FFI-rapport 2010/02499

Optimization of protocol operations in a Public Key Infrastructure

Anders Fongen

Norwegian Defence Research Establishment (FFI)

13 December 2010

FFI-rapport 2010/02499

1174

P: ISBN 978-82-464-1866-7
E: ISBN 978-82-464-1867-4

Keywords

PKI

Informasjonssikerhet

Digital signatur

Offentlige nøkler

Sertifikathåndtering

Optimering

Approved by

Ole Erik Hedenstad	Project Manager
Vidar S. Andersen	Director

English summary

The protocol operations in a Public Key Infrastructure may be optimized in different ways. This report offers an introduction to the problem area, an analysis of the scalability properties of the relevant PKI operations, and an investigation of a range of proposed techniques for the reduction of overhead in the PKI operations related to certificate dissemination and revocation control.

The report concludes that the use of “short lived certificates” has the best effect on a PKI, in the form of reduced transport volume, simple infrastructure, reduced connectivity dependency and a intuitive service semantics. A part of the recommendation is that the sender should include its certificate in a signed message.

Sammendrag

Protokolloperasjonene i et *Public Key Infrastructure* (PKI) kan effektiviseres på ulike måter. Denne rapporten inneholder en innføring i problemområdet, en analyse av skaleringsegenskapene til de aktuelle PKI-operasjonene og en studie av et antall foreslåtte teknikker for å redusere ressursbruken knyttet til sertifikatfordeling og -kontroll.

Rapportens konklusjon er at *kortlevde sertifikater* tilbyr PKI-operasjoner med de laveste krav til båndbredde og fremkommeliget i nettverket. Kortlevde sertifikater har desssuten en intuitiv virkemåte og stiller enkle krav til infrastrukturen. Konklusjonen anbefaler også at en avsender alltid bør inkludere sitt nøkkelsertifikat i en signert melding/dokument.

Contents

Preface 7

1 Introduction 9

1.1 PKI resource requirements 9

1.2 PKIX standards 9

1.3 The use of COTS software 10

2 Motivational background 11

2.1 Symmetric cryptography 11

2.2 Asymmetric or Public Key cryptography 12

2.3 Why certificates? 13

2.4 Public Key Certificates 14

2.5 Important PKI components and operations 19

2.6 Summary 22

3 General optimization techniques 22

3.1 Active replication 23

3.2 Passive replication 23

3.3 Information dissemination 24

3.4 Network path reduction 25

3.5 Message size reduction 25

3.6 Service/component co-location 26

4 Scalability analysis of validation scenarios 26

4.1 Online certificate validation 27

4.2 Cache hit/miss analysis 28

4.3 Validation based on revocation lists 31

4.4 A calculation example 35

4.5 Short lived certificates 36

4.6 Related studies 38

4.7 Summary 40

5 Specific optimization techniques 40

5.1 Push distribution of revocation list 40

5.2 Delta CRLs 43
5.3 Timely caching of OCSP results 44
5.4 Short lived certificates 45
5.5 Compressed message signature structure 49
5.6 Issues of connectivity, cross-domain operation and multi tier CA 50
5.7 Conclusion 52

6 Summary and conclusions 53

Preface

The use of public key based cryptographic algorithms enables new applications for encryption and digital signatures, and offers solution to a range of authentication and information integrity problems. The cause of concern regarding public key cryptography is the amount of computational and network resources required.

The research from FFI project 1174 GISMO offers scalability analysis and optimization techniques for a Public Key Infrastructure (PKI). There is a general concern over scalability properties of PKI systems, and this report provides analyses and suggestions on improvements.

Some of the improvement techniques require changes to the software in use or modified security policies. These requirements will be kept to a minimum and clearly marked in the report.

The report may be read in its entirety, or in single chapters. It has been written with different audiences in mind, and the chapters recommended for reading will depend on the background of the reader:

- Readers who are familiar with the principles, operations and data elements of a PKI may skip Section 2.
- Readers who are familiar to the theories of distributed computing regarding scalability properties may skip Section 3.
- Readers not interested in the formal background for the investigation of optimization alternatives could skip Section 4, but that would reduce the understanding of the conclusions.

The report presents its final conclusions based on a set of estimated environmental and operational parameters. The validity of the conclusions under different parameter values must be assessed by the reader. Therefore, this report is written with the education of the reader in mind. Some of the sections of the report presents background information and techniques which are considered helpful for the reader in this assessment process. The sections are written so that they may be read separately.

1 Introduction

The term *Public Key Infrastructure* (PKI) refers to the set of services, software and protocols necessary to manage a set of public key pairs (private/public key). The services offered by a PKI is normally associated with subject identification, key generation, key deployment, certificate generation, key revocation, certificate directory service and certificate status provision.

1.1 PKI resource requirements

These services require a high amount of system resources, in terms of

- networking capacity (data volume and number of protocol round trips),
- networking connectivity (frequency of protocol invocations),
- software maintenance (cost of configuration and update)
- computing power (cost of necessary calculations)

For most information systems, there is a requirement that the same set of protocols and services are available for all computing nodes throughout the network, including security services for information protection and authentication.

Whilst a business network tends to be rather homogeneous in terms of user equipment and network capacity, a military network will span over a greater range of equipment. In the lower echelons the use of wireless network communication will be prevalent, and the mobility of the communication equipment will create a *dynamic* network.

Information services which are offered for use in military tactical networks must therefore be designed to operate under the conditions that characterizes these networks.

1.2 PKIX standards

PKI services may be implemented using the PKIX standards [1]. These standards describe protocols and data structures relevant to the PKI services. The PKIX standards are governed by IETF and describe important PKI properties like:

- Data structure of digital public key certificates (X.509)
- Data structure of certificate revocation lists
- Procedures for certificate validation
- Protocols for a revocation status service (OCSP)

[1] See PKIX charter at http://datatracker.ietf.org/wg/pkix/charter/

In addition to the standards published by PKIX, the PKI operation may rely on a larger set of underlying IETF standards, or on standards published elsewhere:

- General IETF standards like HTTP, LDAP etc.
- Public Key Cryptography Standard (PKCS), published by RSA Data Security. A selection of these standards are used for transportation of key pairs and certificates.
- Standards for signature representation (XML-DSig, S/MIME etc.)

Please keep in mind that the use of these standards are not *essential* for the operation of a PKI, they merely support the interoperability aspects so that independently developed software may communicate successfully. The PKI services and related data structures can be implemented with any suitable data structures and protocols.

1.3 The use of COTS software

The reason why PKIX standards are important, however, is because commercial off-the-shelf (COTS) software are using them for the purpose of message signing, message encryption and certificate validation. Alternative architectures may perform better, but that would require custom built end-user software if they were to replace the PKIX protocols. For this reason, the PKIX protocols are unlikely to be replaced.

The purpose of this report is to investigate optimization techniques which may mitigate possible performance problems related to PKIX. The optimization techniques should, to the extent possible, be applicable in an environment where standard software for information management is used, e.g. for messaging and document management. The proposals should not require changes to existing security policies unless unavoidable.

It is necessary to keep in mind how the performance properties of a PKI interact with its security properties. A relaxed distribution mechanism may be more effective from a performance perspective, but also delay the necessary actions when e.g. keys are compromised.

The remainder of the report will be organized as follows: First, a technical introduction to public key systems and related key management applications will be given. Then, a general discussion of the general repertoire of optimization techniques within distributed computing will be presented. Then, a formal framework for scalability analysis of certificate validation operations is proposed. Furthermore, a range of suggested techniques for PKI optimization will be discussed and analyzed from a scalability and connectivity perspective. The report will conclude with a summary and final recommendations for further work.

2 Motivational background

This section provides a short explanation to the technical principles of public key cryptography, certificates, signature generation, verification and validation. Parts of this section are copied from an earlier FFI report 00278/2008 [7].

The explanation is intentionally kept brief. For a fuller discussion, please see [7] or a computing security textbook.

2.1 Symmetric cryptography

The traditional form of cryptography uses the same key for sender and receiver, it is therefore called *symmetric* cryptography. Symmetric cryptography (or simply crypto) can be formally expressed as:

$$C = E(P)_K \tag{2.1}$$

$$P = D(C)_K \tag{2.2}$$

Where C denotes the cipher text (unreadable), P denotes the plain text, K denotes the common key and E and D denotes the functions for encryption and decryption, respectively.

Received crypto text C which is successfully decrypted with key K to meaningful information P can only have been made by an actor with knowledge of K, and have not been altered during transport. Others may receive or eavesdrop C but will not be able to understand the content unless they know the crypto key K. Consequently, symmetric crypto may protect transport integrity, confidentiality and authenticity to the extent described by the assurances of symmetric cryptography.

2.1.1 Assurances of symmetric cryptography

Symmetric crypto offers the following guarantees:

- The encrypted information C cannot be understood by anyone
- C can only be transformed to plain-text P with the knowledge of the crypto key K
- If C is modified, the resulting decrypted $D(C)_K$ will be unintelligible. Modification of C will therefore not remain undetected by a human recipient.
- It is *computationally infeasible* to compute K through observation of C or P, given good cryptographic algorithms and key generation.

In symmetric crypto the shared key represents an authorization to take part in “trusted” communication. Symmetric crypto offers a form of authentication as well, since a valid P only can be made by someone who possesses the corresponding C.

2.1.2 Key management of symmetric cryptography

No trusted communication can take place until the parties have agreed upon a shared key, which must be generated and distributed to the members through a secure ("out of band") channel.

Keys should be replaced regularly and also when they may be compromised. New keys must be distributed using secure separate channels (one cannot use old keys to protect the distribution of new keys). When a member leaves a group that uses a shared secret key, a new key must be generated and distributed to the new set of group members. The cost of this process scales with the square of group size[2].

Key management poses the biggest problem in symmetric crypto. Every user or node must possess a secret key for every (pairwise or group-wise) trusted communication channel. New keys must be generated and distributed often and impose a potential large cost (e.g. a courier service).

2.2 Asymmetric or Public Key cryptography

Asymmetric cryptography uses different keys for the encryption and decryption process. The two keys are mathematically related and are denoted *public* and *private* key.

$$C = E(P)_{K_{pub}} \tag{2.3}$$

$$P = D(C)_{K_{priv}} \tag{2.4}$$

It is computationally infeasible to find K_{priv} given P, C, and K_{pub}. K_{pub} can therefore be known by everyone in order to encrypt plain text P into cipher text C. The cipher text can be decrypted only through knowledge about K_{priv}, which is assumed to be secret to the owner of the key pair.

Anyone can now send encrypted messages to a person given his/her public key. The resulting cipher text can only be decrypted by the person's private key (which is secret to the person).

A key pair may also be used to generate *electronic signatures* through an operation where the sender's *private key* is used in a signature algorithm $C = S(P)_{K_{priv}}$. The resulting signature (also called C) may be verified through the use of the sender's public key $P = V(C)_{K_{pub}}$. Based on the assumption that only the owner of the key pair is able to produce $S(P)_{K_{priv}}$ anyone can verify that this person is the originator of a message that can be transformed to plain text through verification with the user's public key.

The RSA algorithm [11] has the following property:

$$P = E(D(P)_{K_{priv}})_{K_{pub}} \tag{2.5}$$

Consequently, RSA may be used both for encryption and signature. Other algorithms with key pairs that does not have the symmetric property of RSA may still be used for signing, providing that

[2]In addition, the number of groups may grow linearly with the number of users, leaving symmetric key distribution as an $O(n^3)$ problem.

the result of a transformation with the private key can *only* be verified by an operation using the corresponding public key. The DSA algorithm is an example of an algorithm that is signature-only. It cannot be used for encryption.

The E, D, S and V operations are computationally expensive, so for efficiency reasons the full plain text is not subject to encryption or signing. A hashing algorithm $H(P)$ generates a *message digest* which is subsequently signed:

$$\text{Signature} = S(H(P))_{K_{priv}} \tag{2.6}$$

The receiver of the signature produces a new message digest $H(P)$ and compares it with $V(Signature)_{K_{pub}}$.

In a similar manner, a temporary key K_{temp} is used to encrypt the plain text with a symmetric (and computationally cheap) crypto algorithm. K_{temp} is then encrypted with the receiver's public key and added to the message sent:

$$C = E_{symm}(P, K_{temp}) + E_{asymm}(K_{temp}, K_{pub}) \tag{2.7}$$

2.2.1 Assurances of asymmetric cryptography

On the condition that the users possess the correct public keys of each other and the private keys are kept secret, the users can safely assume that:

- Content encrypted with the public key of a receiver (a subject) can only be read by that receiver
- Content encrypted with the public key of a receiver can not be modified without detection by human recipient (or any mechanism that can distinguish between meaningful and meaningless information).
- A signature made with the private key of a sender can only have been made by that sender

It is important to realize that an electronic signature not necessarily bears legal or contractual implications. The legal consequences of an electronic signature must be defined by a contract between the parties.

2.3 Why certificates?

Unless care is taken to ensure that a public key of a subject is safely distributed to others, a so-called "man-in-the-middle-attack" is possible. The attack happens as an attacker intercepts and modifies the communication between the sender and the receiver. During a man in the middle attack the attacker diverts the client (the one who initiates a session or sends a message) to the attacker's computers instead of the correct receiver, e.g. through false DNS information[3].

[3]DNS - Domain Name System, the service that translates Internet names to IP addresses.

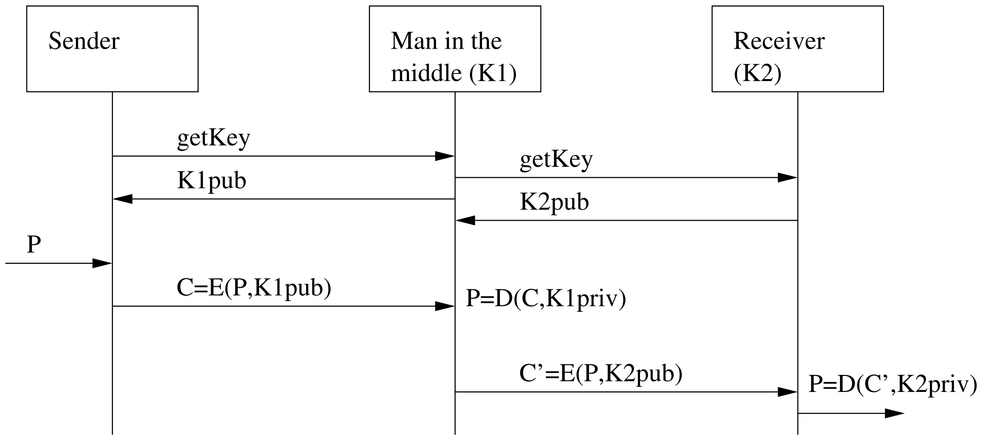


Figure 2.1 Man in the middle attack with forged keys

Figure 2.1 shows a sequence of events whereby an attacker has successfully intercepted a communication between sender and receiver. The attacker presents to the sender a forged public key, to which the attacker holds the private key himself. In the example shown an encrypted message is decrypted by the attacker before it is passed on to the receiver using the receiver's correct public key. The message is delivered as expected, so the attack may remain undetected. Not only will the content be exposed to the attacker, the attacker may even modify the message before it is passed on.

A signature operation would likewise be corrupted by this type of attack. The attacker may modify the message which would still be considered as correctly signed.

The man-in-the-middle vulnerability creates a need to assure the correctness of public keys. A public key must be associated with a *subject* and the correctness of that association must be verifiable by everyone. *Public Key Certificates* is a means for such verification.

2.4 Public Key Certificates

The most common method for public key assurance is through digital certificates and the idea of a *Trusted Third Party*. Everyone in the network is assumed to trust the validity of the public key of the *Certificate Authority* (CA). This trust can be accomplished by pre-installation of the key during e.g. software installation.

Given some kind of proof of **User1**'s identity[4], the CA will issue a public key certificate of User1's public key with the following structure:

- User1's public key $U1_{pub}$
- User1's identification (globally unique name, e-mail etc.) $U1_{id}$
- Validity period for the certificate

[4]"Proof of identity" means a *verified association* between the identity and identification of a subject.

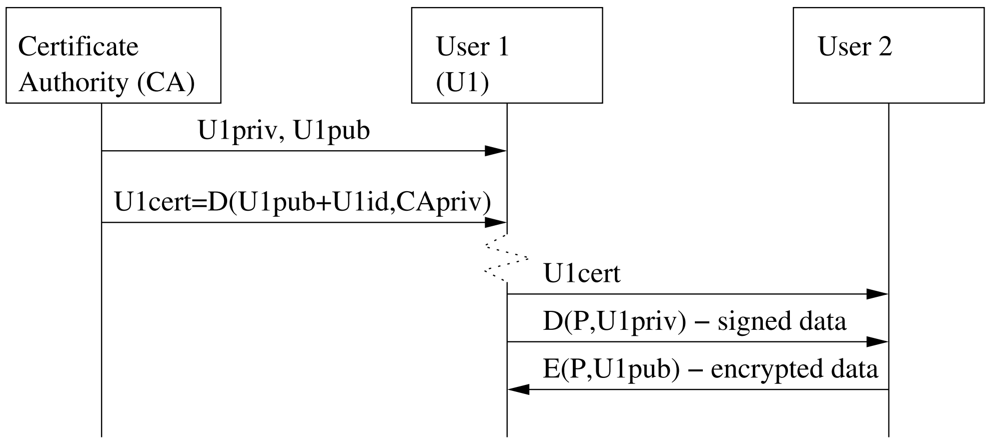

Certificate Authority (CA)
User 1 (U1)
User 2
U1priv, U1pub
U1cert=D(U1pub+U1id,CApriv)
U1cert
D(P,U1priv) – signed data
E(P,U1pub) – encrypted data

Figure 2.2 Certificate Authority issues a certificate on U1's public key

- Description of the authorized use of the key.
- CA's signature on the information above
- Technical info (CA's name, chosen crypto algorithms etc.)

The public key certificate of User1's key is denoted $U1_{cert}$ and is expressed by the following formula:

$$U1_{cert} = S(U1_{pub} + U1_{id} + \text{Validity} + \text{Authorizations})_{CA_{priv}} + \text{techinfo} \qquad (2.8)$$

The verification of User1's identity may be delegated to a *Registration Authority* (RA), which informs the CA through a protected communication channel.

$U1_{cert}$ may be distributed to anyone, it is *not* a confidential document. It can be made available in the CA's certificate repository, disseminated by User1 or by any other practical means.

The corresponding private key must be installed in User1's computer. If the key pair is generated by the CA, the private key must be safely communicated to User1 over a protected channel[5].

Alternatively, the subject may generate the key pair herself, and pass to the CA a *self-signed certificate*, which proves to the CA that the subject possesses both the private and the public key. The CA (or RA) may then verify the identity of the subject (using any method) and issue a certificate on the subject's public key. The advantage of this method is that the private key never needs to be transferred over a network link.

2.4.1 Assurances of public key certificates

Anyone that possesses User1's public key certificate may safely assume that:

- The association between the public key and the identity given in the certificate has been verified by the Certificate/Registration Authority

[5]The PKCS#12 container format can be used for this purpose.

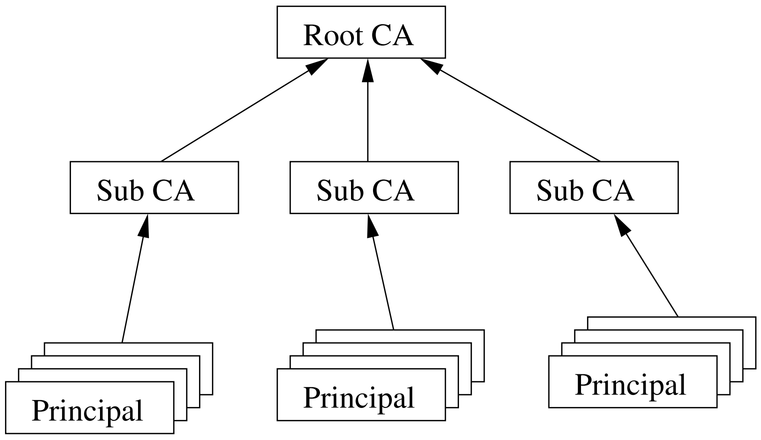


Figure 2.3 Certificate paths from a principal to root CA

- The private key has been transferred to the key owner in a way that maintains the secrecy requirements
- The identity or the public key cannot be modified undetected

on the condition that:

- The CA has a trustworthy way of establishing the identity of a subject
- The private key of the CA is kept secret

2.4.2 Trust model of public key certificate chains

A recipient of a public key certificate trusts its validity based on trust in the CA. The CA is trusted to keep its private key secret and to implement a policy to establish the identity of the subjects before a key pair is generated and a certificate issued.

This is called a *trust chain*: I trust the certificate of User1 because I trust the CA. In the future I can trust another certificate signed by User1 if I trust User1's certificate issuing policy[6]. Certificates should not normally be issued by normal users, but by Certificate Authorities. The intended use of trust chains is therefore one where one CA delegates its authority to a sub-CA "branch office". The sub-CA is equipped with its key pair and a certificate indicating the authority to issue certificates (called a CA certificate). The sub-CA could be authorized to issue certificates with only a limited validity period, certificates with limited use (e.g. no CA certificates), or only to subjects with identities within a given name space.

Figure 2.3 shows how principals (subjects) are represented by certificates issued by a sub-CA. The sub-CA are represented by certificates issued by the "parent" CA. The certificates contain the identity of the issuer (indicated by the arrow), thus forming a *certificate path* from any certificate to the

[6]Anyone can issue a certificate signed with their own private key, but it is at the receiver's discretion to trust it.

the root CA. The certificate of the root is self-signed (using the root CA's own key) and becomes the end of the path.

The root CA is called the *trust anchor* and is (by definition) trusted by everyone in the domain. In order to *validate* a certificate, each certificate on the path to the trust anchor must be inspected and approved. These certificates may be sent to the validating party together with the leaf certificate, the validating party can request them from a repository, or they may reside in a local cache.

2.4.3 Certificate revocation

Although certificates have an expiration date, circumstances may require that they are revoked before they expire (e.g. the person represented by the certificate is being reassigned). Below is mentioned four distribution methods for revocation information:

1. **Certification Revocation List (CRL):** The CA maintains a list of references to revoked, non-expired certificates. A validating party can inspect the CRL during certificate validation. The CRLs must be disseminated to every party which intends to validate certificates. The distribution scheme used can be a "pull"-based scheme, whereby the validating party retrieves the CRL when needed (e.g. when the previous list expires), or a "push"-based scheme where the CA initiates the transfer. Both approaches are resource-consuming processes. The high demand for resources created by a CRL service is raising concern, and the entire concept of CRL is somewhat controversial.

2. **Delta CRLs:** Rather than to transfer the full CRL each time, one can send only the additions to the last full CRL (called *base CRLs*) in order to reduce the list volume. There can be several delta CRL issues between base CRLs. They are distributed with either a "pull"- or "push"-based scheme. Delta CRLs reduce the traffic volume considerably, but creates some latency concerns. Delta CRLs are not well supported in COTS software.

3. **Partitioned CRLs:** The total CRL volume can be split into several lists, each representing a specified range of certificates. A revoked certificate is being listed on "its" CRL partition. Each certificate is annotated with the *distribution point* (a URL) of the corresponding CRL partition. A client validating a certificate can download this CRL partition if not already in cache. Partitioned CRLs may reduce the total volume of CRLs transferred over the network, but require a "pull"-model for distribution where clients always initiate the CRL transfer.

4. **Online Status Checking:** All revoked certificates are registered with a server, and a validating party may request the revocation status of a certificate using a client/server protocol. Online checking relieves the system from distribution of CRLs, but requires constant availability of network connections between the status provider and the clients. The response from a status provider will have a reasonable validity lifetime and would therefore be cacheable (although no COTS software have been observed to do this).

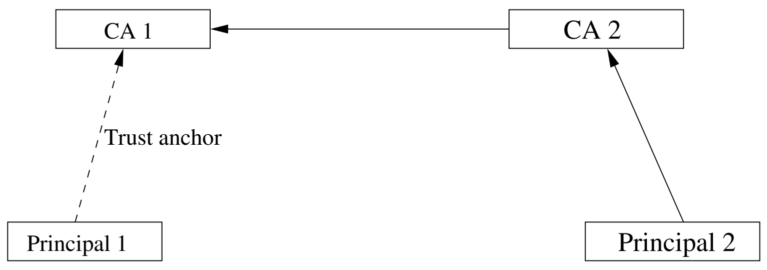


Figure 2.4 Certificate paths between CAs (cross domain)

2.4.4 Cross domain certificates

When *Principal 2*, belonging to CA domain 2, wishes to communicate with *Principal 1* which belongs to CA domain 1, it present its certificate to *Principal 1* (see Figure 2.4) in order to attest its public key.

Principal 1 cannot validate this certificate since it does not approve *CA2* as its trust anchor. The solution is to create a *Cross Domain Certificate*, issued by *CA1* to attest the public key of *CA2*. The Cross Domain Certificate creates a validation path from *Principal 2*'s certificate to *CA1*, which is the trust anchor of *Principal 1.*

CA2's certificate is (by definition) self-signed, so there is no reference to the cross domain certificate issued by *CA1*. Validation involving cross domain certificates must therefore use other means to locate them. This process is called *certificate path discovery*, which is an active field of research [10].

Although cross domain validation appears to be a matter of a small number of inter-CA certificates, it seriously aggravates the resource consumption of revocation handling. In order for *Subject 1* to validate a certificate issued by *CA2*, it not only needs the cross domain certificate issued by *CA1*, but also needs access to *CA1*'s revocation information. This access can either be as an online service or a CRL distribution service. If *CA1* represents a large domain (e.g. US DoD), its CRL distribution can be overwhelming for a small domain.

2.4.5 Key management of asymmetric cryptography

In a large community of subjects/users, a facility is required for distribution and management of public key certificates and certificate revocation lists. The tasks of key management includes:

- Generate key pairs and certificates
- Renew certificates of existing public keys
- Revoke certificates on invalidated keys
- Serve client requests for stored certificates, revocation status and revocation lists

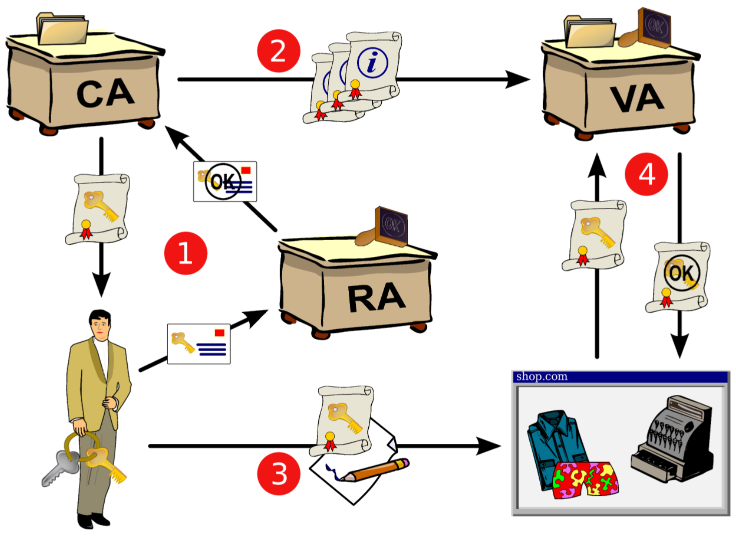


Figure 2.5 Main components of a PKI system (Source: Wikipedia Commons)

A set of services which issues, stores and disseminates certificates, status information and CRLs is called a *Public Key Infrastructure* (PKI).

In order to disseminate certificates (and revocation lists) among a large population of clients scalable distribution and storage services are required. These services should respond to interactive requests for certificate retrieval, certificate validation and CRL requests. It may also employ message-oriented middleware to disseminate data efficiently. The service must offer low latency and high availability and is therefore likely to employ replicated storage and distributed servers.

2.5 Important PKI components and operations

Figure 2.5 indicates the main components of a PKI and the flow of information between them. The figure will form the basis for the explanation to follow. The main components shown on the figure are:

Certificate Authority (CA) The CA issues public key certificates. It may also generate key pairs on behalf of subjects. The signature of the CA is trusted by everyone in the domain, making it a *trust anchor*.

Registration Authority (RA) The RA plays a part during certificate generation by verifying the correct association between the identity of the subject and the identification of the certificate. I.e., that it really is the person “John Doe” who asks for a certificate with that identifier name. The RA may employ any procedure for this purpose, depending on the security policy in force. Identity verification may involve manual procedures like checking id cards.

Validation Authority (VA) The VA is delegated (from the CA) the responsibility to decide whether a certificate is valid. Upon requests from clients, asking ”Is this certificate valid?”, the RA will respond ”Yes”,”No”, ”Don’t know”,”Yes, but only for encryption purposes” etc. Most often, the RA is a rather simple service which only responds with the *revocation status* of the certificate (which is semantically simpler than an actual validation). A response from a VA will be signed to protect its integrity, which means that the VA's certificate becomes a trust anchor (there is no one else to ask for revocation status than the VA, so how could we possibly check the revocation status for the VA's certificate?).

Please keep in mind that these are *functional* component, not architectural. They can be co-located, distributed or even abandoned.

The operations of a PKI is partly associated with individual business operations (certificate validation), and partly part of a preparation stage before business operations can take place (certificate issuance). The illustration on Figure 2.5 numbers the main operations which are described below:

1. *Certificate Issue*. The figure indicates that the client generates his own key pair and passes his/her public key (together with identity information) on to the RA for verification. The standard format for this message is called PKCS#10. The RA employs the chosen procedure for verification and (if passed) forwards the attested information to the CA. For this forwarding there is no obvious standard protocol. Existing standards are not seen widely used, proprietary protocols have been observed, and e.g. a simple authenticated HTTP session (with SSL) will also suffice. The final response from the CA is a valid certificate sent to the client. The widely used format for this message is the PKCS#7 format, which is likely to be recognized by the client's software and loaded into its *certificate store*.

   Variants of this operation includes CA key generation. Software installed in the CA is less likely to be compromised (e.g. through trojans) so key pairs generated in the CA will be ”well computed” (avoiding known weaknesses in the choice of numbers). CA generated keys allows for *key escrow*, which means that lost private keys can be restored. On the other hand, CA generated keys requires that the private key is transferred from the CA to the client, exposing the key for eavesdropping on its way. The message format PKCS#12 is frequently used for sending certificates and private keys together under cryptographic protection. Client programs which require key pairs to be installed (e.g. Firefox or Thunderbird) recognize the PKCS#12 format.

2. *Certificate Revocation*. Certificates are issued with an expiration date. If it is necessary to revoke the certificate (in order to invalidate the key pair), some message mechanism must exist to inform the client (which validates the certificate) about this. On Figure 2.5 the Validation Authority (VA) receive *Certificate Revocation Lists* (CRL) from the CA which contain references to revoked certificates (that have not yet expired). The VA receives client requests regarding the revocation status for certificates, and generates responses based on the CRL information.

   Other mechanisms exists, like allowing the VA online access to CA's certificate database, or sending CRLs directly to clients (which eliminates the need for a VA) based on either “pull”- or “push”-based operations.

3. *Signature Generation*. Digital signatures serve as proof for both authenticity (that the message originates from the holder of the private key) and integrity (that the message has not been modified since the signature was applied). In order to verify the identity of the signer, an attested binding between the public key and the identity of the owner of the corresponding private key must be available to the validating party. This binding is offered by the certificate, which must be made available to the receiver (validator) of the message. The obvious (but far from optimal) solution is to include the certificate as part of the signature, a solution which is used in many COTS products. Other approaches include letting the validating party itself fetch the certificate when needed.

4. *Certificate Validation*. The receiver of the signed message must verify the correctness and validity of the signature, which involve inspection of certificates and checking their revocation status. The receiver will need to construct a *certificate path* which connects the signature to a *trust anchor* through a chain of digital signatures. The certificates on the path must be inspected for expiration, correct use, maximum path length, policy restrictions etc. For this purpose, certificates may have to be retrieved by the validator. Additionally, the revocation status of the certificates must be assessed. As indicated on Figure 2.5, the Validation Authority (VA) can offer revocation status information based on data provided by the CA. Alternatively, the client can assess the revocation status based on CRLs stored locally.

   Figure 2.5 indicates a single-domain validation. In cases where the certificate is issued from a different domain (a different CA), a *cross domain validation* is necessary, as outlined in Section 2.4.4. Cross domain validation requires more than one source for revocation information and increases the architectural complexity considerably.

Table 2.1 lists observed traffic volume (as number of bytes and packets) during a set of PKI operations conducted by a selection of COTS software products [14]. These numbers will be used in subsequent traffic calculations later in the report.

Operation / property	*Number of bytes (approx.)*	*Number of packets*
Issue two certificate pairs	25000	48
Certificate revocation	37000	88
OCSP service invocation	2800	12
Size of CRL	700 + 36 * #entries	
Size of certificate	1200	
Size of signature structure – Adobe Acrobat	12000 alternatively 26000	
Size of signature structure – S/MIME	4400	
Size of signature structure – MS Word	3600	
Size of signature structure – Entrust Entelligence	2000	

Table 2.1 Summary tables for file size increments and network traffic during signature operations

2.6 Summary

The purpose of this section has been to describe the principles of public key cryptography, and which operations which must take place during the exchange of authenticated and secret information. The protocols and services necessary to conduct these operations have been described and put in context of the relevant use cases. Issues of complexity, trust and volume have also been addressed.

The focus of this report is on scalability problems and optimization of validation operations, since these are identified to be the largest consumer of system resources. As a principle, a validating agent need *certificates and revocation status* for the entire certificate path in order to verify a signature.

The report will now commence with discussions on how the different services can be optimized. First, a general discussion on optimization of distributed systems will be presented, followed by analyses of operations specific to public key infrastructures.

3 General optimization techniques

In general, efforts on optimization of a distributed system will follow these guidelines:

- Identify the bottlenecks. Do not optimize if the resources are abundant. In a wireless/tactical system, the network service is likely to be the scarce resource.
- Eliminate message duplication: Employ multicast middleware, transfer information only when needed or demanded.
- Reduce the size of messages. Employ compression, eliminate non-essential and redundant information elements.
- Reduce frequency of messages. Employ piggybacking, avoid polling, transfer only when required.
- Reduce transport path length. Employ cooperative caching to find the closest copy of a required data element. Mobile agents may move the processing activities closer to the data store.

In particular for systems which deals with information security, there is a *cost-security tradeoff* which must be considered during an optimization activity: A detection of a problem or a risk (like a revoked certificate) should be acted upon as soon as possible, so message *latency* translates into *risk*. A similar tradeoff is found where non-essential security features are left out for cost reasons, like DOS (denial of service) protection and clock integrity protection.

The following sections provides a brief and general discussion on optimization techniques which could be used in a PKI.

3.1 Active replication

The term “replication” refers to the process of making functionally identical copies of a data store. “Active replication” refers to a process where the data is copied in a pro-active manner, unrelated to the individual client operations on the data store (but possibly influenced by the statistics of past operations). Active replication has several effects:

- Service redundancy in combination with a fail-over mechanism improves the availability of the service. Redundant services can also offer load balancing and traffic peak mitigation.
- Service endpoints closer to clients reduce the round-trip delay and may reduce total network traffic.

Traffic volume calculations are straightforward when the client request volume, latency requirements and the rate of change for the data storage are known. The latency requirement is the only “negotiable” variable in the formula, so if the intention of replication is a net reduction of network traffic, the allowed latency may have to rise in order to reduce the frequency of replica updates.

The simplest configuration for active replication is to declare one copy as the “master”, and the other as “slaves”. The master contains all data, possibly including transaction history, logs, offline backup etc. The slaves contains selections or projections of the data reflecting the requests from clients. If update operations on slave copies are allowed, a resolve mechanism for potential update conflicts must be in place.

Replication schemes raise concern over *update consistency*. Updates must take place so that the replica never exposes inconsistent data to the clients. Active replication (as opposed to passive replication) has the advantage of being able to employ locking schemes to ensure transactional semantics during replication.

Another consistency aspect is the *temporal* one. Different replica may at one time instance hold different generations of a data item, i.e. some with recent updates included, some without. A client which connects to different replica over time (e.g. under a load balancing arrangement) may be exposed to generations in an unchronological order. This is not welcome, in particular if it is the client’s own updates that “disappears”.

3.2 Passive replication

“Passive replication” refers to replication where the copy operations between replica takes place *on demand*. When a client requests an operation on the data item which is not present in the replicum, the replicum requests that data item from somewhere else, possibly the “master” store. The client request is then processed, and subsequent requests for this data item is processed on the local copy. Passive replication is also known under the term “caching”.

Passive replication requires little configuration and maintenance, which is probably why it is so popular. It also adapts well to changes in client demands and network topology. For a collection of static web pages or other similar content, a passive replication scheme works very well. For dynamic web content (which is dominating the web at present) it is less efficient, since much of the content will be non-cacheable. For mutable[7] data items, passive replication suffers from the lack of a consistency model, since the replica are only loosely connected - a replicum will not distinguish between a request from a client and a request from another replicum. The path by which an update should propagate to other replica is difficult to determine in a passive replication scheme.

A replicum which holds a data item may choose to check other stores for updates for every client request, or after some "expiration time". For data items which are rarely updated, often requested, but has low latency requirements (updates are considered as urgent), the passive replication scheme works less efficient than active replication schemes since the replica must spend network resources on useless "polling".

The HTTP protocol lends itself well to passive replication with its header elements "If-Modified-Since" (request) and "Last-Modified" (response), and the result code "303 - Not Modified".

3.3 Information dissemination

There are many situations where a central resource needs to disseminate the same set of information to a large audience of clients: Software updates, revocation lists, virus signatures etc.

3.3.1 Push vs. pull

The clients may "pull" these data when needed, through scheduled requests or based on expiration time of existing data items. This approach is the simplest one from a configuration perspective, but is clearly inefficient. The server (and the network) must transfer the same content for every client, even clients that are sitting next to each other at the other side of the planet.

A "push" based scheme can employ multicast algorithms which may use *multicast trees* and exploit *broadcast media*. A push based scheme, where the server initiates the transmission, can schedule transmissions when data data store is updated. In comparison, pull-based transmission takes place when initiated by client demand.

Push-based distribution is clearly more efficient in terms of communication cost, since a push-based scheme transmits one copy of the data per network link, whereas a pull-based scheme transmits one copy per client.

3.3.2 Client dynamics

When a client node starts its operation for the first time, it may need the data set which has been disseminated most recently. Using a pure push scheme, there is no other option for the client than to wait for the next issue. Regard must therefore be paid to this situation, whether it calls for a separate configuration step on the new client node, or a simple pull based service for these situations.

[7]Mutable means "may be updated"

3.3.3 Reliable delivery

Normally, the information dissemination operation require a reliable delivery to all receivers. Infrastructure for this purpose, called *Message Oriented Middleware* (MOM), forms a store-and-forward overlay network of servers. MOM is well understood technology and is found in mature implementations. A large organization is likely to have a messaging system already which could undertake the dissemination task with little extra effort.

A dissemination job may not require reliable delivery, however. If data items are issued often and the cost (or risk) involved by a missed transmission are low (e.g. a weather forecast is probably unimportant for a node which is not in operation), then the dissemination could use non-reliable mechanisms instead. The use of UDP multicast is an example on cheap but unreliable dissemination.

Consequently, the problem of information dissemination is associated with *update latency*, *client dynamics*, *reliable delivery* and of course *transmission cost*.

3.4 Network path reduction

The resource consumption of network communication is proportional to the number of links used in the network path between sender and receiver. Some of the replication techniques described in the previous sections will contribute to a shorter network path as a *side effect*. Measures can be taken also to shorten the path as its primary effect. In this section, we will analyze *mobile agents* as a path reduction technology.

The term “Mobile agents” refers to processes (context, data and code) represented in a form which can traverse the network and execute in different places. It is normally assumed to be “accommodated” in a host and allowed to operate on that host through a programming interface (API). The agent cannot do anything across that API that could not be done from a program on a different place in the network, in principle. But a mobile agent can operate with higher speed (bus speeds) and even if the host is temporarily disconnected from the network.

It is based on this perspective that mobile agents are categorized under “path reduction” techniques, because it does the same as a remote client, only over a shorter network path. Contrary to replication, where the data is moved closer to the client, the client is now moved closer to the data through a mobile agent which operates on the client’s behalf.

The use of mobile agents has not gained widespread use for a number of reasons: Lack of a widely accepted platform, API and architecture, crash recovery problems, security problems and the absence of a “killer application”.

3.5 Message size reduction

A rather obvious optimization strategy is to reduce the size of messages. Repeated information, derived information and previously transmitted information may be excluded.

Information elements are found to be included for reasons of convenience, e.g. to ensure that the receiver is in the correct state or has the necessary information. The sender's public key certificate is an example of such an information element: the receiver probably has got it already, but the easiest thing to do by the sender is to include it anyway.

A solution to this problem is *not* for the sender to assume the receiver's state (whether it has received the certificate earlier). The receiver may have been rebooted or changed its storage medium in between. A better approach is to include only essential information in a message and let the receiver decide if more information is needed. In case the receiver needs more information it takes extra round-trips to retrieve this, which is considerably more costly than to include it in the original message. The trade off should be calculated based on the probability for this to happen.

Message sizes may also be reduced through the exclusion of derived or repeated information, and by ordinary compression techniques. Messages coded in XML syntax lend themselves well to specific compression since they contain much repeated information [9].

3.6 Service/component co-location

Clients need different information taken from different servers: Configuration data, revocation lists, service discovery, status information etc. The information elements may be used in different software components and will be retrieved independently through different network connections.

The number of protocol round-trips involved in these operations may be reduced if the information servers are co-located and several retrieval operations are combined into one. It is clearly an optimization potential, but in order to exploit that potential the software component involved would have to be re-written to accommodate new retrieval protocols and new message formats.

This approach could be called "cross layer", since different protocol levels could cooperate on the use of a common information server. The services used to configure the IP protocol (DHCP) could easily be used to configure the message system, service discovery system, clock configuration etc.

In the PKI, this approach contradicts the former section somewhat. A sender of a signed message could co-locate the message system with the certificate store and the status provider by adding not only the certificate, but also a signed response from a recent request to a status provider (or even the CRL). The use of "piggybacking" to reduce the number of protocol round trips requires careful analysis of the net effect, since the bundled information elements may be wasting bandwidth when they are not needed by the receiver.

4 Scalability analysis of validation scenarios

The scalability problems of PKI operation is first and foremost connected to the validation operations, as the preceding discussions have shown. It is therefore the purpose of this section to provide a scalability analysis of a series of certificate validation scenarios. The section has been independently published in MILCOM 2010 [8].

Although the scalability perspectives of public key infrastructure (PKI) operation has been addressed in many papers, there are few contributions to formal scalability analysis. This section establishes a framework based on stochastic process modeling and a scale free distribution of message traffic.

Scalability of PKI based systems is a matter of key distribution, added message size due to signature and certificates and, in particular, cost of certificate validation. Certificate validation usually implies a check for the revocation status of the certificate. Revocation status can be obtained by an online status provider, or through the distribution of revocation lists. Both these alternatives may be subject to a number of different optimization techniques, some of which are explored in this section.

Three simple but essential scenarios for certificate validation has been chosen for the analysis. They represent only slightly optimized protocols, but are still more effective than what is found in COTS software. Studies of validation mechanisms in popular software (Adobe Acrobat, Firefox, Thunderbird, Java standard library etc.) reveals a simplistic and unoptimized approach to validation without even the most basic caching mechanisms.

The chosen scenarios assume that the certificates are issued directly by the Certificate Authority (CA), so no multi-step certificate path need to be validated. Although this assumption is slightly unrealistic, the presented analysis lends itself well to extension.

The three scenarios about to be presented are: (1) Invocation of an online status provider service, with a local cache for storing recent validation results. (2) Distribution of revocation lists, based on multicast and the use of *delta revocation lists*. (3) Certificates without a revocation mechanism, but with short lifetime.

4.1 Online certificate validation

The first scenario for certificate validation is shown in Figure 4.1. During a transaction between two nodes, the *sender* passes its public key certificate together with a signed message (1). In order for the *receiver* to accept the signature, it must ensure that the certificate is still valid. The receiver inspects a local cache of recently validated certificates (2,3) and accepts the certificate if it has been validated in the recent past. Otherwise, it contacts a status provider service in order to obtain the validity status of the certificate (4,5).

This section will now analyze this scenario from a scalability perspective. First, it is necessary to establish a probability distribution model for certificate exchange.

4.1.1 Probability distribution of certificates

Scale free distributions as proposed by Barabasi et al. [3] have often been applied to different social networks. In a scale free distribution there exists an inverse proportion between the number of links from a node (the degree of the node) and the number (the frequency) of nodes with this particular

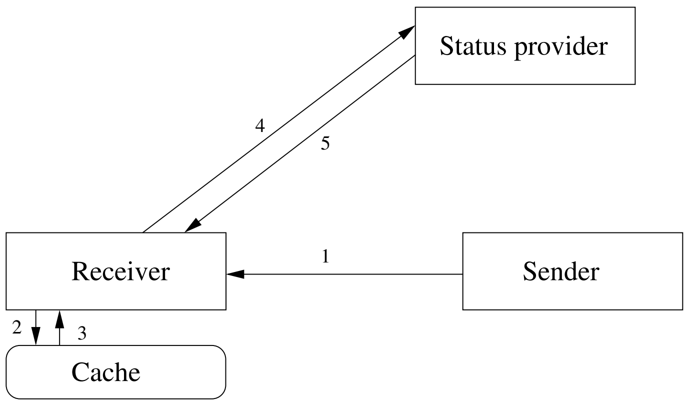


Figure 4.1 Basic certificate validation

degree.

Barabasi shows how telephone traffic, sexual habits, airline flights and genetic effects all have the properties of a scale-free distribution.

Given the indications of the scale free nature of social interactions, it is a sound assumption that the exchange of certificates (or public keys or sender's identifications) follows the same pattern; the most frequent certificate occurs twice as often as the second most frequent etc. Certificates can thus be *ranked* by the frequency of which it occurs in received messages.

Let r denote the rank of certificate C_r. The relative frequency f of this certificate is:

$$f(C_r) = \frac{a}{r} \tag{4.1}$$

where the value of a is determined so that

$$\sum_{r=1}^{N} \frac{a}{r} = 1 \tag{4.2}$$

The value N is the size of the certificate population (number of distinct certificates).

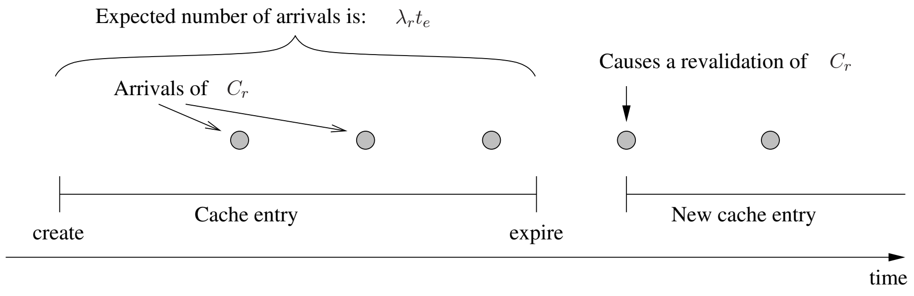
4.2 Cache hit/miss analysis

The receiver (Figure 4.1) receives b number of certificates, according to the scale-free distribution just mentioned. For each of the received certificates, the probability that this certificate is C_r is a/r. The probability that certificate C_r is *not* among these b number of certificates is

$$(1 - \frac{a}{r})^b \tag{4.3}$$

The probability that the bth received certificate is C_r and that C_r has not been received earlier in $b - 1$ arrivals is

$$(1 - \frac{a}{r})^{b-1}\frac{a}{r} \tag{4.4}$$



Expected number of arrivals is: $\lambda_r t_e$
Causes a revalidation of C_r
Arrivals of C_r
Cache entry
New cache entry
create
expire
time

Figure 4.2 Cache expiration and subsequent re-validation

For any received certificate, the probability that the bth received certificates is previously unseen and consequently has not been received in the series of $b - 1$ number of previously received certificates (cache miss) is

$$P_1(b) = \sum_{r=1}^{N} (1 - \frac{a}{r})^{b-1} \cdot \frac{a}{r} \tag{4.5}$$

Equation 4.5 is a continually decreasing function, which confirms the intuitive behavior of a caching function where cache entries never expire. As more certificates are known, new and unseen certificates become rarer.

4.2.1 Cache expiration

As was mentioned in the scenario presentation, the cache of validated certificates will contain only *recent* entries, which means that they will expire and require a re-validation in order to enter them into the cache again. Therefore, the "cache hit/miss" analysis in Section 4.2 must be augmented with an analysis of expiration mechanisms.

It is reasonable to assume that all certificates in the cache have the same expiration time, denoted t_e.

The *arrival rate* of messages now becomes interesting. We will denote the total arrival rate λ_{tot}. We assume a Poisson distribution for the arrival events.

We still assume the scale free distribution of certificates. Accordingly, the arrival rate of C_r, denoted λ_r, is:

$$\lambda_r = \frac{a}{r} \lambda_{tot} \tag{4.6}$$

The expected number of C_r during a time interval t is

$$E(C_r, t) = \lambda_r \cdot t \tag{4.7}$$

$$= \frac{a}{r} \lambda_{tot} \cdot t \tag{4.8}$$

During the cache expiration period t_e the expected number of arrived instances of C_r is $\lambda_r t_e$. The next arrival of C_r will cause a re-validation. Consequently, $\lambda_r t_e + 1$ instances of C_r are expected to be received between re-validations. This sequence of events is illustrated in Figure 4.2. The probability for any certificate C_r to cause a re-validation is

$$P(revalidation|C_r) = \frac{1}{(a/r)\lambda_{tot}t_e + 1} \tag{4.9}$$

Given the scale free distribution of certificates, the overall probability that any received certificate will cause a re-validation is found as the sum over the probability distribution of certificates:

$$P(revalidation) = \sum_{r=1}^{N} \frac{a/r}{(a/r)\lambda_{tot}t_e + 1} \tag{4.10}$$

This probability estimates the fraction of the message traffic which will trigger a re-validation and can therefore be used in the scalability analysis for a validation service.

The expression only applies to the received certificates which are stored in the cache of the receiver. The next section will present a combined expression of cache miss/hit and cache expiration.

4.2.2 Cache hit/miss and expiration combined

Equation 4.5 presented an expression of the fraction of the certificates which misses the cache and require a certificate validation. For the rest of the certificates, the analysis in Section 4.2.1 may be applied to find the fraction of messages which hit an expired cache entry. These messages will also require a certificate validation through an invocation of status provider services.

Equation 4.5 gives the first fraction, and the complementary value $(1 - P_1)$ is multiplied with the expression in Equation 4.10 in order to find the second fraction. The combined expression P_2 is

$$P_1(b) = \sum_{x=1}^{N} (1 - \frac{a}{x})^{b-1} \cdot \frac{a}{x} \tag{4.11}$$

$$P_2(b) = P_1(b) + (1 - P_1(b)) \sum_{r=1}^{N} \frac{a/r}{(a/r)\lambda_{tot}t_e + 1} \tag{4.12}$$

Figure 4.3 shows a plot of the probability for a certificate validation operation during a series of received certificates. The parameters are: $N = 50000$, $\lambda_{tot} = 400/\text{day}$, $t_e = 4\text{hours}$. b ranges from 1 to 1000 along the horizontal axis.

During the course of operation (the value of b is increasing), the P_1 element becomes so small that the value from Equation 4.10 becomes dominant. With large values of b the "asymptote" of the plot approximates the values given by Equation 4.10. The plot in the figure clearly shows the asymptotic property of the function.

The analysis assume that both negative and positive validation results expire from the cache, which is not necessary; negative validations need not expire from the cache, in which case the probability expressed in Equation 4.12 may be too high, depending on the revocation rate.

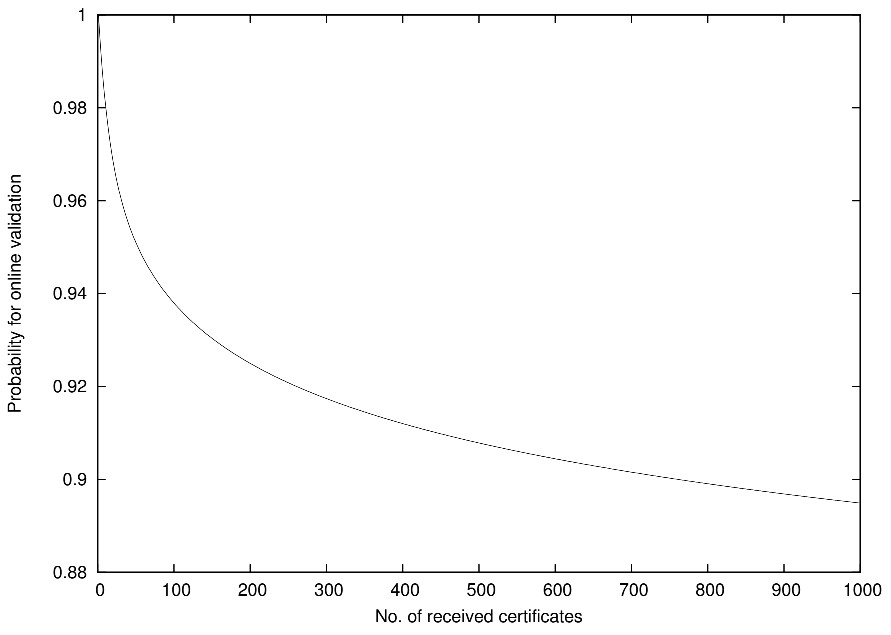


Figure 4.3 Plot showing how often a received certificate must be validated. The vertical axis shows the probability for a required certificate validation as expressed by Equation 4.12. Horizontal axis shows total number of certificates received.

4.2.3 Data rates

The analysis in Section 4.2.2 has shown how often a receiver must consult an online service during certificate validation when an on-line validation model is used. The i/o activity of the receiver is now readily estimated: The arrival rate of certificates, λ_{tot}, and the size of a request/response pair s_v for validation must be known. The receiver (client) side network traffic rate R_{online} is:

$$R_{online} = \lambda_{tot} P_2(b) s_v \tag{4.13}$$

In addition to this traffic, the increased size of messages also contribute to the load. The value of this contribution is $\lambda_{tot} \cdot s_r$, where s_r denotes the increased size of messages due to the appended certificate and signature. Since this load is present in every model presented, it is left out of the comparative perspective in this analysis.

4.3 Validation based on revocation lists

A different approach to certificate validation is to inspect a Certificate Revocation List (CRL) to see if the certificate is blacklisted. A certificate is represented on a CRL if it is invalidated before its expiration date[8]. Invalidation may be necessary if the associated private key has been compromised or if the owner of the certificate no longer belongs to the certificate domain (has quit or been reassigned). The CRL is distributed to every client at regular intervals[9].

[8]Expiration date for the certificate, not the cache entry

[9]From a security perspective, the distribution interval of CRLs is analogous to the cache expiration t_e in Section 4.2.1

A revoked certificate will be on the CRL until the certificate expires.[10] The size of the CRL therefore depends on the expiration period, the size of the certificate population and the rate of revocations. Since CRLs may grow large and are widely distributed, they raise much concern over their network resource consumption.

It is possible to make *delta CRLs* as a means for not having to send information about the same certificates over and over again. The delta CRL offers only additions to a *base CRL* sent sometimes in the past. Receivers of a delta CRL must have a copy of the most recent base CRL (sent earlier). The delta CRLs will grow bigger at each issue (since they accumulate changes to the base CRL) until they are "reset" by a new base CRL issue.

The trade off of delta CRLs is that they consume less network resources, but new clients that join must have a separate copy of the most recent delta CRL sent to them, in addition to the most recent base CRL. A population of highly dynamic clients (who join and leave often) will thus reduce the effect of delta CRLs.

Please observe that many COTS programs that checks certificate revocation status through a CRL mechanism do *not* employ delta CRLs. They download and cache base CRLs only. For this reason, this paper also present traffic analysis of a CRL mechanism without the use of delta CRLs.

4.3.1 Size of base CRLs

The CRL contains some header information and a *CRL entry* for each certificate on the list. CRL entries will have varying sizes, but 35-40 bytes per entry is a realistic figure. In the following analysis, the symbol s_C represents the size of a CRL entry. The size of the CRL header is disregarded.

According to NIST statistics [4], 10 percent of the certificates need to be revoked during a year, which means that a certificate has 0.1 probability (termed p_0) for being revoked during this period (denoted t_0). The probability for *not* being revoked during the period $n \cdot t_0$ is $(1 - p_0)^n$. For any time period t, the revocation probability is a function $p_{rev}(t/t_0)$, which can be expressed as follows:

$$p_{rev}(1) = p_0 \tag{4.14}$$

$$p_{rev}(k) = 1 - (1 - p_0)^k \tag{4.15}$$

where

$$k = \frac{t}{t_0} \tag{4.16}$$

Assuming that the certificates are created at a uniform rate, and they all have a validity period of t_c, the certificate population will at any time instance have a uniform age distribution, ranging from 0 to t_c. The fraction of the population which is revoked (expectation value) is called $r_{rev}(k)$ and can be expressed as an integral over the probability distribution function $p_{rev}(k)$:

$$r_{rev}(k) = \frac{1}{k}\int_0^k p_{rev}(t)dt = 1 - \frac{[(1-p_0)^t]_{t=0}^k}{k\ln(1-p_0)} \tag{4.17}$$

[10] A typical expiration period for a certificate is one year.

$$r_{rev}(k) = 1 - \frac{(1-p_0)^k - 1}{k \ln(1-p_0)} \tag{4.18}$$

For the values $t_c = t_0 = 1$ (year) and $p_0 = 0.1$, Equation 4.18 gives the value $r_{rev}(t_c/t_0) = 0.0509$.

If the validity period t_c is one year (a typical figure), then the CRL will represent 5.09 percent of the total certificate population N. The symbol B represents the base CRL and s_B represents the size of B.

$$s_B = r_{rev}(t_c/t_0) \cdot N \cdot s_C \tag{4.19}$$

4.3.2 Size of delta CRLs

The use of delta CRLs reduces the network traffic by reducing the frequency of base CRL distribution. Between the issues of complete CRLs a set of delta information is distributed, each of them contains updates to the last complete CRLs (not to the previous delta CRL). The size of delta CRL issues will consequently grow according to the rate of revocations.

Let the issuing interval of base CRLs and delta CRLs be termed t_b and t_d, respectively. The first delta CRL distribution (after a base CRL distribution) contains revocations during the last interval t_d. Let s_{D1} represent the size of the first delta CRL.

$$s_{D1} = r_{rev}(t_d/t_0) \cdot N \cdot s_C \tag{4.20}$$

The nth delta distribution will consequently have the expected value $s_{Dn} = r_{rev}(n \cdot t_d/t_0) \cdot N \cdot s_c$. Since there are room for $t_b/t_d - 1$ successive delta distribution, the expected average size of the delta distributions will be

$$s_D = \frac{N s_c}{m} \sum_{n=1}^{m} r_{rev}(n \cdot t_d/t_0) \tag{4.21}$$

where

$$m = \frac{t_b}{t_d} - 1 \tag{4.22}$$

The reduction in the total CRL size when delta CRLs are in use (compared to base CRL distribution only) can be expressed using the Equations 4.19, 4.21 and 4.22.

$$\frac{m s_D + s_B}{s_B(m+1)} \tag{4.23}$$

4.3.3 CRL push distribution

The CRLs can be disseminated to users based on a *push* model, i.e. a model where the CRL issuer (the CA) initiates the data transfer. The client nodes cannot request data transfer. This model can exploit a multicast service, and responds well to emergency situations where critical revocation data must be distributed immediately.

The volume of data being transferred in this model is easy to calculate, since the CRLs are sent at regular intervals. It is assumed that the delta CRL distribution is skipped when the new base CRL is distributed (every t_b time units).

The client side data rate R_{push} is

$$R_{push} = \frac{s_D}{t_d} + \frac{s_B - s_D}{t_b} \tag{4.24}$$

4.3.4 CRL pull distribution

A *pull* distribution model relies on clients that regularly polls the CAs CRL distribution service for new issues of CRLs. A CRL is annotated with an expiration date, so that the client will know when to fetch a new one. This model adapts better to client dynamics, i.e. when new clients start their operation they may request the most recently issued base and delta CRLs and become operational immediately.

The pull model can also employ the principle of *partitioned CRLs*. CRLs may be partitioned and represent only a range/subset of the certificates. The clients download and cache the CRL partitions according to the certificates that are to be validated. This section does not offer an analysis of partitioned CRLs.

An analysis of the pull model requires an estimate of the *new client node arrival rate* (termed λ_{node}) which is the rate of which nodes start their operation and need to obtain fresh CRL information, and the total number of operational client nodes (termed T). The data rate for one client node, R_{pull} is the same as R_{push} in Equation 4.24. For the CRL service (server side traffic rate), the total traffic is $T \cdot R_{pull}$ (assuming only unicast distribution).

A "pure" implementation of the pull distribution model implies that every client at regular intervals downloads the CRLs (base or delta), i.e. exactly the same information to everyone. Dissemination of the same CRL to everyone is a problem well suited for *message oriented middleware*, *cooperative caching* schemes or other forms of *peer-to-peer distribution*. A pure pull scheme is clearly an inefficient approach for the network as a whole.

4.3.5 Combined push/pull distribution

Since push and pull model have distinct advantages, they could be combined: New clients request CRL at startup (pull) and maintain their CRLs through a server-initiated distribution service (push).

Figure 4.4 illustrates how data is being transferred to the client nodes under the two models: The push model with a regular distribution rate determined by the CRL lifetime and the revocation latency requirements, and the pull model determined by λ_{node}. The sum of these two data flows is:

$$R_{comb} = R_{push} + (s_B + s_D)\lambda_{node} \tag{4.25}$$

which is the server side traffic rate. The client side rates are still R_{push} in a long term perspective .

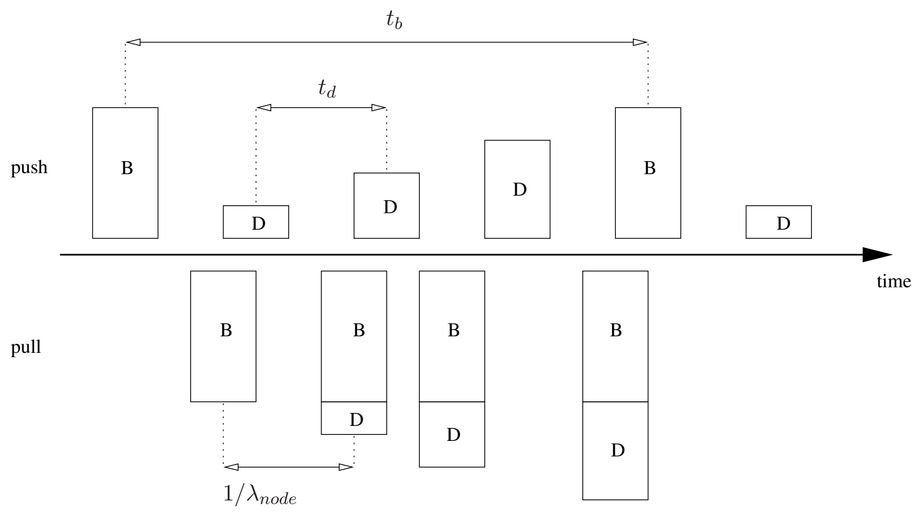


Figure 4.4 A combined push/pull CRL distribution mechanism.

4.3.6 Push/pull without the use of delta CRLs

As previously mentioned, there are several examples on COTS software which are unable to employ delta CRLs. They rely exclusively on base CRLs, and an analysis of the traffic volume created in this mode will now be presented.

A base CRL of size s_B will be delivered to a client at regular intervals determined by the revocation latency, also called t_d. Each time a new client starts (determined by λ_{node}), it will receive the base CRL. The sum of these two data flows on the server side is:

$$R_{noDelta} = \frac{s_B}{t_d} + s_B\lambda_{node} \tag{4.26}$$

The client side rates are s_B/t_d in a long term perspective since the pull operation only takes places at startup.

4.4 A calculation example

The analysis of the online validation model and the CRL distribution validation model will now be complemented with a calculation example. There is a large number of variables going into the two models, many of which are distinct for only one of the models. The variables can be tweaked in order to manipulate one model while leaving the other unmodified.

The graph in Figure 4.5 plots the client side data transfer rates as a function of the *revocation latency*, which is to be interpreted as the time period from the moment when a revocation is registered in the CA until the certificate is invalidated in every client. The revocation latency equates to t_e in the online validation model and t_d in the CRL distribution model. The variables used in the calculation were given the values shown in Table 4.1.

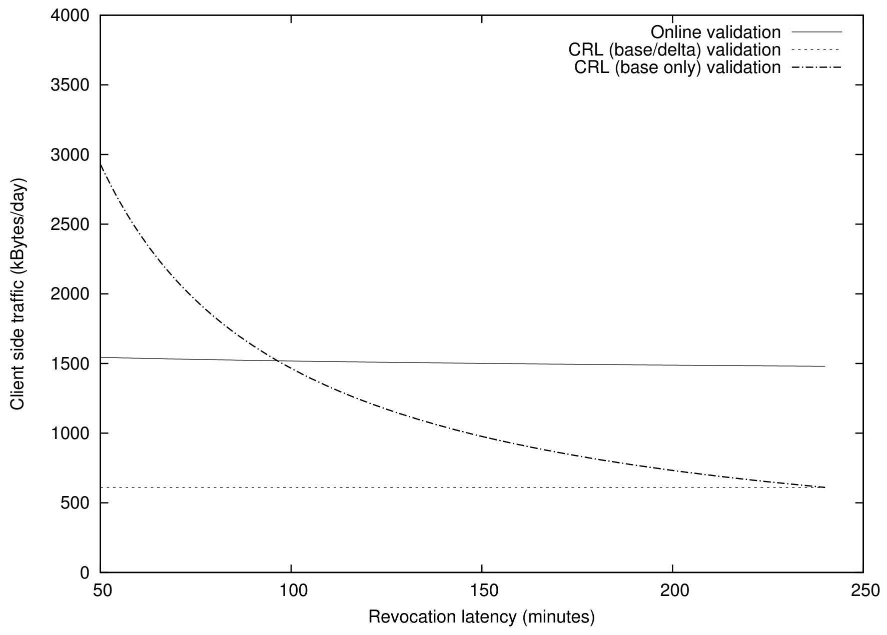


Figure 4.5 Data transfer rate for online and CRL validation under different revocation latency requirements

The plot in Figure 4.5 has one line for each of the three compared models shown in Sections 4.2.3, 4.3.5 and 4.3.6.

The analysis shows that the resource consumption by the online based validation model is proportional to the values of λ_{tot}, whereas the CRL based validation model relies much on the values of certificate revocation rate (p_0, t_0).

4.5 Short lived certificates

The use of revocation status information have generated much debate both on its principle (since "not revoked" is not an appropriate answer to "is it valid?") and over scalability concerns.

An alternative to revocation status information is to issue only short-lived certificates, and abandon any revocation mechanisms [11]. The expiration time for these certificates should be equivalent to the *revocation latency* discussed earlier in the paper (t_d, t_e). In order to maintain the same latency value, a certificate expiration time (t_c) should be set accordingly.

In the models presented earlier in this section, only the networking traffic associated with certificate validation was taken into account, not the traffic associated with the issue of certificates. With a validity period of e.g. one year, the client side data rates (which is the focus of this analysis) becomes negligible. In a model of short-lived certificates, however, this data rate becomes the main variable of a scalability analysis.

Every client who signs messages must have its own certificate. Every receiver who verifies a signa-

N	50000	Number of distinct certificates
b	200	Number of certificates received
λ_{tot}	400	Arrival rate for received certificates (per day)
s_v	4000	Size of a validation request/response (bytes)
p_0, t_0	0.1, 365	Revocation probability after 365 days
t_c	365	Expiration time for certificates (days)
s_C	40	Size of a CRL entry (bytes)
t_b	240	Base CRL issuing interval (minutes)
λ_{node}	50	Arrival rate of new client nodes (per day)
t_d	1-240	Delta CRL issuing interval (minutes on x axis)
t_e	1-240	Expiration time for cached validations (minutes on x axis)

Table 4.1 Example values used in calculation examples

ture must obtain the sender's certificate and all CA certificate up to the trust anchor. These certificates can be included in the signed message, or obtained through certificate retrieval services.

Let the symbol s_{cert} represent the certificate size. The client will receive its own certificate from the CA every t_c time units, provided that it sends at least one signed message during t_c. The resulting client side data rates R_{slc} related to this process becomes

$$R_{slc} = \frac{s_{cert}}{t_c} \tag{4.27}$$

For a certificate size (s_{cert}) of 1.2 kB and value of t_c of 4 hours the R_{slc} becomes 7.2 kB per day, which is a very small number compared to the values found in Figure 4.5.

The data rates for the CA service point will be $T \cdot R_{slc}$, where T is the number of active client nodes in the system (we assume T to be smaller than the actual certificate population). The value 400 will be chosen for T in Section 5.1.2, and the estimated CA service traffic related to certificate issue becomes 2.88 MB per day.

The certificates are also needed by the validating parties. Most COTS programs add the signer's certificate to every signature, e.g. in mail messages and PDF documents. Under these circumstances, the frequent change of certificates does not impose extra network traffic, since these programs already establish a "better safe than sorry" practice. A more efficient approach may be to send a certificate only when necessary, i.e. when the receiver does not have it already. Under this hypothetical scheme (since it has not been observed in COTS software), a frequent change of certificates would increase the volume of message traffic. The cache hit/miss analysis in Section 4.2 provides a basis for an analysis of an on-demand exchange of certificates which will be presented in Section 5.4.

4.6 Related studies

The scalability properties of a public key infrastructure are mainly related to the certificate validation process and the distribution of revocation lists. The research on these issues use a mixed set of evaluation methods: qualitative/informal, simulation experiments or formal analysis.

Adam Slagell et. al. present a thorough discussion on scalability perspectives on a PKI [12, 13], where also the security implications of different alternatives are indicated. The different service components are presented and their significance to scalability is identified. Although informative background papers, they provide no analysis or experimental findings.

André Årnes provides, on the basis of his Master's Thesis, a simulation study of the server peak rate during CRL distribution [2]. Different CRL schemes are investigated, including over-issuing, delta CRLs and segmented CRLs. The CRL distribution model is "pure pull", which means that clients obtain their CRLs directly from a central server.

David A. Cooper's work on the modeling of CRL distribution mechanisms [5, 6] is quite closely related to the analysis presented in this paper. Again, the distribution model is "pure pull", and a variety of distribution variants (ordinary CRL, over-issuing, segmented CRLs, and delta CRLs) are analyzed from a "peak rate" perspective. Client request arrive at a central server according to a Poisson distribution.

The ordinary COTS program (e.g. Firefox or Thunderbird) will fetch a CRL from an address which is either system-wide or specific to a certificate (from the *CRLdp* certificate extension), something that gives a "pure pull" distribution model some justification. On the other hand, this model is largely inefficient since the same piece of information is repeatedly handed out and sent across the cyberspace to clients which may sit next to each other. From a distribution perspective, a multicast service would do the same job much better. Multicast services are offered by a range of message transport middleware products.

We call this model a "push" model, since the server side initiates the transport, which happens asynchronously with regard to the client activities. The push model avoids the peak rate problem since the servers schedule the operations, and it allows emergency information to be timely disseminated to clients. A push model will leave the feature of over-issuing CRLs as unnecessary, and the use of segmented CRLs as impractical.

A push model may coexist with clients using the pull model through *HTTP proxies*. The CRL dissemination service may deliver asynchronously to a service point in the HTTP proxy, which will store the CRL information. The client will have their HTTP request for CRLs redirected to the proxy which will respond with the stored CRLs. A detailed study of this mechanism is provided in Section 5.1. The same may be said about the online validation method presented in Section 4.1, where an expiring cache mechanism may be implemented through an HTTP proxy, see Section 5.3 for a more detailed discussion.

a	Normalizing factor
b	Number of certificates received
B	Base CRL
D	Delta CRL
λ_{tot}	Arrival rate for received certificates
λ_{node}	Arrival rate of new client nodes
r	Rank
C_r	Certificate of rank r
N	Number of distinct certificates
p_0, t_0	Revocation probability after time t_0
R_{comb}	Client side (CS) data dates for CRL pull/push distribution model
R_{online}	CS data rate for online validation
R_{pull}	CS data rates for CRL pull distribution
R_{push}	CS data rates for CRL push distribution
R_{slc}	CS data rates for short lived certificates
s_B	Size of base CRL
s_C	Size of a CRL entry
s_D	Size of delta CRL
s_{Dx}	Size of delta CRL issue no. x
s_{cert}	Size of a certificate
s_r	Size of a received message
s_v	Size of a validation request/response
T	Total number of client nodes
t_b	Base CRL issuing interval
t_c	Expiration time for certificates
t_d	Delta CRL issuing interval
t_e	Expiration time for cached validation

Table 4.2 Table of symbols used in the analysis

4.7 Summary

This chapter has presented a scalability analysis of three different certificate validation scenarios: Online validation, CRL based validation and short-lived certificates. The analysis has focused on the study of how the client-side data rates change with different operational parameter. In particular, the *revocation latency* has been a property of interest, since this value translates into the level of risk that invalid certificates may be exploited and cause authentication forgery.

A summary explanation of the symbols used in this section is presented in Table 4.2.

5 Specific optimization techniques

In this section a number of specific solutions to PKI scalability problems will be presented and discussed. They are based on the analysis and observations which are presented in Section 4.

5.1 Push distribution of revocation list

A Certificate Revocation List (CRL) lists revoked, non-expired certificates. For non-partitioned CRLs, the same CRL is used to validate every certificate issued by the same CA. The naïve approach to CRL distribution is to let every validating client fetch its own copy (pull operation) from a central distribution point. This is an obviously inefficient method since the infrastructure between the CRL issuer and the clients must transfer multiple copies of the same information. A push based distribution scheme may avoid this problem with a messaging middleware system.

A push based scheme requires a service point in (or near) the client where CRL data can be delivered asynchronously with regard to the client main activity. Candidate technologies varies with network topology; an SMTP/Sendmail based system is appropriate for a WAN, so is a dedicated MOM (Message Oriented Middleware) system. For a LAN, a UDP-based multicast scheme would exploit the inherent broadcast nature of a shared media, although some robustness should be added to offer reliable delivery.

The client side service point for reception of push based transmissions may reside behind a firewall or NAT (Network Address Translation) device, which would require configuration of these devices for the inbound connections to succeed. Alternatively, the service point could be placed in the DMZ-branch (DeMilitarized Zone) of the local network, together with web- and mail servers.

COTS client programs, which need the CRLs in order to validate certificates, do not recognize a push based arrangement. They all require a URL through which they can fetch the CRL when needed.

As a matter of fact, the pull based distribution scheme is inherent in the CRL syntax definitions. The association between a CRL and a certificate is given in a certificate extension (CRLdp) as a service point URL, so a holder of a certificate will know from where to pull the CRL on which this certificate may be listed. There is no opposite link, i.e. no information in the CRL regarding the

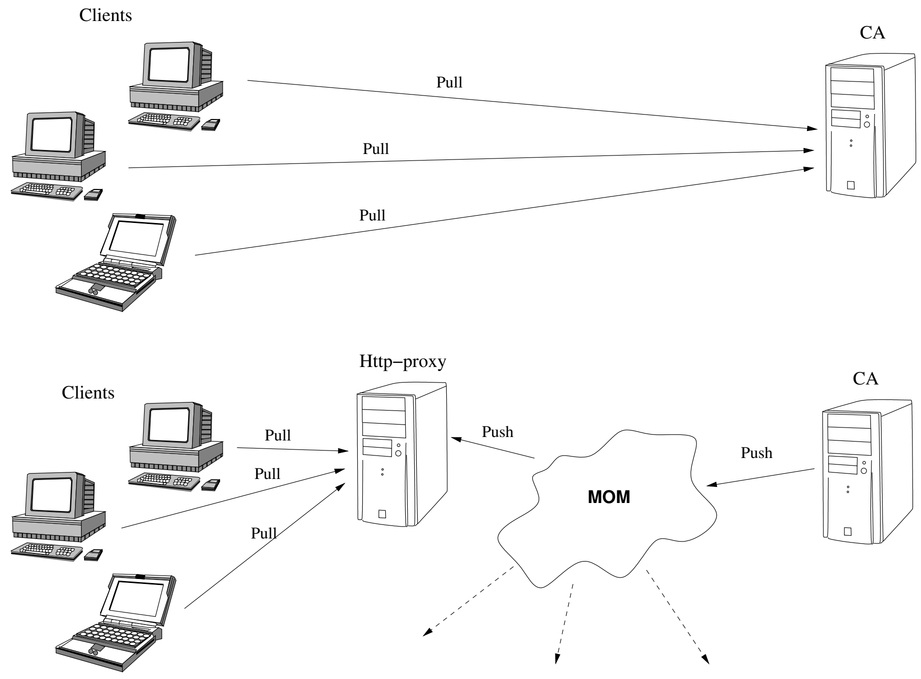


Figure 5.1 Arrangement of an HTTP proxy to bridge pull and push based distribution. The upper part shows clients pulling CRLs directly from a CRL distribution point, whereas the lower part shows an arrangement using a MOM middleware and a HTTP proxy so that the client can pull and the server can push.

range of certificates it is representing. In a push based distribution, this association (represented by the service point URL) must be maintained.

A proxy program in (or near) the client can connect the two distribution schemes. The proxy has two service points: One that offers CRL pull through e.g. an HTTP protocol, and another that accepts CRL push through e.g. the SMTP/MIME protocols. CRLs received through the push protocols are stored according to their validity period and service point URL and offered to clients through the pull service port. This arrangement is shown in Figure 5.1.

For this arrangement to work, the clients' pull requests for CRLs must be directed to the proxy's service point, and not to the CA. This can be accomplished in at least three ways:

1. The value of the CRLdp certificate extension must contain the URL of the service point of the proxy
2. The proxy configuration of the client must be set to the service point of the proxy
3. The HTTP proxy must be integrated in the firewall, inspect all HTTP transactions and intercept those which are addressed to the CA's CRL distribution point

The two latter options in the list require that the HTTP proxy also acts as a "normal" web proxy.

5.1.1 Traffic volume considerations

In the client's end, the two different distribution alternatives shown in Figure 5.1 does not make any difference in terms of traffic volume. The client sends requests for a CRL when needed and gets a CRL in return.

In the server end, however, the arrangement makes a big difference since the CA transmits only *one copy* of the CRL to the MOM distribution architecture each time a new CRL is issued. In the basic arrangement (upper part of the Figure 5.1), the CA transmits one CRL copy for every client request. The number of requests for CRLs per issue scales with the number of client *and the number of applications*, since application which need certificates and CRLs tend to store them separately.

Clients are assumed to fetch CRLs when needed, which is at the first validation operation after the old CRL has expired. A high density of requests are therefore expected shortly after the expiration of a CRL since all clients will need new CRLs at approximately the same time. A proxy as outlined in Figure 5.1 will reduce the size of the traffic peak as well as reduce the total traffic volume.

5.1.2 Traffic calculations

The numbers used in subsequent calculations are taken from Table 2.1 and Table 4.1 in addition to the values listed below.

Size of CRL (s_B)	103 kB
Number of relying parties	400
Validation rate (λ_{tot})	400 per day = 0.0046 per second

5.1.2.1 Pull distribution, no proxy

The peak server traffic following a new CRL issue (pull distribution, no proxy arrangement) has been analyzed by Cooper [5] and found to be

$$R(t) = Nve^{-vt} \tag{5.1}$$

where $R(t)$ denotes the server-side CRL fetch rate at time instance t relative to the new CRL issue instance. N denotes the number of relying parties, which in practice means the number of running clients multiplied with a number of client applications with separate CRL stores. v denotes the client validating rate, which is denoted λ_{tot} elsewhere in this section.

The plot of server side traffic rates resulting from the clients' request for new CRLs is shown in Figure 5.2. The total volume for the CRL issue (every 4 hours) distribution is 103 kB * 400 = 41.2 MB, leading to an average data rate of **2.86 kB per second**.

On the client side, a CRL (103 kB) has to be fetched by the client every 4 hours, which results in an average rate of **7.15 bytes per second**.

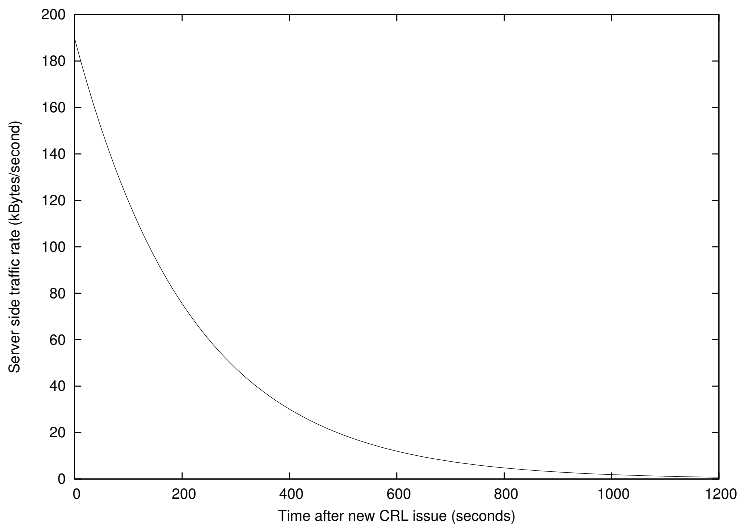


Figure 5.2 Server side traffic rates due to pull operations of new CRLs

5.1.2.2 Push distribution with MOM and a proxy

When push distribution is employed, the ideal arrangement would allow the middleware to send only one copy of the CRL over a network link. Since the transmission is scheduled by the CA server, there exists no traffic peak as shown in Figure 5.2. The total server side traffic is limited to one CRL copy (103 kB) every 4 hours, leading to an average data rate of **7.15 Bytes per second**. The client traffic traffic rate is the same number.

5.1.2.3 Parameter sensitivity

The advantage of push based distribution (in terms of traffic volume reduction) grows with the number of relying parties (which is 400 in this example).

The traffic rates are also sensitive to the issuing intervals of CRLs. A reduction of issuing interval by a half will double the traffic rates.

5.2 Delta CRLs

A *delta CRL* lists the certificates that have been revoked since the last issue of the *base CRL*, see Section 4.3.2 for a fuller discussion. The effect of the use of delta CRL will now be analyzed.

COTS programs (Firefox, Thunderbird, Adobe Acrobat etc.) are often found to disregard delta CRLs. The use of delta CRLs may therefore require modified or custom built software.

An aspect of the use of CRLs is how to deal with "new born" clients which need their initial revocation list to start their operation. When delta CRLs are in use, the new born clients will require both the most recent CRL as well as all subsequent delta CRLs, as outlined in Figure 4.4. It is not reasonable to expect them to sit still and wait for regular CRL issues, so a separate pull based service is necessary to distribute most recent revocation info.

5.2.1 Traffic volume considerations

One may replace e.g. 3 out of 4 CRL issues with delta CRLs, which means that the base CRL traffic is reduced with 75 percent. The size of delta CRLs may be calculated on the basis of Equation 4.21 in Section 4.3.2.

5.2.2 Traffic calculations

Using the same values as in Table 4.1, we assume a CRL issue interval of 4 hours, with 3 delta CRL issues between base CRL issues. The 3 delta CRLs must hold references to the certificates revoked during the last 4, 8 and 12 hours. Revocation rate is 0.1 (p_0) over a period of 1 year (t_0). The value of k becomes $4/8760$, $8/8760$,$12/8760$, respectively. The number of certificate references in the three delta CRLs together can be calculated as follows:

$$r_{rev}(\frac{4}{8760}) + r_{rev}(\frac{8}{8760}) + r_{rev}(\frac{12}{8760}) = 1.44 \cdot 10^{-4} \tag{5.2}$$

With a chosen certificate population of 50000 certificates, the aggregate size of the three delta CRLs is 2.359 kB. The relative improvement is expressed in Equation 4.23 and can be calculated to:

$$\frac{103\text{kB} + 2.359\text{kB}}{4 \cdot 103\text{kB}} = 0.256 \tag{5.3}$$

I.e., a reduction of 74 percent in CRL traffic volume, which is a substantial improvement. The traffic rate under these conditions is for the client side 105.4 kB$/16$ hours $= \mathbf{1.83}$ bytes per second.

On the server side, the traffic rates are 1.83 bytes per second if push distribution is employed, otherwise 1.83 bytes per second $\cdot\, 400 = \mathbf{732}$ bytes per second.

5.2.3 Parameter sensitivity

In the same manner as for base CRLs, the traffic rates are sensitive to the issuing intervals of CRLs. A reduction of issuing interval by a half will double the traffic rates. The push based distribution mechanism will likewise profit from a larger population of relying parties.

5.3 Timely caching of OCSP results

An OCSP transaction is initiated by a client during a validation operation. The purpose of the OCSP service is to report the revocation status of a certificate. The service endpoint of the OCSP responder is determined by the client through the AIA[11] extension of the certificate, or through a “per-issuer” table in the validating application.

A scheme for caching OCSP responses has been outlined in Section 4.1. The OCSP proxy will cache recent results from certificate validations and resolve a fraction of the validation requests locally as shown in Figure 4.1. Under the conditions on which Figure 4.3 is based, a cache will resolve *10 percent* of requests after 2-3 days of operation.

[11] Authority Information Access

The OCSP invocations from the client must be redirected to the OCSP cache proxy, through the value in the AIA extension, through client configuration or through interception of the network operations in the firewall.

5.3.1 Traffic calculations

OCSP-based validation does not exhibit peak traffic rates like what was shown in the CRL distribution analysis in Section 5.1.1. Rather, a fraction of the validation requests is sent to the central OCSP service for processing. Based on the values in Table 2.1 the traffic volumes will now be estimated both for “classic” and “cached” OCSP service.

5.3.1.1 Classic OCSP service

An OCSP transaction (request and reply) have been observed to consume 2800 bytes of network capacity. A rate of 400 validations per day and 400 relying parties, the total average traffic in the server end is **5.19 kB per second**. In the client end, the validation traffic generates **13 Bytes per second** on average.

5.3.1.2 Cached OCSP service

An OCSP proxy with a 10 percent “hit rate” (based on Figure 4.3) reduces the traffic to **4.67 kB per second** in the server end. The average OCSP traffic between the client and the cache will remain unaffected as **13 Bytes per second**.

5.3.1.3 Parameter sensitivity

The effect of the OCSP cache will be higher with a smaller certificate population (currently 50000), longer expiration lifetime (currently 4 hours) and higher validation rate (currently 400 per day).

5.3.1.4 Cooperative caching

The cache hit rate in the example above is not promising, since it only reduces the traffic with 10 percent. A higher validation rate will improve the results, something that can be obtained if the cache is shared among several validating clients.

At a validation rate of 4000 per day the cache will resolve 20 percent of the requests locally (calculations based on Equation 4.12). A group of 10 clients which share an OCSP cache will achieve this validation rate. Similarly, a cache which is shared among 100 clients will resolve 30 percent of the requests.

5.4 Short lived certificates

An alternative to revocation checking is to issue certificates with a short lifetime. If the lifetime is the same as the normal CRL issue interval (e.g. 4 hours), then revocation checking makes little sense. The same key pair is re-certified by the CA in a series of short lived certificates, unless the key pair is invalidated. This approach was discussed and analyzed in Section 4.5

Please keep in mind that re-certification of key pairs is not allowed for “end entities” according to regulations from Norwegian National Security Authority (NSM) [1]. This approach therefore requires changes to existing security policies. Besides, COTS software has not been observed to re-load expired certificates automatically.

5.4.1 Traffic volume considerations

Revocation checking is not necessary when short lived certificates is in use, but clients still need the certificates in order to construct and to verify signatures. Frequent distribution of new certificates will require network capacity which must be weighted against the savings in CRL distribution.

The push distribution scheme which was proposed in section 5.1 is not well suited for certificate dissemination, since the number of certificates (50000 in our examples) is too high to justify a pro-active distribution. Only a few of the replicated certificates would be used, the other would just waste bandwidth.

Certificates are fetched on demand as needed by clients. The signer will need the certificate in order to construct the signature structure, and the validating party will need the certificate for signature verification. Some COTS programs choose to include the signer’s certificate in the signature structure, in which case the receiver does not need to retrieve the certificate in a separate operation.

On a small scale, the clients can fetch the required certificates directly from a central distribution point (the CA). On a larger scale, a replicated certificate retrieval system should be deployed. A passive replication system, also known as a *distributed cache*, is well suited for that purpose. One possible distributed cache arrangement is illustrated in Figure 5.3 and shows how caches may form a tree with the authoritative storage (CA) at the root of the tree. Clients send their certificate retrieval requests to the nearest cache, which either returns the certificate if it is in store, otherwise passes the request on to the parent cache, and stores the result locally upon return.

This arrangement ensures that one certificate is sent to the child cache only once, maintaining the scalability of the distribution system. This arrangement is quite similar to what was proposed for CRL distribution in Section 5.1, but exclusively uses the pull distribution mechanism in order to maintain the on-demand retrieval properties. The traffic peak associated with the CRL expiration which was discussed in Section 5.1 does not occur under these circumstances since the certificates expire on “random” time instances, and not at the same time.

The distributed cache arrangement shown in Figure 5.3 has an effect only in those cases where the same certificate is required multiple times. This is the case where the validating client (the receiver) need to retrieve the signer’s certificate. In applications where the signer’s certificate is enclosed as a part of the signature structure, each certificate is retrieved only once (by the signer), and the caching arrangement does not make an improvement.

Also keep in mind that on-demand distribution results in a larger *latency*, since the first requests

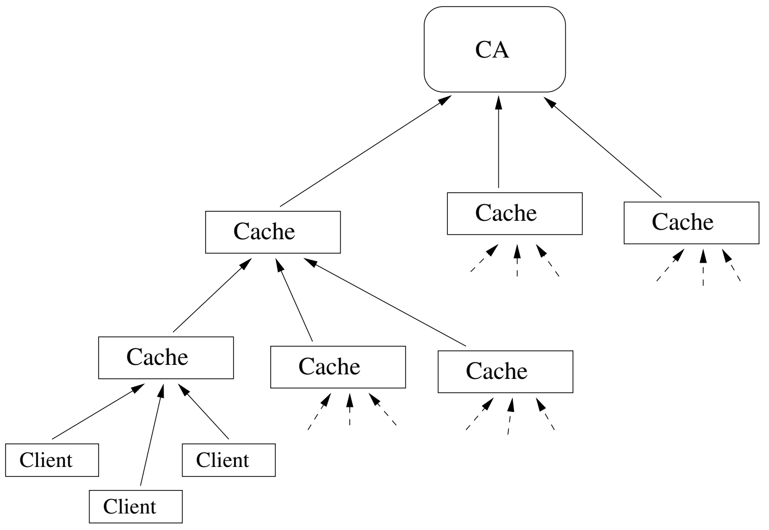


Figure 5.3 A cooperative caching system for the distribution of certificates in an on-demand manner

need to propagate all the way up the CA. The latency will likely be proportional to the height of the tree, which scales to the logarithmic value of the number of caches (as long as the tree is balanced).

5.4.2 Traffic calculations

Every client who wish to create a signature will need to obtain its own certificate. Based on the chosen example values in Table 2.1, we assume that the 400 active clients are sending signed messages and need to fetch their new certificates as the old one expires (every 4 hour).

A certificate retrieval operation generates approximately 2500 bytes of network traffic. Even though this operation is repeated on many tiers of the distribution tree (see Figure 5.3), they are repeated on disjoint links and do not generate more traffic than a single route between the client and the CA.

The receiver of the message will need the sender's certificate in order to verify the signature. The certificate is often found to be included in the certificate structure (see Section 5.5), in which case no retrieval operation is necessary. In other cases, the situation is similar to what was analyzed in Section 5.3, since the receiver holds a timely cache of certificates associated with a sender. The certificate population (50000), certificate lifetime (4 hours), number of relying parties (400) and rate of received messages (400 per day) is the same as in Section 5.3, so the graph in Figure 4.3 can be used to determine that the receiver will need to retrieve the sender's certificate for *90 percent* of the received messages (after 1000 received messages).

Under these conditions, a separate retrieval mechanism in the receiver is *not* an improvement over the inclusion of certificates in each message. The calculated traffic volumes are:

5.4.2.1 Client end

Every client is assumed to fetch its certificate during the 4 hour validity period. A certificate retrieval operation is estimated to consume 2.5 kB of network capacity. In addition, it is expected to retrieve certificates for 90 percent of the received messages unless they are included with the signature (400 messages per day).

Certificate included Only retrieval of own certificate is necessary. The traffic rate becomes

$$\frac{2.5\text{ kB}}{4\text{ hours}} = 0.17\text{ bytes per second} \tag{5.4}$$

Certificate not included Without the sender's certificate in the signature, the client will need to obtain the sender's certificate in 90 percent of all messages, which is 400 per day. The traffic rate becomes

$$\frac{2.5\text{ kB}}{4\text{ hours}} + \frac{2.5\text{ kB} \cdot 400 \cdot 0.9}{24\text{ hours}} = 10.6\text{ bytes per second} \tag{5.5}$$

5.4.2.2 Server end

Certificate included The server only need to respond to one request per active client over the expiration period, *regardless the use of a distributed cache*. The traffic rate becomes

$$\frac{2.5\text{ kB} \cdot 400}{4\text{ hours}} = 69\text{ bytes per second} \tag{5.6}$$

Certificate not included In this case, also receivers will need to retrieve certificates for 90 percent of the received messages. If no distributed cache is in operation, the server traffic is equal to the aggregated client traffic and the rate becomes

$$\frac{2.5\text{ kB} \cdot 400}{4\text{ hours}} + \frac{2.5\text{ kB} \cdot 400 \cdot 400 \cdot 0.9}{24\text{ hours}} = 4.2\text{ kB per second} \tag{5.7}$$

With a distributed cache in place, each certificate will only be requested once from each child cache, a number which is given by the *degree* of the distribution tree. For degree k the traffic rate becomes k times Equation 5.5. E.g., $k = 3$ gives a server side traffic rate of 31.8 bytes per second.

Only the use of included certificates will be listed in the final comparison in Table 5.1, since the calculations in Section 5.5 concludes that this most efficient.

5.4.3 Parameter sensitivity

The traffic rates associated with the use of short lived certificates are mostly relying on the certificate lifetime. A shorter lifetime will increase the traffic rates.

5.5 Compressed message signature structure

Part of the PKI resource demand is not related to validation/revocation, but to the increased sizes of application messages (e.g. e-mail messages) due to signatures and certificates. The networking cost of PKI operation may be reduced through the reduction of message overhead, which is why this aspects are brought up for discussion.

Although a digital signature can take several different forms, an often seen structure includes the certificate (approx. 1200 bytes), the message hash (32 bytes[12]) and the signature value (256 bytes[13]). For inclusion in text messages, a BASE-64 encoding may be used, and will increase the structure size with 33 percent up to 2 kilobytes.

Larger signature structures include the XML-DSig standard, which also communicates the choice of hashing and crypto algorithm, and possible transformations on the associated XML document. XML-DSig based signatures easily takes up 6-10 kilobytes of information.

A selection of observed signature sizes is shown in Table 2.1. The sizes varies widely from the most reasonable to Adobe Acrobat's signature which is suspected to include validation proof as well (in form of a CRL or OCSP response).

Signature elements are included for reason of interoperability (algorithm choices), convenience (certificate, message hash) or proof (signature value). The signature value is the only *essential* element; the message hash must be calculated by the receiver anyway, the choice of algorithm can be made by design, and the receiver may be in possession of the sender's certificate already. In case the certificate is left out of the signature structure, a reference to it must be included, e.g. by its serial number (8 bytes).

Modification of the signature structure requires modified software, and is therefore not a viable strategy where COTS products is a requirement. In tactical systems however, the software is likely to be tailor-made for that purpose, and a modified signature structure is an valid option.

Signature structure compression is not an alternative to the use of revocation status, but may be used in combination with the other optimization techniques being discussed in the report. It is therefore left out of the comparison in Table 5.1

5.5.1 Traffic volume considerations

The message size can be reduced with a few number of kilobytes if non-essential parts of the signature is left out. This may lead to a much better utilization of the message channel in those cases where the payload of the message is small,

In a message which consists of 80 bytes message text and 4 kB signature the signature takes up 98 percent of the size, making the signature overly expensive. If the size of the signature could be

[12]Using SHA-256

[13]Assuming 2048 bit RSA key

reduced to 256 bytes, the overhead would be reduced to 76 percent, and the utilization *increased 12 times* from 2 to 24 percent.

On the other hand, there is a chance that the receiver need to obtain the sender's certificate if this is the first time it receives a message from this sender. This will happen often as the receiver starts its operation with an empty certificate store, but less frequent as the store is filled. The analysis in Section 4.2 and in particular Equation 4.12 can be used to model this probability.

In order for the receiver to obtain the sender's certificate, the certificate reference (serial number, 8 bytes) included with the signature must enable the receiver to locate the certificate. Any retrieval protocol can be used for that purpose, e.g. HTTP or LDAP. An HTTP transaction returning a certificate of 1200 bytes may generate approximately 2.5 kB of network traffic. Section 5.4 discusses a cooperative caching system for scalable distribution of certificates.

5.5.2 Traffic calculations

The numbers used in subsequent calculations are taken from Table 2.1 and Table 4.1, and used in Equation 4.12. The calculation results presented in Section 5.4 will change due to the longer lifetimes of certificates; the certificates do not expire after 4 hours in this calculation, but after 1 year. A receiver will consequently cache a certificate for this amount of time.

After 1000 received messages (2-3 days of operation), the receiver will need to retrieve the sender's certificate for *58 percent* of the received messages (calculation based on Equation 4.12 and $t_e = 1$ year).

The 1.3 kB reduction in messages size achieved by leaving out the sender's certificates (1200 bytes) and message hash (32 bytes) must be weighted against the 0.58 probability that the certificate must be retrieved at the "costs" of 2.5 kB in network traffic.

The result is

$$(1.3\text{ kB} - 0.58 \cdot 2.5\text{ kB}) \cdot 400/\text{day} = -60\text{ kB/day} \tag{5.8}$$

which is negative, meaning that leaving out certificates from the signature structure is a loss under the chosen parameter values.

Improvements may be found under different conditions, which may be identified using the methods just demonstrated.

5.6 Issues of connectivity, cross-domain operation and multi tier CA

So far, this section has investigated alternative revocation checking mechanisms from the perspective of network traffic volume. Alternatives which require modifications to software or security policies have been identified.

There are still aspects of the alternatives that should be addressed, however, since it affects the feasibility of the solution:

5.6.1 Connectivity aspects

A distributed system that can continue its operation even during loss of communication with other computers is considered to more robust and resilient, so this property is generally desired.

A client which receives a message has obviously a connection with the sender or a message exchange (otherwise it would not receive it). The following signature verification may involve invocations of services (like a OCSP provider or a certificate store) on which the receiver depends to complete the validation. Otherwise, the receiver may complete the validation based on data previously fetched (CRLs) or based on data contained in the message itself (short lived certificate included with the signature).

In this perspective, the use of online status and certificate providers seems less attractive than the use of CRLs and short lived certificates.

5.6.2 Cross-domain operation aspects

This report has not conducted any detailed investigation or calculation on cross-domain validation scenarios. During cross-domain validation, which means checking the revocation status of a certificate issued by a different domain (CA), the revocation check must take place based on lists or status providers from that foreign domain.

Cross-domain revocation checking seems to be almost an insurmountable problem. Where CRLs are in use, all cooperating domains must exchange their revocation lists, which means that potentially huge lists may be imported in a low bandwidth network and cause resource outage. Where revocation status providers are being used, the required connectivity for clients in one domain to access servers in a different one may cause concern over security and access control.

The best arrangement for certificate validation in cross-domain scenario is to avoid the use of auxiliary information sources or service providers. It seems that the use of short lived certificate has the best potential for successful operation under these conditions.

5.6.3 Multi tier certificate authorities

During the discussion in this section the assumption has been made that the CRLs and the OCSP responses have been issued by the trust anchor, in which case the validation operation can be conducted on the basis of one CRL/OCSP response. This is a simplification. In a real PKI of large scale, there exists a multi tier tree of certificate authorities, generating a *certificate path* from the certificate being validating to the trust anchor of the client.

The certificates from the intermediate CAs also need to be validated, but these certificates will appear in validation operations far more often than “leaf” certificates, and will consequently be better suited for caching arrangements. Caching of OCSP responses as discussed in Section 5.3 could be profitable with the frequency and lifetime of intermediate CA certificates.

Optimization of a multi tier PKI will not be analyzed in detail in this report, but it is likely that a combination of the techniques discussed in this section would provide the best results.

5.7 Conclusion

Table 5.1 summarizes the discussion of different optimization alternatives. They are listed row wise in the order that they have been discussed in this section. The different columns are to interpreted as:

Client traffic Traffic rates in each client related to validation, given as bytes per second.

Server traffic Traffic rates in central client (CA), given as bytes per second.

Connectivity demand An indication of the clients dependency on frequent connection to a central server in order to complete a validation or renew a certificate.

Traffic variability An indication of the server's tendency to experience high traffic peaks.

Optimization alternative	Client traffic	Server traffic	Connectivity demand	Traffic variability
Pull based CRLs	7.15 Bps	2860 Bps	Medium	High
Push based CRLs	7.15 Bps	7.15 Bps	Medium	High
Delta CRLs (push)	1.83 Bps	1.83 Bps	Medium	High
Basic OCSP	13 Bps	5190 Bps	High	Low
Cached OCSP	13 Bps	4670 Bps	High	Low
Short lived certificates (certificate included)	0.17 Bps	69 Bps	Medium	Low

Table 5.1 Traffic volumes related to certificate validation under the different alternatives

Traffic variability for all CRL schemes are classified as "high" due to both the traffic peak explained in Section 5.1.2.1 and the fact that a large object (the base CRL) will itself generate a traffic peak during its distribution.

Connectivity demand is set to "high" where most validation operations require a connection to auxiliary services (e.g. a revocation status provider), and set to "medium" where such connections are necessary more infrequently.

The data in Table 5.1 clearly indicates the use of short lived certificates as the alternative that generates the least traffic and is only moderately dependent on connectivity to central resources. Besides, this alternative does not require a separate distribution infrastructure to be deployed, although a modified certificate policy must be established. A solution to the validation of intermediate CA certificates must also be decided.

On the other hand, the use of push based CRLs as discussed in Section 5.1 offers an substantial improvement over the naïve pull based CRL distribution, and requires no changes to software or certificate policy. The efforts associated with the development of a HTTP proxy and deployment of a distribution middleware may be substantial, however. Besides, cross domain validation poses a scalability problem for this approach.

6 Summary and conclusions

The purpose of this report has been to investigate optimization techniques for PKI operation. Some background information have been provided in the form of a general introduction to public key cryptography and optimization of distributed systems.

A more focused analysis of validation scenarios has been provided together with an investigation of a list of candidate optimization efforts. The investigation has concluded with numbers on average network traffic, evaluation of traffic variability, connectivity demands and suitability for cross domain operation.

The use of short lived certificate seems to be the optimization technique that consumes the least network capacity, has moderate connectivity demand, works in a cross domain environment and is a conceptually simpler mechanism for signature verification since the entire verification process relies only on one object (the certificate), whereas other alternatives rely on two objects (certificate and revocation status).

References

[1] *Digital Certificates and PKI version 2.0.* Norwegian National Security Authority (NSM), 2007. https://www.nsm.stat.no/Documents/Veiledninger/Digital%20Certificates%20and%20PKI%20v2.0.pdf.

[2] Andre Aarnes, Mike Just, Svein J. Knapskog, Steve Lloyd, and Henk Meijer. Selecting revocation solutions for PKI. In *Proceedings of NORDSEC 2000 Fifth Nordic Workshop on Secure IT Systems*, Reykjavik, Iceland, 2000.

[3] Albert-Laszlo Barabasi. *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Plume Books, April 2003.

[4] S. Berkovits, S. Chokhani, J. Furlong, J. Geiter, and J. Guild. Public key infrastructure study: Final report. *Produced by MITRE Corporation for NIST*, April 1995.

[5] David A. Cooper. A model of certificate revocation. In *Proceedings of the 15th Annual Computer Security Conference*, December 1999.

[6] David A. Cooper. A more efficient use of delta-crls. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 190, Washington, DC, USA, 2000. IEEE Computer Society.

[7] Anders Fongen. Xml based certificate management. Technical Report 2008/00278, Forsvarets Forskningsinstitutt, 2008.

[8] Anders Fongen. Scalability analysis of selected certificate validation scenarios. In *IEEE MILCOM*, pages 1–7, San Diego, CA, USA, Nov. 2010.

[9] Dinko Hadzic, Trude Hafsøe, Frank Trethan Johnsen, Ketil Lund, and Kjell Rose. Web services i nettverk med begrenset datarate (in norwegian). Technical Report 2006/03886, Forsvarets Forskningsinstitutt, 2006.

[10] D. Pinkas and R. Housley. Delegated path validation and delegated path discovery protocol requirements, 2002.

[11] Ronald L. Rivest. Can we eliminate certificate revocations lists? In *Financial Cryptography*, pages 178–183, 1998.

[12] Adam J. Slagell and Rafael Bonilla. PKI scalability issues. *Computing Research Repository*, cs.CR/0409018, 2004.

[13] Adam J. Slagell, Rafael Bonilla, and William Yurcik. A survey of PKI components and scalability issues. In *Proceedings of the 25th IEEE International Performance Computing and Communications Conference, IPCCC*, Phoenix, AZ, 2006.

[14] Eli Winjum and Anders Fongen. Model and specification for analyzing the scalability of a public key infrastructure. Technical Report 2009/01546, Forsvarets Forskningsinstitutt, 2009.