/pki/PKImodels/welcome.html.

[4] Crypto++ 3.1 benchmarks. http://www.eskimo.com/~weidai/benchmarks.html.

[5] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, IT-22(6):644–654, 1976.

[6] J. Ellis. The possibility of secure non-secret digital encryption. UK Communications Electronics Security Group, January 1970.

[7] C. Ellison. SPKI Requirements. RFC2692 , IETF Simple Public Key Infrastructure, September 1999. http://www.ietf.org/rfc/rfc2692.txt.

[8] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC2693 , IETF Simple Public Key Infrastructure, September 1999. http://www.ietf.org/rfc/rfc2693.txt.

[9] B. Fox and B. LaMacchia. Certificate Revocation: Mechanics and Meaning. In R. Hirschfeld, editor, *Advances in Cryptology: Proceedings of Financial Cryptography '98*. Springer, 1998. LNCS1465.

[10] C. Gunter and T. Jim. Generalized Certificate Revocation. In *ACM Symposium on Principles of Programming Languages*, 2000.

[11] P. Hallam-Baker. OCSP Extensions. Technical report, Internet Draft, September 1999. http://www.ietf.org/internet-drafts/draft-ietf-pkix-ocspx-00.txt.

[12] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure. Certificate and CRL Profile. RFC 2459, IETF, January 1999. http://www.ietf.org/rfc.rfc2459.txt. Basis for [13].

[13] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure. Certificate and CRL Profile. Internet Draft , IETF, March 2000. http://www.ietf.org/internet-drafts/draft-ietf-pkix-new-part1-01.txt. Update to [12] defining X.509 Version 3 certificates and X.509 Version 2 CRLs.

[14] International Organization of Standardization. http://www.iso.ch.

[15] International Telecommunication Union. http://www.itu.int.

[16] International Communication Union (ITU). Series X Recommendations: X.500 and up. http://www.itu.int/itudoc/itu-t/rec/x/x500up/index.html.

[17] ITU-T. Draft revised ITU-T Recommendation X.509. ISO/IEC 9594-8: Information Technology – Open Systems Interconnection – The Directory: Public-Key and Attribute Certificate Frameworks. Recommendation || International Standard , ITU-T, 2000. ftp://ftp.bull.com/pub/OSIdirectory/4thEditionTexts/X.509_4thEditionDra%ftV2.pdf and http://www.itu.int/itudoc/itu-t/rec/x/x500up/index.html.

[18] H. Kikuchi, K. Abe, and S. Nakanishi. Dynamics Analysis on Public-Key Certificate and Limitations of Online Verification Protocols (in Japanese). In *Proc. of the 1999 Symposium on Cryptography and Information Security (SCIS'99)*, volume 2, pages 615–620, 1999. http://www.ep.u-tokai.ac.jp/~kikn/.

[19] H. Kikuchi, K. Abe, and S. Nakanishi. Performance Evaluation of Certificate Revocation Using $K$-Valued Hash Tree. In M. Mambo and Y. Zheng, editors, *ISW'99*, pages 103–117. Springer, 1999. LNCS 1729.

[20] P. Kocher. A Quick Introduction to Certificate Revocation Trees (CRTs). http://www.valicert.com/html/more_info.html.

[21] P. Kocher. On Certificate Revocation and Validation. In *Financial Cryptography*, volume LNCS 1465, pages 172–177, 1998.

[22] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure: Onlice Certificate Status Protocol - OCSP. Technical Report RFC2560, IETF, June 1999.

[23] M. Naor and K. Nissim. Certificate Revocation and Certificate Update. *IEEE Journal on Selected Areas in Communications*, 18(4):561–570, 2000.

[24] IEEE p1363: Standard specifications for public-key cryptography. http://grouper.ieee.org/groups/1363/P1363/index.html.

[25] Public-Key Infrastructure (X.509) (PKIX). http://www.ietf.org/html.charters/pkix-charter.html.

[26] R. Rivest. Can We Eliminate Certificate Revocation Lists? In *Financial Cryptography*, volume LNCS 1465, 1998.

[27] R.Wright, P. Lincoln, and J. Millen. Efficient fault-tolerant certificate revocation. In *Conference on Computer and Communication System Security*. ACM, September 2000.

[28] Simple Distributed Security Infrastructure (SDSI). http://www.toc.lcs.mit.edu/~cis/sdsi.html.

[29] Simple Public Key Infrastructure (SPKI). http://www.ietf.org/html.charters/spki-charter.html.

[30] Joint ISO/ITU-T Object Identifier. http://asn-1.com/x660.htm and http://www.alvestrand.no/objectid/2.html and http://www.furniss.co.uk/maint/regist/index.html.

[31] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.