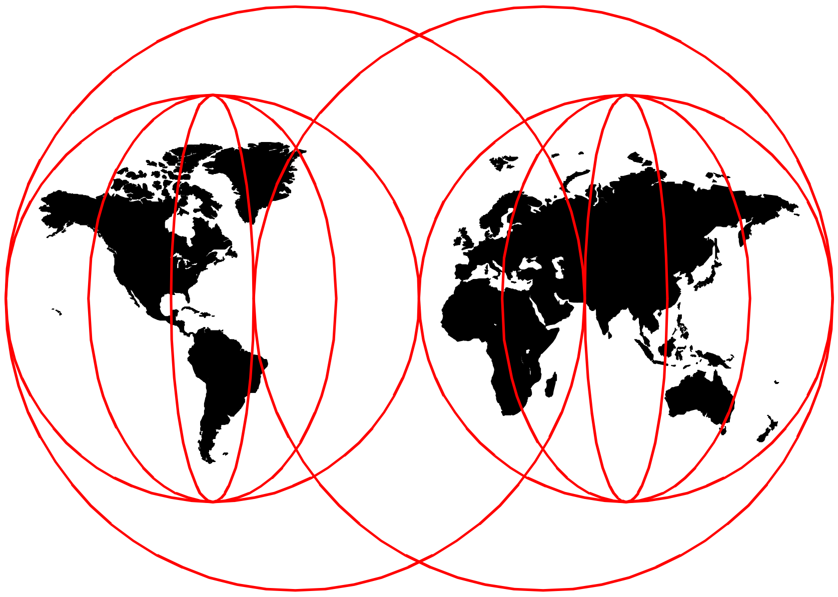
Deploying a Public Key Infrastructure

Heinz Johner, Seiei Fujiwara, Amelia Sm Yeung, Anthony Stephanou, Jim Whitmore



International Technical Support Organization

www.redbooks.ibm.com

SG24-5512-00

SG24-5512-00

International Technical Support Organization

Deploying a Public Key Infrastructure

February 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special notices” on page 217.

First Edition (February 2000)

This edition applies to Release 3.1 of the IBM SecureWay Trust Authority product.

This document created or updated on February 29, 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures

Preface

In the recent past, applied security has gained focus across the I/T industry as the share of *electronic* business conducted over the Internet, as opposed to traditional business, keeps raising drastically.

The terms *public key cryptography* and *Public Key Infrastructure* (PKI) are not new to the I/T industry. The practical use, however, has mostly been limited to cryptography at the communication protocol level, such as in the widely used SSL protocol. In theory, public key cryptography deployed in a PKI provides a very high level of security and there is little doubt throughout the industry that a fully deployed PKI overcomes most of the current security and management issues that organizations are concerned about. This redbook explains the theory, the concepts, and the practical implications of a PKI, and thus, how a PKI can be deployed within an organization. It helps you understand, design, and create a solution that utilizes public key technology.

Also contained in this book is a comprehensive overview of the IBM SecureWay Trust Authority product that supports a PKI. Its features, installation, and configuration are explained.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Heinz Johner is a technical professional and author at the International Technical Support Organization, Austin Center. He writes extensively on security-related areas, eNetwork deployment, and the Distributed Computing Environment (DCE). Before joining the ITSO, he worked in the Professional Services organization of IBM Switzerland and was responsible for systems management and DCE in medium and large customer projects.

Seiei Fujiwara is an Information Technology Specialist at the Advanced Technical Support in IBM Japan. He has 11 years of experience in S/390 products technical support. His areas of expertise include OS/390 UNIX System Services, WebSphere Application Server, and S/390-related Internet security functions. He has written extensively on S/390 e-business solutions.

Amelia Sm Yeung is an Information Technology Specialist at IBM Hong Kong. She has two years of experience in providing e-business solutions to banking customers and government. She has worked at IBM for three years.

Her areas of expertise include SNA communications, TCP/IP, platform design, AIX, OS/2, VisualAge for Smalltalk, and electronic commerce.

Anthony Stephanou is an Information Technology Specialist at IBM South Africa. He has two years experience in providing security solutions to the banking finance and securities industry. He has worked in a R and D environment for microcontroller-based real-time solutions. His areas of specialization include hardware-based cryptographic solutions and IT security.

Jim Whitmore is a Solution Architect within the networking and I/T security disciplines. Over the past eight years, Jim has led consulting, architecture, and integration services engagements for both government and industry, including intranet design and deployment, Internet connectivity with firewalls, and e-commerce. He is a member of the board of Certified I/T Architects within IBM Global Professions, currently working on a methodology for architecting secure solutions that incorporate international standards for common criteria.

Thanks to the following people for their invaluable contributions to this project:

Messaoud Benantar	PKIX Development, IBM Austin
Tom Gindin	Trust Authority Design and Support, IBM Gaithersburg
Amy Jones	PKI Solution Center, IBM Gaithersburg
Ron Kenney	PKIX Development, IBM Austin
Denise Mallin	PKI Solution Center, IBM Gaithersburg
Amy Wang	Trust Authority Development, IBM Gaithersburg
Taoling Xie	PKI Solution Center, IBM Gaithersburg

Special thanks go to the editor for her invaluable help in finalizing this redbook:

Elizabeth Barnes	ITSO, Austin Center

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks evaluation” on page 237 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Part 1. PKI overview and deployment

Chapter 1. Introduction to security systems

Security has become a major focus throughout the entire I/T industry. With the advent of the Internet and the new ways of doing business it enables, organizations experience some market pressure to be present on the World Wide Web, but, at the same time, have to ensure that such connections to the outside world do not pose any risks to them. Even inside the organizations, information that is available on interconnected computers may need to be protected from unauthorized access. This fact, in turn, requires methods that protect information in transfer and means are required that uniquely identify individuals.

Several technologies are available today and even widely used that solve part of these problems, such as the encryption of data for secure data transmission or the use of personal passwords for authentication. Most of these systems, however, are islands in the whole and it has become clear that security must be designed and implemented from a higher-level perspective.

1.1 Recap: Security basics

The requirement and basic principles for security solutions can easily be derived from a few fundamentals. Computer systems are used to process increasing amounts of data in a fast, convenient, and reliable way. Most of the data is in some way important to individuals for performing their job duties, or to a whole organization for conducting their business. In fact, most business data is handled by computers nowadays, which includes electronic mail, asset information, customer databases, process control information, and so on. Some of the data is considered critical because an organization would suffer damage if the data was lost or incorrect.

Just like critical or classified paperwork, it is desirable that not all data is available to anyone in an organization. The same is true for the computer systems themselves because their proper functioning becomes critical for companies. Access to computer systems is not limited by physical access to the computer itself because computers are normally connected through an any-to-any network. An issue that comes up is improper use of computers, be it inadvertent or malicious. Improper use of computers also potentially means access to critical data. Trust in people has proven to be not enough; additional means need to be put in place to ensure that only authorized individuals can access the computer systems and data. This requires the two fundamental security services: *authentication* and *authorization*. An individual (or even a computer system or service) must be authenticated before any checks can be done as to whether or not any requested access or operation is allowed. This, however, does not prevent data in transit, say using a network connection, from being spoofed or even altered. Even data stored on a system's local storage device may not be safe if multiple users share that system. Thus, additional means need to be in place to protect data. This is even more the case when data represents legal obligations and rights. *Encryption* is a technology that can, if properly used, protect data from being disclosed to unauthorized parties. Sometimes, especially when legal obligations are involved, means must be in place such that a creator of a document cannot later decline to have created that document. This is called *non-repudiation*. These are the most important terms used in computer security as far as public key cryptography is concerned. The following is a more accurate and more complete list of terms that will be used throughout this book.

The world of security has its own set of terminology. The International Organization for Standardization (ISO) has defined the common security services found in modern I/T systems. The list was first put in ISO 7498-2 (OSI Security Architecture) and later updated in ISO 10181 (OSI Security Frameworks). To have a better understanding of security systems and services, some security terms with explanations are listed below:

Authentication	Authentication is the process of verifying the validity of a claimed individual and identifying who he or she is. Authentication is not limited to human beings; services, applications, and other *entities* may be required to authenticate also.
Access Control	Assurance that each user or computer that uses the service is permitted to do what he or she asks for. The term *authorization* is often used as a synonym for access control, but it also includes granting the access or rights to perform some actions based on access rights.
Data Confidentiality	Sensitive information must not be revealed to parties that it was not meant for. Data confidentiality is often also referred to as *privacy*.
Data Integrity	Data integrity assures that the data is not altered or destroyed in an unauthorized manner.
Non-Repudiation	Assurance that a sender cannot deny being the source of a message, and that the recipient cannot deny the receipt of a message.

Availability	The availability (over time or in terms of response time) of a system. While this is normally not a security-related term, it may become so when dealing with denial of service attacks that may intentionally slow down or completely paralyze a system.
Security Audit	A process where an independent party checks the security level of an organization or computer system.
Key Management	Key management deals with the secure generation, distribution, authentication and storage of keys used in cryptography.

A *Public Key Infrastructure* (PKI) provides the technical framework (including protocols, services, and standards) to support applications with five security capabilities: user authentication, data confidentiality, data integrity, non-repudiation, and key management.

1.2 Secret key versus public key cryptography

If data is transmitted over the network intact, someone else on the network may be able to look at the data. This is very insecure. A better way is to scramble the data to other forms based on some criteria before the transmission. This is called *encryption*. When a recipient receives the scrambled data, he or she can change it back to something meaningful (if he or she knows the way to do so). This is called *decryption*.

Many algorithms have been developed to describe how to scramble (encrypt) data. *Cryptography* is the science of how to do encryption and decryption. The most widely used cryptographies are secret key cryptography and public key cryptography. They are used to build schemes or protocols to secure information transmission. This section explains and compares the differences between the two cryptographic techniques.

1.2.1 Secret key cryptography

The first commonly-used cryptography is *secret key cryptography*. The reason for this name is because any two communicating parties have to keep a secret between themselves of how to encrypt and decrypt the data. In secret key cryptography, a single key, called a *symmetric key* or a *secret key*, is used for both the encryption and decryption processes. That is why secret key cryptography can also be referred to as *symmetric cryptography*. This is illustrated in Figure 1 on page 6.

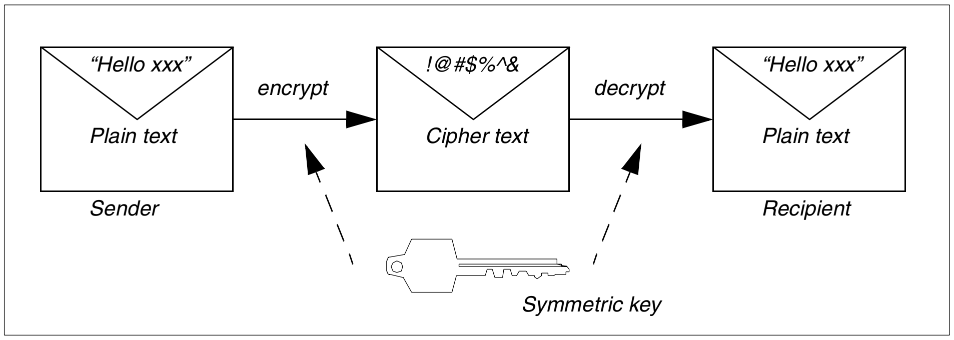


Figure 1. Secret key cryptography

Data confidentiality is achieved by using secret key cryptography, since a third party cannot understand the cipher data. However, how can one party exchange the secret key with the other party securely? In other words, how can the sender and the recipient agree on the secret key without it being disclosed to anyone else? This is a valid question and requirement because the communication path cannot be considered trustful; if it was, they would not need to encrypt their messages. Moreover, for each pair of communication partners, there will be one secret key. The management of keys needs extra care. For example, if one has to securely communicate with twenty other parties, one has to keep and manage twenty secret keys. These twenty keys have to be different from each other. Otherwise, if the sender is communicating with a recipient, a third party holding the same secret key may hack into the communication. Therefore, as the number of communication partners grows, the sender has to find some better way to maintain and keep the growing number of secret keys. He or she may need some directory structure to help.

However, there is one great advantage of secret key cryptography. Encryption and decryption using a secret key is usually fast and easy, due to a less complex computation when compared with public key cryptography (to be introduced in the next section). This is why secret key cryptography plays an important role for data encryption and decryption in many Internet security protocols, as in, for example, the Secure Sockets Layer (SSL) protocol.

Here are some commonly used examples of symmetric key cryptosystem:

- Data Encryption Standard (DES): 56-bit key plus 8 parity bits, developed by IBM in the middle 1970s
- Triple-DES: 112-bit key plus 16 parity bits or 168-bit key plus 24 parity bits (that is two to three keys)
- RC2 and RC4: variable-sized key, often 40 to 128 bits long

To summarize, secret key cryptography is fast for both the encryption and decryption processes. However, it is a nightmare for users to maintain a large number of secret keys, unless some kind of infrastructure can help users to store and manage keys in an easy manner.

1.2.2 Public key cryptography

Another commonly-used cryptography is called *public key cryptography*. In public key cryptography, an *asymmetric key pair* (so-called a *public key* and a *private key*) is used. The key used for encryption is different from the one used for decryption. Public key cryptography requires the key owners to protect their private keys while their public keys are not secret at all and can be made available to the public. The computation algorithm relating the public key and the private key is designed in such a way that an encrypted message can only be decrypted with the corresponding other key of that key pair, and an encrypted message cannot be decrypted with the encryption key (the key that was used for encryption). This is a very important design principle of public key cryptography, where messages are encrypted using a recipient's public key. If this encrypted message could be decrypted with the encryption key (that is the recipient's public key, which is obtainable in public), everyone could decrypt and read the message from the sender. So, an encrypted message in public key cryptography cannot be successfully decrypted by the encryption key. Figure 2 illustrates the principle of public key cryptography.

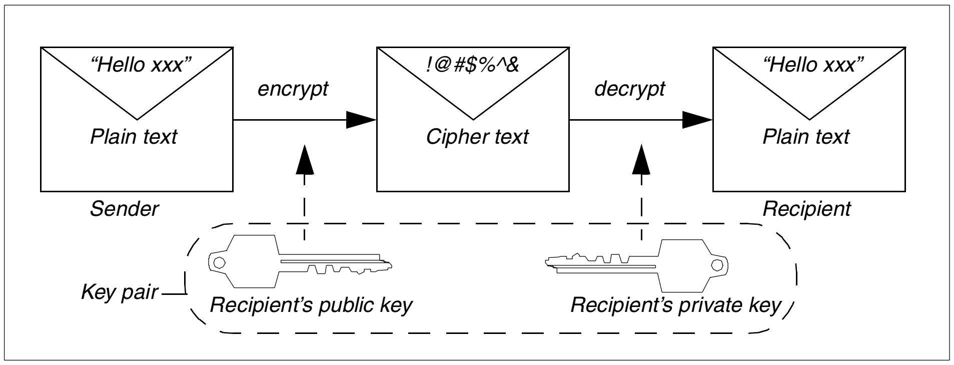


Figure 2. Principle of public key cryptography

Data confidentiality is again achieved by public key cryptography, since a third party cannot understand the cipher data. In the example shown in Figure 2, only the recipient can read the message from the sender because only he or she has the corresponding private key to decrypt the message. In fact, everybody can send encrypted messages to a recipient using the recipient's public key which, by definition, is publicly available. Only the recipient is able to decrypt these message because only he or she is in possession of the corresponding private key. However, the recipient cannot be sure if the message is really from a specific sender, since everybody can see the recipient’s public key and can fake to be that sender. In later sections, more detailed explanations will be given to show how public key cryptography is used for more secure communications.

In the example shown in Figure 2 on page 7, the recipient’s public key is used for encryption. However, public key cryptography is not limited to this use. The sender’s private key can also be used to encrypt messages. The recipient can then use the sender’s public key to decrypt and read the message. In this case, the recipient can be sure that the message is really from the sender, provided that the recipient can trust that the sender’s public key is really from the sender. When used in this way, the term *signing* is used rather than *encrypting* a message, although it is technically the same. One usage of this case is discussed in 1.3.2, “Digital signatures”.

The asymmetric key pair is generated by a rather complex computation. Even though the public key is widely distributed, it is practically impossible for computers to calculate the private key from the public key. This improves security, but implies a long encryption time, especially for large amounts of data. So sometimes, public key cryptography is not used to encrypt the whole message, but only to encrypt a small, fixed-length summary of the whole message. This is covered in more detail in 1.3.2, “Digital signatures”.

Here are a few commonly-used examples of public key cryptosystem:

- Rivest, Shamir, and Adleman (RSA): variable-sized key, generally 512-2048 bits
- Elliptic curves: variable-sized key, 160 bits, comparable security to RSA 1024 bit

One may think that public key cryptography is a replacement of secret key cryptography. In fact, these two cryptographic techniques can supplement each other to achieve good security while not sacrificing speed and performance.

To summarize, public key cryptography can improve security, convenience, and flexibility. Public keys can be distributed freely and users do not have to manage many keys. Users have the responsibility to keep their private keys safely. Users cannot deny what they have sent out if they have encrypted the message with their own private keys. This provides for non-repudiation. The main disadvantage of public key cryptography is its computing-intensive encryption and decryption.

1.3 Basic elements of public key cryptography

Public key cryptography is an important concept throughout this book. In 1.2, “Secret key versus public key cryptography”, public key cryptography was introduced. Descriptions have also been given on how a message is sent from one party to another party using public key cryptography. However, there are some unresolved issues. For example, if a sender uses a recipient’s public key to encrypt and send a message, even though the recipient can use his or her private key to open and read the message, the recipient has no idea of the real identity of the sender. Another unresolved issue is the following: when a sender signs (encrypts) a message with his or her private key, and the recipient decrypts the message with the sender’s public key, although the recipient can read the message, the recipient cannot be sure whether the sender’s public key is really from the sender, or from someone else, since no one can certify to the recipient that this public key is really from that sender.

In this section, we describe how to apply the public key cryptography to other security designs, such as digital signatures and certificates, so as to resolve the above issues. This section details public key cryptography more from a user’s perspective, that is, only the basic components and usages will be covered.

1.3.1 Keys

The term *key* refers to some information (usually a binary number of a specific length) that is used with a cryptographic algorithm for encryption and/or decryption purposes. During data encryption, the key affects how the data is scrambled. During data decryption, only the correct key is allowed to get back the original data. This basic concept is illustrated in Figure 3 on page 10. For secret key cryptography, the encryption key is the same as the decryption key. For public key cryptography, the encryption key is different from the decryption key.

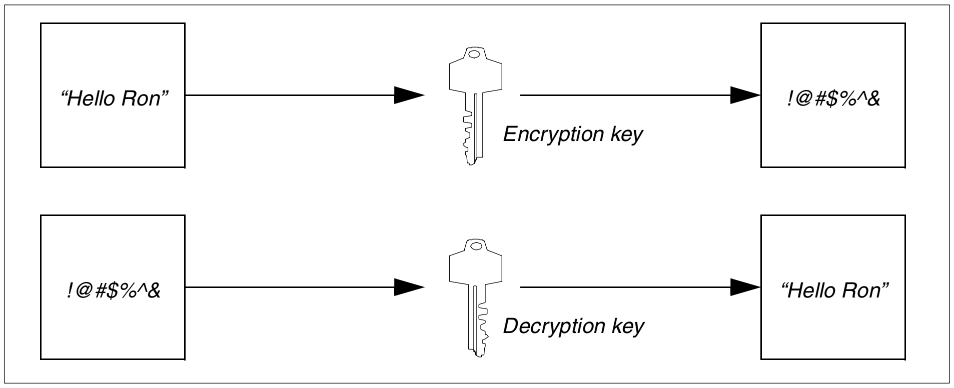


Figure 3. Encryption keys versus decryption keys

There has been a common misconception that by breaking the cryptographic algorithm all the messages can be decrypted. Please note that cryptographic algorithms are usually widely known or can be broken. Therefore, it is not the cryptographic algorithm that protects the data, but much more the key. For secret key cryptography, the key has to be kept securely by the communicating parties. For public key cryptography, the private key is kept by the key owner in a safe place. Moreover, the public key and private key are generated by complex computations. For both cases, the keys are very important for the encryption and decryption processes. A private key should not be easily calculated from other known information, for example, the well-known public key or the published algorithm. Thus, decryption without the correct key is very difficult or even impossible for all practical purposes.

1.3.2 Digital signatures

From previous descriptions, it is learned that keys play an important role in the encryption and decryption processes. However, keys alone may not always be trusted. This can be seen from the public key cryptography example in 1.2.2, "Public key cryptography". When a recipient's public key is used to encrypt a message, and the recipient uses his or her private key to open the encrypted message, he or she cannot be sure about the real identity of the sender. Everybody can see the recipient's public key and can send him or her a message. However, the recipient is not sure if the sender is using his real identity to send or not. For the case of secret key cryptography, if a recipient receives a message from the sender, the message may actually be from an non-trusted party, since the secret key may have been disclosed to a party other than the sender.

To have better authentication, the idea of *digital signatures* was introduced. For a clearer understanding, it is good to think of a digital signature as a signature that we use to sign checks or other legal documents. A digital signature is created by first generating a message digest from the source message using a hash function (to be explained in the next paragraph). Then, the message digest is encrypted with the sender's private key. This encrypted message digest of the original message is called the digital signature. The generation process of a digital signature is depicted in Figure 4.

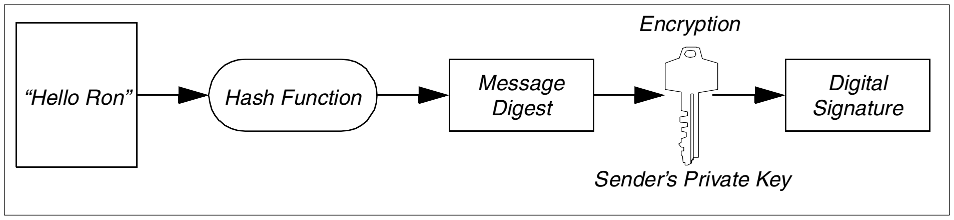


Figure 4. Generation of a digital signature

Before explaining how a digital signature improves user authentication and data integrity in a communication, we need to understand what a hash function is. A *hash function* is not an encryption mechanism, but a tool for creating a digest of a message. A hash function should have three main characteristics:

- It takes a message of any size and generates a small, fixed-size block of data from it (called a *message digest* or *hash value*). Re-executing the hash function on the same source message will always yield the same resulting digest.
- It is not predictable in operation. That means, a small change in the source message will have an unpredictably large effect on the final digest.
- It is irreversible. In other words, there is no way to derive the source message from the digest.

Therefore, the message digest calculated by the hash function can be thought of a good fingerprint of the original message.

Here is how the whole thing works together. During a communication, the digital signature is appended to the original message and sent to the recipient. When the recipient gets the whole message, it will break it into the message and the digital signature. How can the recipient be sure that the message received is really original, that is, the message was not altered and was sent by the real sender? The recipient can do the following steps to check:

1. For the message part, create a message digest using the same hash function.
2. For the digital signature, decrypt it with the sender's public key, which results in the message digest.
3. Compare the two message digests obtained in steps 1 and 2.
4. If the two message digests are the same, this ensures the message was not altered and that it is from the real sender.

Therefore, by appending a digital signature to a message, this assures the recipient that the sender is really the person who created the message and the message has not been changed along the communication path, that is, user authentication is observed and data integrity is preserved.

However, the above scenario is just an ideal situation. Actually, an assumption has been made. In step 2, the sender's public key is used for decryption. But can this public key be trusted? This key may not be from the real sender. Someone can cheat by first altering the message, and then providing a faked public key. The two message digests generated in step 1 and 2 may still be the same, but the message and the sender cannot be trusted in this case. More secure design and trust hierarchy have to be in place and adopted to overcome this. This brings us to the next topic: certificates.

1.3.3 Certificates

In the past few sections, it was shown how public key cryptography overcomes the problem of having to pass a secret from a sender to a recipient. Now, there is still a need to send a key, the public key, to the public. As pointed out in the last section, how can we be sure that the public key really came from whom we think it came from? The mechanism for doing this authentication is by using *certificates*. Certificates, also called *digital certificates*, act very much like passports; they provide a means of identifying individuals. Unlike passports, digital certificates can also be used to identify non-human entities, such as server systems or applications. Another difference between a digital certificate and a passport is that a certificate can (and should) be distributed and copied without restriction, while people are normally very concerned about handing their passports to someone else. Certificates do not normally contain any confidential information and their free distribution does not create a security risk.

More technically speaking, certificates are digital documents that associate an individual or *End-Entity* (EE) with its specific public key. A certificate is a data structure containing a public key, pertinent details about the key owner, and, optionally, some other information, all digitally signed by a trusted third party, usually called a Certificate Authority (CA), see Figure 5.

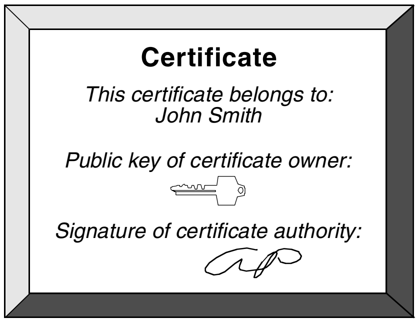

Certificate
This certificate belongs to:
John Smith
Public key of certificate owner:
Signature of certificate authority:

Figure 5. A certificate

The important fact to know and understand about digital certificates is that a *trusted party* (therefore, also called an *authority*) certifies that the enclosed public key belongs to the entity listed in the certificate. The technical implementation is such that it is considered impossible to alter any part of a certificate without easy detectability.

When a sender wants to send a message to a recipient, he or she does not attach the public key to the message, but the certificate instead. (Optionally, the certificate can be published to and later retrieved by the recipient from a public place, such as a public electronic directory.) The recipient receives the message with the certificate and then checks the signature of the third party on the certificate. If the signature was signed by a certifier that he or she trusts, the recipient can safely accept that the public key contained in the certificate is really from the sender. This prevents someone from using a fraudulent public key from impersonating the public key owner.

The owner (entity) that is listed in the certificate and associated with a public key can be identified in various ways. In a simple form, the owner is only identified (represented) by his or her e-mail address. Such certificates are common for secure e-mail communication, but they are not sufficient to represent an individual for doing electronic business, especially when large liabilities or money amounts are involved. In such cases, a Certificate Authority may elect to do a thorough validation of an applicant before signing and issuing a digital certificate.

A digital certificate also normally includes additional information that describes or limits the scope of use.

More details about digital certificates is provided in 2.1.5, “X.509 certificates” on page 25 and subsequent chapters.

1.3.4 Administration

The idea of using certificates seems very convenient. If users or organizations (or simply *end-entities*) decide to use certificates during their communications with other parties, how can they proceed? As previously mentioned, a publicly trusted Certificate Authority (CA) is needed at least. But what functions will this CA perform? How can the end-entities talk with the CA? Is CA alone enough, or are there any supporting authorities and structures for the CA?

Before a CA can sign and issue certificates for others, it has to do the same thing to itself so that its identity can be represented by its own certificate. That means a CA has to do the following:

1. The CA randomly generates its own key pair.
2. The CA protects its private key.
3. The CA creates its own certificate.
4. The CA signs its certificate with its private key.

If an End-Entity needs to communicate with other end-entities, it surely needs a certificate signed by a CA. The End-Entity has to enroll to the CA by filling in some registration forms specifying his or her own information. The information submitted will be used by the CA to identify if this End-Entity is real and should be granted a certificate. Also, the End-Entity (or sometimes the CA) will generate a key pair and send the public key to the CA. The CA will put this public key into the certificate if this End-Entity is eligible to get the certificate. This certificate, after approval and issuance, will be passed back to the requesting End-Entity. So, one basic function of a CA is to create and issue certificates to end-entities. Some organizations may set up a subordinate server for the CA, called a *Registration Authority* (RA), to handle some of the CA’s tasks, for example, processing certificate requests and authenticating users.

Certificates are usually not valid forever. For example, a bank may host a CA issuing certificates to its customers. If a customer closes all his or her accounts with that bank, the certificate of this customer must be revoked. Therefore, another important function of the CA is certificate revocation. The revoked certificates are listed in a *Certificate Revocation List* (CRL), which can be accessed by end-entities to check the certificate status of other end-entities.

For safety reasons, a CA usually issues certificates that are valid for a certain period of time. So, if an End-Entity needs to continue using the certificate, it has to renew its certificate. This policy helps to ensure that all the certificates are in use, and if a certificate expires, and nobody requests a renewal, the certificate will become useless. Thus, the CA must provide a certificate renewal function for the end-entities.

After all these discussions, one may think how can another End-Entity verify the identity of the subject End-Entity? This can be accomplished if the CA publishes the certificates and CRLs to a directory immediately after a certificate is issued or revoked. End-entities can refer to this directory to see the updated changes. This directory can be managed individually by setting it up as a storage space connecting to the CA, or, more appropriately, is embedded in a widely accessible, public directory. This is called a *Certificate Repository* (CR).

To summarize, the CA is an important party who is trusted and can sign and issue certificates for end-entities. Some of its tasks can also be performed by a subordinate function, the RA. The updated certificates and CRLs are kept in a CR for end-entities to refer to. These are the basic elements and procedures required for certificate-based communications.

1.4 Public key and the supporting infrastructure

The explosive growth of the Internet has attracted many companies and organizations to adopt Internet technologies as an alternative to the traditional methods of communication. The Internet is full of potential, but it also exploits the business to security risks. A strong and total security solution is essential. The conventional user ID/password-based systems are inadequate. Certificates and public key cryptography seem to be good candidates for supporting strong security. But how are they going to be used? What will the security system look like? How can organizations plan and implement such an infrastructure? And, how can existing and new applications integrate with this security system?

1.4.1 The secure environment

A strong, secure infrastructure incorporating certificates and public key cryptography may be a solution to the security needs of most organizations. A Public Key Infrastructure (PKI) is a security infrastructure based on certificates and public key cryptography as its security components. It has a set of security services that enable the use and management of public key cryptography and certificates. As the word *infrastructure* suggests, a PKI serves to be a basic security framework of a system or organization with business applications running on top of it. The term PKI is generally used to describe the mechanisms, entities, policies, and relationships that are employed to retrieve cryptographic keys and to associate public cryptographic keys with their owners. It consists of the programs, data formats, communication protocols, institutional policies, and procedures required for enterprise use of public key cryptography.

With the introduction of a PKI, a simplified business structure is shown below in Figure 6.

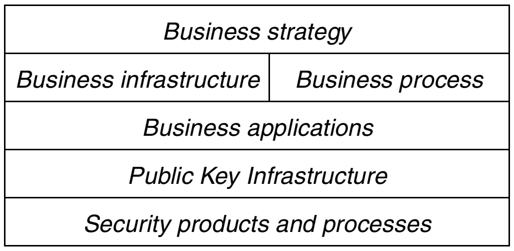


Figure 6. Public Key Infrastructure in business

A PKI can be integrated with all major operating systems, network operating systems, and key applications. If applications want to use PKI for user authentication or encryption, they may need to be PKI-enabled, that is, applications may have to be adapted or new applications have to be written to use the services provided by the infrastructure.

PKI is not simply a system or a product, it really is an infrastructure. Therefore, organizations may have to invest in some basic design initially. However, a PKI brings in a number of benefits. It offers organizations opportunities for automated services that are more secure and economical than in the past. With a PKI, not only the security objectives are met, business applications can also gain benefits. For example, financial institutes will face less frauds due to better authentication of users, shops may have greater sales due to increased customer satisfaction, companies will save money since they no longer have to maintain the expensive private networks for communicating with their business partners or branches, and so on.

1.4.2 Business processes

If an organization determines to proceed with a PKI, what should it do? This process is called *deployment* of a PKI. Some issues concerning how to deploy a PKI in an organization are discussed in this section. Further suggestions and discussions are in Chapter 4, “PKI deployment” on page 59.

Usually, the deployment of a PKI is driven by some business or security need. Organizations have to be clear about their business requirements and how technology can suit their needs. The business needs may be due to a corporate e-business strategy, a country’s legal requirements, or a particular application’s security needs. Organizations should evaluate if and how the current PKI technology and techniques can align with their business needs.

Another impact brought by a PKI deployment is the change to business processes. Organizations have to plan carefully beforehand to see if new business processes have to be defined, old processes have to be re-engineered, resources have to be reallocated and/or added, training has to be provided, what applications have to be modified or added, what the migration schedule is like and whether to take a phase approach, and so on. All these have to be considered thoroughly and carefully. Since the whole organization might be affected, consensus and support from all involved parties should be gained. The budget for the PKI deployment should also be planned and acquired.

The legal aspect is another major issue that must be considered. This is a determinative factor on how the organization can move toward using PKI. What legislation is in place or pending that may impact an organization’s ability to do business electronically? What is the legal relationship between the organization and the end-entities? Can the product be used internationally, or just domestically? What liability will the organization bear?

From a technology point of view, the organization has to consider the following: I/T architecture, standards compliance, components to offer, application design, system management, network design, choice of vendor and solutions, prototyping, product customization, application development, system integration, testing, system rollout, and so forth.

PKI deployment is important and needs detailed planning. Security consulting expertise may help to guide the decisions and provide solutions and consultancy services. Organizations need to plan from these aspects: business needs, impact to business processes, legal issues, and technology.

1.5 Current use of Public Key Infrastructures

Currently, a few organizations have started their plan on implementing a PKI. Although PKI is new to corporate customers, it has been deployed substantially in many well-known network protocols by embedding the cryptographic algorithms (both public key and/or secret key). Figure 7 illustrated some of these network protocols.

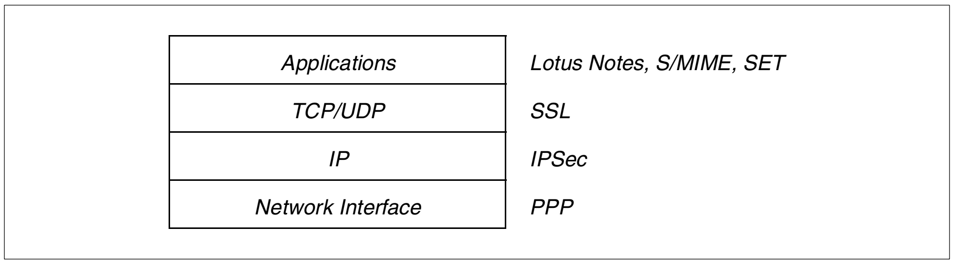


Figure 7. Network protocols using cryptography

Lotus Notes and S/MIME are secure mail applications using a combination of public key and secret key cryptography. They also provide digital signature capability.

Lotus Notes is an enterprise mail solution developed by Lotus Development Corporation. Lotus was one of the first companies to recognize the value of a PKI, and it is one of the few to have implemented it on a large scale. Currently, Lotus Notes and Lotus Domino use a proprietary PKI based on RSA technology to support user and application security. In Lotus Domino Server 5.0 and Lotus Notes 5.0, some Internet PKI standards are adopted to allow interfaces with other vendors' PKIs. In Version 5.0, the Internet certificates can be used for S/MIME and SSL in the Notes client. For more information about the products, please refer to the following Web site:

`http://www.lotus.com`

S/MIME is the short form of Secure Multipurpose Internet Mail Extensions and was developed by RSA Data Security, Inc. It is an emerging standard for interoperable secure e-mail, by adding digital signatures and encryption to Internet MIME messages. For more information, please refer to:

`http://www.rsa.com/smime`

The *Secure Electronic Transaction* (SET) specification is an open technical standard for the commerce industry as a way to facilitate secure credit card transactions over the Internet. It was jointly developed by Visa, MasterCard, IBM, and other technology companies. The core protocol of SET is based on digital certificates issuance. For more information about SET, please refer to the following:

```
http://www.setco.org
http://www.ibm.com/software/commerce/payment
```

The *Secure Sockets Layer* (SSL) protocol was developed by Netscape Communications Corp. to provide security and privacy between clients and servers over the Internet. The SSL protocol is application independent, and thus, it allows protocols, such as Hypertext Transfer Protocol (HTTP), file transfer protocol (FTP), and Telnet to run on top of it transparently. SSL uses RSA public key cryptography and is capable for client authentication, server authentication and encrypted SSL connection. For more information, please refer to the following:

```
http://developer.netscape.com/docs/manuals/security.html
```

IPSec is a set of specifications for authentication, integrity, and confidentiality services based on cryptography at the IP datagram layer. This standard can be used for building secure host-to-host pipes, encapsulated tunnels, and virtual private networks (VPNs) over the Internet. For more details, please see:

```
http://www.ietf.org/html.charters/ipsec-charter.html
```

Point-to-point protocol (PPP) is used on Internet Service Provider (ISP) dial-up connections. It is an Internet standard for transmission of IP packets over serial lines from point to point. There are two protocols for authentication: the Password Authentication Protocol (PAP) and the Challenge Handshake Authentication Protocol (CHAP), in which CHAP uses encryption. For more details about PPP, please see:

```
http://www.ietf.org/html.charters/pppext-charter.html
```

Chapter 2. PKI overview

Today, the Internet is an established networking infrastructure and nobody doubts that a number of new e-business solutions will appear on the Internet cyberspace in the near future. But, naturally, the Internet is an un-secure medium compared to private networks. On the Internet, data transmissions might be eavesdropped by someone with sniffing tools, and the communication partner might be fake. As described in Chapter 1, “Introduction to security systems” on page 3, public key cryptography is suitable for resolving most of these issues. A PKI offers the base of practical usage of public key cryptography. Originally, PKI was a generic term that meant a set of services that make use of public key cryptography. As previously described, PKI has been exploited in many applications or protocols, such as Secure Sockets Layer (SSL), Secure Multimedia Internet Mail Extensions (S/MIME), IP Security (IPSec), Secure Electronic Transactions (SET), and Pretty Good Privacy (PGP). On the other hand, X.509 V3 digital certificate (to be explained later) exploitation within PKI has been one of the most desired standardization issues in e-commerce. Since 1995, the Internet Engineering Task Force (IETF) PKIX working group started to fully involve X.509 V3 certificates into the PKI standards and make PKI worthy of practical use for critical business on the Internet. The IETF PKIX working group standard is generally considered to be most widely accepted.

2.1 The core PKI components

Figure 8 on page 22 is an illustration of a PKI structure. The core components of a PKI include:

- The End-Entities (EE)
- The Certificate Authority (CA)
- The Certificate Repository (CR)
- The Registration Authority (RA)
- Digital Certificates (X.509 V3)

We discuss the above PKI core components in the sections that follow.

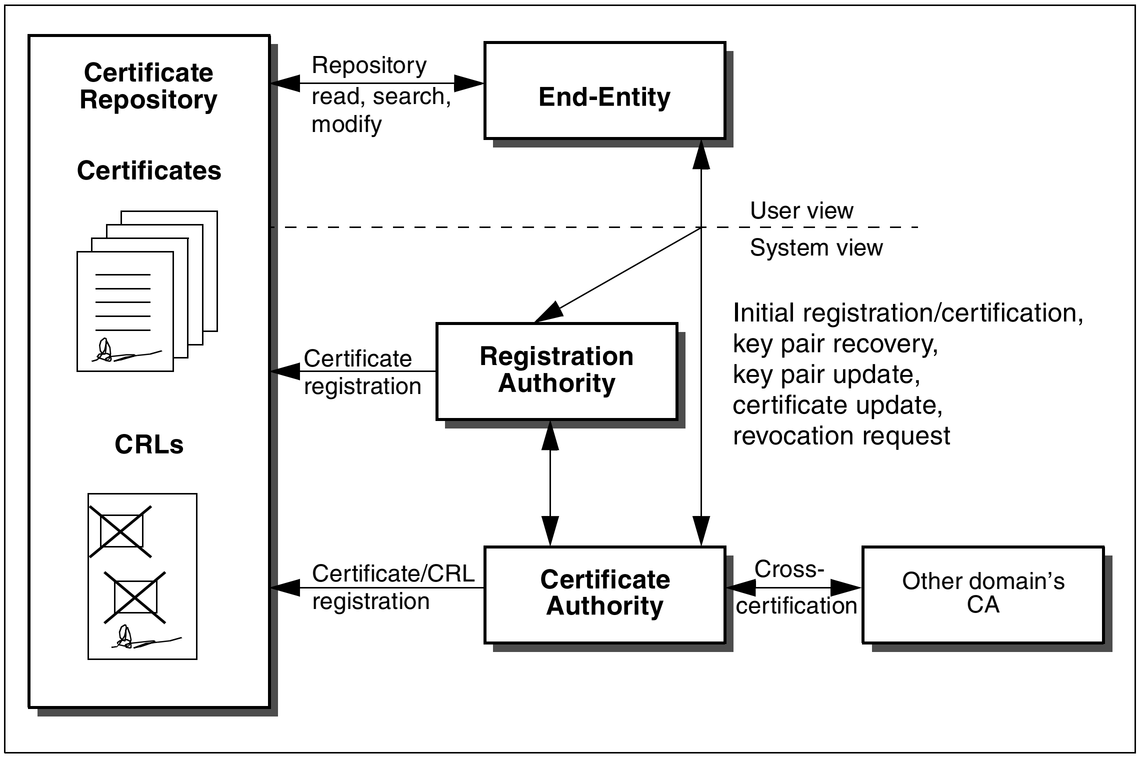


Figure 8. General PKI structure

2.1.1 End-Entity (EE)

An *End-Entity* is defined as a user of PKI certificates and/or end-user system that is the subject of a certificate. In other words, in a PKI system, End-Entity is a generic term for a subject that uses some services or functions of the PKI system, which may be a certificate owner (human being or organization or some other entities), or a requestor (it might be application program) for certificate or CRL.

2.1.2 Certificate Authority (CA)

The Certificate Authority (CA) is the signer of the certificates. The CA, often together with the RA, which is described in 2.1.4, “The Registration Authority (RA)” on page 25, has the responsibility of the certificate subject entity's identification. The logical domain in which a CA issues and manages certificates is called *security domain,* which might be implemented to cover an organization, company, a large department, a test cell, or another logical community in real cases. A CAs primary operations include certificate issuance, certificate renewal, and certificate revocation.

2.1.2.1 Certificate issuance

A CA creates a digital certificate by signing it with its digital signature. Basically, a public and private key pair is generated by a requesting client (EE). The client then submits a request for certificate issuance to the CA. The certificate request contains at least the client's public key and some other information, such as the client's name, e-mail address, mail address, or other pertinent information. When an RA (see 2.1.4, "The Registration Authority (RA)" on page 25) is established, the CA delegates the client verification process or other management functions to the RA. After the client request is verified, the CA creates the digital certificate and signs it. As an alternative to the above, a CA can generate a client's key pair and, subsequently, the signed certificate for clients. This process, however, is seldom implemented because the private key needs to be forwarded from the CA to the client, which can be a weak link. It is generally considered more secure when the clients generate their key pairs, in which case, the private keys never leave their area of authority.

In order for a PKI system to work completely, the basic assumption is that any party who wishes to verify a certificate must trust its digital signer CA. In PKI, "A trusts B" means that "A trusts the CA that signed B's certificate". In general, "A trusts CA" means that "A" holds a copy of the CA's certificate locally. In case of secure HTTP using SSL, for example, most commonly used Web browsers have a list of several trustworthy CA certificates already incorporated when they ship, such as for VeriSign, Entrust, IBM World Registry, and others. If a Web server uses a certificate that is signed by such a trusted CA, clients are automatically considered to trust the server, unless the client intensionally deletes the signer CA certificate.

A CA can issue several kinds of certificates, including:

User certificates	User certificates may be issued to an ordinary human being or other entities, such as servers and applications. They are trusted end-entities for the CA. An RA should also have this certificate. A user certificate may be limited to e-mail, servers, or for other specific purposes.
CA certificates	When a CA issues a certificate for itself, it is called a *self-signed certificate*, or *root certificate* for that CA. If a CA issues a certificate for a subordinate CA, the certificate is also called a CA certificate.
Cross certificate	Cross certificates are used for *cross-certification*, which is an authentication process across security domains.

2.1.2.2 Certificate renewal

Every certificate has a validity period with an expiration date associated with it. When a certificate expires, a renewal process may be initiated and, upon positive approval, a new certificate will be issued to the End-Entity.

2.1.2.3 Certificate revocation

The maximum lifetime of a certificate is its expiration date. In some cases, however, a certificates needs to be revoked before its expiration date. When this happens, the CA posts the certificate to a Certificate Revocation List (CRL). (To be more precise, the CA posts the certificate's serial number, along with some other information, to the CRL.) Clients that need to know the validity of a certificate can search the CRL for any revocation notice.

2.1.3 The Certificate Repository (CR)

The CR (Certificate Repository) is a store of issued certificates and revoked certificates in a CRL. Although a CR is not a required component in a PKI system, it significantly contributes to the availability and manageability of a PKI system.

Because the X.509 certificate format is a natural fit to an X.500 directory, a CR is best implemented as a *directory* and it is then able to be accessed by the dominant Directory Access Protocol, the *Lightweight Directory Access Protocol* (LDAP). RFC 2587, *Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2*, defines the access method to a repository with which an End-Entity or a CA can retrieve or modify the certificate and CRL information stored in a CR. Thus, a CR can be accessed with LDAP commands or procedures, such as *bind*, *search* or *modify*, and *unbind*. In RFC 2559, *Internet X.509 Public Key Infrastructure LDAPv2 Schema*, the attributes and object classes to be supported by an LDAP server acting as server of a CR are defined. The latest LDAP is Version 3, and the new Internet Draft based on LDAP V3 is in the process of being finalized at the time of writing. With the new LDAP V3 protocol, we can expect some improvements, such as UTF-8 exploitation, flexible security mechanism, or better scalability of a CR server.

Although not recommended, there are alternative methods to obtain certificate or CRL information if a CR is not implemented in a directory. However, after considering the requirements that a CR must meet, it turns out that a directory is actually the best place to store CR information. Such requirements include easy accessibility, standards-based access, up-to-date information storage, built-in security (if required), data management issues, and possible merge with similar data.

A CR makes a CRL distribution system design quite simple. Furthermore, the simplicity and flexibility of LDAP makes a CR more applicable for many purposes.

2.1.4 The Registration Authority (RA)

The Registration Authority (RA) is an optional component in a PKI. In some cases, the CA incorporates the role of an RA. Where a separate RA is used, the RA is a trusted End-Entity certified by the CA, acting as a subordinate server of the CA. The CA can delegate some of its management functions to the RA. For example, the RA may perform personal authentication tasks, report revoked certificates, generate keys, or archive key pairs. The RA, however, does not issue certificates or CRLs.

2.1.5 X.509 certificates

X.509 is the most widely used certificate format for PKI, being used in major PKI-enabled protocols and applications, such as SSL, IPSec, S/MIME,Privacy Enhanced Mail (PEM), or SET. A rare example of one that does not support X.509 certificate format is Pretty Good Privacy (PGP), which uses its own certificate format. Figure 9 on page 26 is a basic structure of an X.509 certificate. Initially, X.509 V1 appeared in 1988 as ITU-T definition. X.509 V2 supports new fields over Version 1; they are *issuer* and *subject identifier*. The latest X.509 V3 was defined in 1996, which introduced the *extension* field. Currently, many PKI deployment applications are based on X.509 V1 or V2, but the PKI technology direction trends clearly to X.509 V3 based implementation.

Certificate data is written in *abstract syntax notation 1* (ASN.1) syntax rule as can be seen in Figure 10 on page 27. It is then converted into binary data along with ASN.1 *distinguished encoding rules* (DER). ASN.1 is a data description language and defined as X.208 and X.209 standard by the ITU-T. This operation enables certificate data independent from each platform's encoding rule.

In some fields of a certificate, an *object identifier* (OID) is used to represent the specific series of parameter values. For example, in Figure 10 on page 27, you can see the *AlgorithmIdentifier* for *signatureAlgorithm*, which actually consists of an object identifier (OID) and optional parameters. This OID represents a specific algorithm used for digital signatures of the certificate issuer (CA). The application that verifies the certificate's signature has to understand the OID that represents the encryption algorithm and message digest algorithm, along with other information.

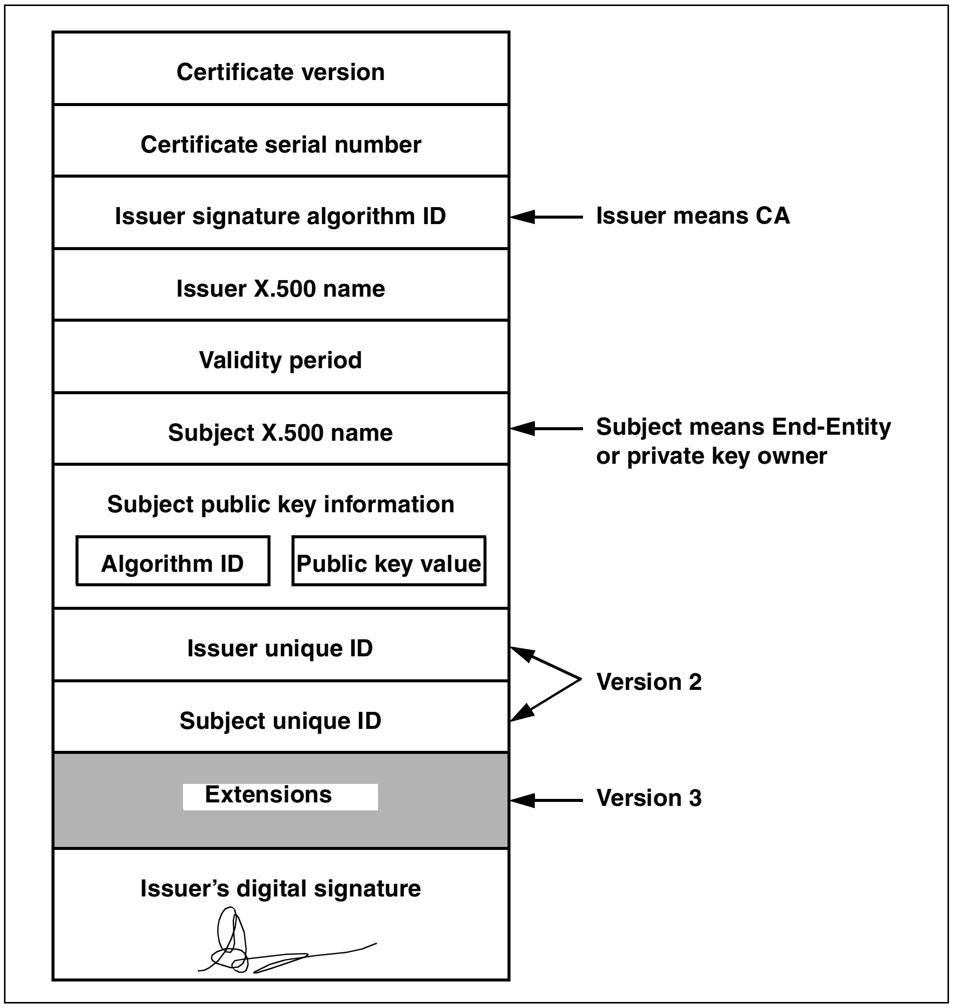


Figure 9. X.509 certificate format

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
     extensions      [3]  EXPLICIT Extensions OPTIONAL
     }
```

Figure 10. Abstract syntax notation 1 (ASN.1) representation of a certificate

2.1.5.1 Extensions

Extensions are new fields that were introduced in X.509 V3 and they add flexibility to certificates. One problem, however, is that an extension does not always have a clear definition of its usage; the exploitation of extension fields is, therefore, to a certain degree, subject to the freedom of the implementor. The IETF PKIX working group aims to fix this situation and to make X.509 V3 certificates a more clearly-defined standard.

In RFC 2459, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, several extensions' usages are defined and they are called *standard extensions*. Standard extensions are minimum requirements for PKIX standard conforming applications. Such applications still have the freedom to implement some proprietary extensions other than standard extensions. IBM Trust Authority 3.1, which is explained later in this book, uses three extension categories: *standard extensions* (PKIX-conforming standard extension), *common extensions* (extensions that are unique to Trust Authority) and *private extensions* (extensions available for each application's private use).

Each extension field contains a boolean field that marks it as *critical* or not. If an extension is marked as critical, a certificate using the system must be able to recognize the field; otherwise, it must reject the certificate. The following extensions should be critical or must be critical:

Key Usage (KeyUsage)
Key usage defines the purpose of the key contained in the certificate, such as digital signature, key encipher, data encipher, and so on. This field should be critical.

Subject alternative name (subjectAltName)
Subject alternative name defines additional identities of a certificate private key owner (subject of certificate). subjectAltName might be an e-mail address, DNS name, IP address, or other defined name that can be used to specify the certificate owner. If this field is provided, a CA (or RA) must do verification of it in the certificate issue process. This field must be critical.

Basic constraints (BasicConstraints)
Basic constraints is used for a CA certificate and is not used in an End-Entity certificate. It relates to *certification paths* and designates the subordinate CA certificate number to End-Entity certificate. If the pathLenConstraint value in BasicConstraints is 0, the CA private key can be used only for End-Entity certificates. If the number is 1, the CA has a maximum of 1 subordinate CA between itself and an End-Entity in the certification paths. This field must be critical.

Name constraints (NameConstraints)
Name constraints are also used for a CA certificate and designates which naming patterns are acceptable in its subordinate name space. This constraint is applicable to the certificate subject name, that is, the subject's distinguished name (DN) and subject alternative name in the certificate extension. For example, a CA with NameConstraints as ou=abc,o=ibm,c=us will issue certificates for End-Entities whose subject DN is cn=mike smith, ou=abc,o=ibm,c=us but will not be able to issue certificates for another End-Entity whose DN is cn=ken brown,ou=xyz,o=ibm,c=us. This field must be critical.

The above standard extension fields are considered especially important in the PKIX standard. Besides the above fields, other standard extensions are defined: Authority Key Identifier (authorityKeyIdentifier), Subject Key Identifier (keyIdentifier), Private Key Usage Period (PrivateKeyUsagePeriod), Certificate Policies (certificatePolicies), Policy Mappings (PolicyMappings), Issuer Alternative Names (IssuerAltName), Subject Directory Attributes (SubjectDirectoryAttributes), Policy Constraints (PolicyConstraints), Extended Key Usage Field (extKeyUsage), and CRL Distribution Points (cRLDistributionPoints). These are described in RFC 2459, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*.

2.1.5.2 Certificate classes

The certificate class is a generally acceptable idea to distinguish each certificate's level of trust. VeriSign's certificate classification is a famous example, and currently, VeriSign has four levels of certificate classes, class1 through class4. The major difference between each certificate class is its registration process (and, thus, the level of trust). A class1 certificate is limited to an individual only. For business entities, such as a Web server certificate, at least a class3 certificate must be required. The usage rules are defined in VeriSign's Certificate Practice Statement (CPS). You can access VeriSign's CPS as follows:

```
http://www.verisign.com/repository/CPS
```

The class1 certificate registration process only requires a user name and an e-mail address to identify a user. Class1 certificates can be used to secure e-mail messages. As the number of certificate classes increases, the registration process becomes more strict.

Most implementations incorporate different certificate classes for the different usages of certificates (see also 2.3.6, "Policies and processes" on page 38).

2.2 The supporting components

In this section, we discuss the supporting components of a PKI. A Certification Revocation List (CRL) offers a mechanism that makes the PKI system more reliable and trustworthy. The defined protocols for PKI, which are discussed later in this section, offer online processing capability to the PKI system and encourages PKI deployment on the Internet. Auditing is a necessary feature that keeps or improves the PKI system's trustworthiness.

2.2.1 Certificate Revocation Lists (CRLs)

A CRL is a means of notifying clients who wish to verify the revocation of certificates. A CRL is a critical factor in a PKI deployment. CRLs are issued to mark some certificates unusable, even though their expiration has not come yet.

The circumstances that may lead to certificate revocation include:

- When a certificate owner was removed from the security domain, for example, due to retirement or a move to another organization that resides in another security domain.
- When a certificate owner recognizes that his or her private key (file or Smartcard) has been lost or might have been compromised.

- When the certificate owner has forgotten his or her private key's password or PIN.
- When some information contained in the certificate has become incorrect, which led to a re-issuance of a new certificate.
- When security attacks are suspected with a particular certificate or associated private key.

In PKIX, the X.509 V2 CRL format is accepted and proposed. The format is illustrated in Figure 11.

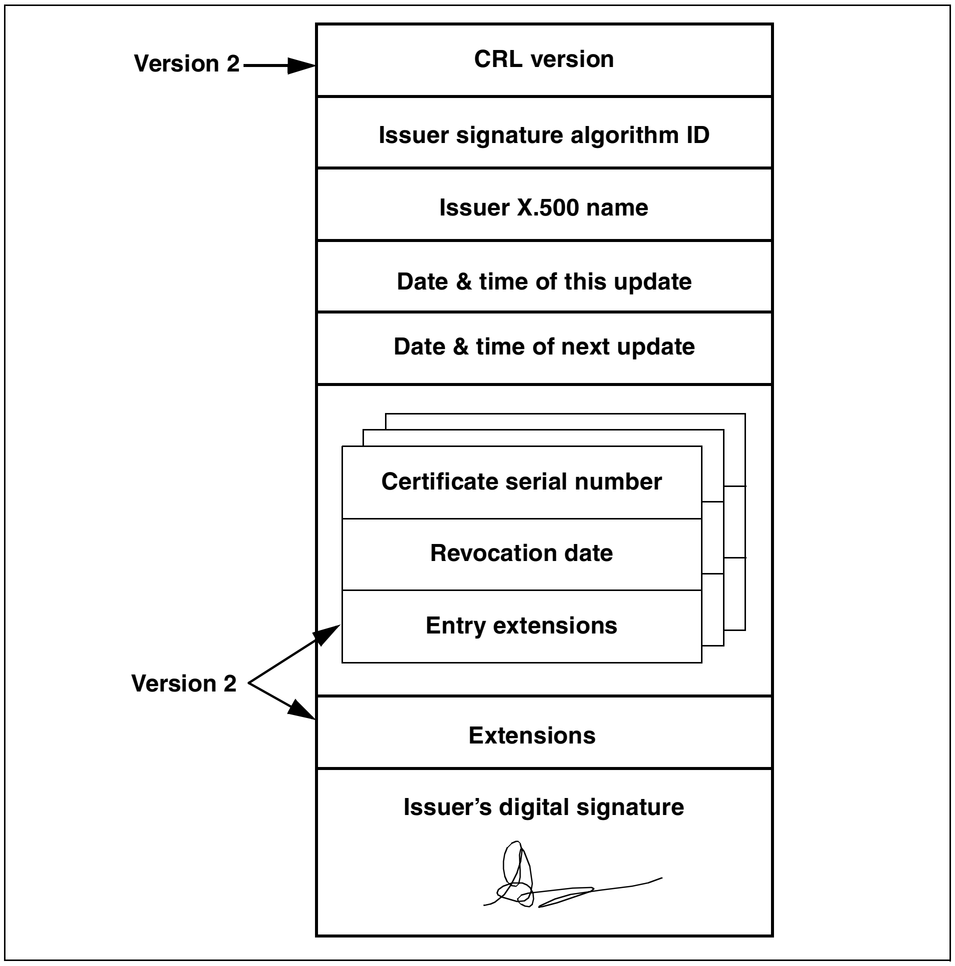

Version 2
CRL version
Issuer signature algorithm ID
Issuer X.500 name
Date & time of this update
Date & time of next update
Certificate serial number
Revocation date
Entry extensions
Version 2
Extensions
Issuer's digital signature

Figure 11. X.509 CRL format

A CRL includes the serial numbers of revoked certificates, time stamps, the CA signature, and some extensions. CRLs may be issued periodically or whenever an update needs to be posted. Any client can verify certificates by checking the existence of a certificate's serial number in a CRL. If the serial number exists, the certificate has been revoked.

There are several methods for maintaining and publishing a CRL. A CRL could be available using HTTP from a Web site. The recommended implementation in a fully-deployed PKI environment, however, is to store the CRL in an LDAP directory, as described in 2.1.3, "The Certificate Repository (CR)" on page 24. The PKIX standards allow a CRL distribution point to be named in a certificate extension field named *CRL Distribution Points* (cRLDistributionPoints) extension. Applications can then get the CRL location information, such as a URL, from the certificate to access the CRL. As more and more applications access the CR to obtain the latest CRL, a performance issue for the CR, and eventually for the application, may arise. CRL freshness and CR performance are often mutually exclusive. Several options are available for reasonable implementation, including:

- Caching a CRL in a local cache, which is feasible when the CRL latency is acceptable.
- Rather than one long CRL, keep multiple shorter CRLs available.
- Distribute the CRL to multiple places and spread the load using the certificate extension field cRLDistributionPoints. This, of course, causes some additional management overhead.
- Use a sufficiently scalable and powerful CR server.

If a CRL is being used by applications for certificate validation, provisions must be in place for adequate availability of the CRL service (or applications should incorporate some backup procedures in case the CRL service is unavailable).

2.2.2 Protocols

In the beginning, network protocols that support PKI have not been considered too critical. Later, in order to exploit PKI sufficiently over a network infrastructure, some network protocols become desirable for transactional communication in a PKI system, such as for management use or for certificate and CRL retrieving operations.

2.2.2.1 Administration protocols

In regard to PKI deployment, a common administrative protocol is thought to be very convenient. PKIX defines the *Certificate Management Protocols* (CMP) in RFC 2510, *Internet X.509 Public Key Infrastructure Certificate Management Protocols*. CMP enables online base interactions in a PKI. CMP base interaction occurs between End-Entities and a CA (or RA), or between a

CA and RA (in fact, an RA is considered an End-Entity), or between a CA and another security domain's CA for cross-certification. Although CMP can be used with HTTP, FTP, or e-mail, the most common use is over TCP/IP, using the default port 829. CMP is a message-based protocol, and its message format is defined as *Certificate Request Message Format* (CRMF) in RFC 2511, *Internet X.509 Certificate Request Message Format*. The management task accomplished with CMP includes the following (please also refer to Figure 8 on page 22):

Initial registration and certification
This is the process where an End-Entity makes itself known to a CA or RA, prior to the CA issuing a certificate or certificates for that End-Entity. As a result of this process (when it is successful), a CA issues a certificate for an End-Entity's public key and returns the certificate to the End-Entity and/or stores the certificate in a publicly available CR.

Key pair recovery
As an option, a client's private key may be backed up by a CA, an RA, or a key backup system associated with a CA or RA. If an entity needs to recover such a backed-up key (for example, as a result of a forgotten password or a lost key chain file), this protocol can be used for recovery.

Key pair update
Ideally, a key pair should be used as long as possible. In some cases, a key pair needs to be updated (this means the old key pair is replaced with a new key pair, and a new certificate needs to be issued). For example, this operation is required when the private key has been compromised. Another example is that as technology evolves, the key length or its algorithm becomes insufficient to protect the certificate's trustworthiness.

Certificate update (renewal)
As certificates expire, they need to be refreshed if nothing relevant in the environment has changed.

Revocation request
An authorized person advises a CA of an abnormal situation requiring certificate revocation (see 2.2.1, "Certificate Revocation Lists (CRLs)" on page 29).

Cross-certification
A CA requests the issuance of a cross-certificate from another CA that resides in another security domain.

2.2.2.2 Operational protocols

An End-Entity (including an RA) and a CA can use *operational protocols* to retrieve PKI information, such as X.509 certificates and a CRL stored in a CR. Because in PKIX the CR is implemented as an LDAP directory, the operational protocols are implemented on the LDAP protocol basis. This standard is defined in RFC 2559, *Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2* and is also closely related to another standard defined in RFC 2587, *Internet X.509 Public Key Infrastructure LDAPv2 Schema.* Operational protocols involve the following three operations:

Repository read

This operation is used for the retrieval of an entity's information from a repository. The process in general is done in a `BindRequest` -> `BindResponse` -> `SearchRequest` -> `SearchResponse` -> `UnbindRequest` sequence.

Repository search

This operation is similar to the repository read operation. Additionally, this operation includes sub-tree scope searching and enhanced filtering capability.

Repository modify

In LDAP terminology, modify means modify, add, and delete. If an End-Entity or CA needs to perform any of these tasks, they will use this operation. This process will be done as a `BindRequest` -> `BindResponse` -> `ModifyRequest` -> `ModifyResponse` or `AddRequest` -> `AddResponse` or `DelRequest` -> `DelResponse` -> `UnbindRequest` sequence.

Since LDAP is still an emerging technology, in some cases, the repository might be not a directory but a Web site. For that reason, PKIX defines FTP and HTTP base access protocol in RFC 2585, *Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP.* An End-Entity can retrieve the (non-directory) repository location from the Universal Resource Identifier (URI), which is stored in the certificate or CRL extension field. Such a URI may look like these two examples:

```
ftp://ftp.server.name/pki/abc.cer
http://www.server.name/pki/abc.crl
```

2.2.2.3 Online Certificate Status Protocol

The *Online Certificate Status Protocol* (OCSP) is an alternative method for obtaining a certificate's current status without requiring CRLs and it can be used to satisfy some of the operational requirements of more timely revocation information than what is possible with CRLs. The OCSP is defined in RFC 2560, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. If using a CRL, the certificate user may have the latest CRL, but in some application requirements, the delta between the current CRL and next issued CRL cannot be accepted. An *OCSP responder* exists between a certificate user (OCSP client) and a CR, which should have the current CRL and any delta CRL information. The delta CRL information is an exploitation of a CRL V2 extension field, deltaCRLIndicator. OCSP clients request certificate revocation status from the OCSP responder.

2.2.3 Auditing

A PKI relies on the trustworthiness of the CA. When a PKI system is being designed in an organization or company, special care must be taken in order to protect the CA system. Furthermore, the CAs private key is one of the most critical parts in a PKI system. Hacker attacks must be prevented and internal threats, such as secret leaks, which may be more serious than a hacker attack, must be taken care of.

In addition to the basic security precautions, a thorough auditing process allows for the tracking of any suspicious actions after they have occurred. Legal aspects may also dictate a process that logs any PKI-related activity. An auditing subsystem in a PKI logs all transactional and other activities with an accurate timestamp and any relevant data. All audit trails are ideally secured (signed) and securely stored in such a way that they cannot easily be modified or removed thereafter.

2.3 The supporting infrastructure

A PKI is one of the most suitable solutions to guarantee secure communication in distributed computing environments, such as the Internet. At the same time, PKI is still emerging, so if we desire the deployment of a PKI-enabled system in an organization, we might have to overcome some obstacles. At the time of writing, some of the IETF PKIX working group standardization efforts are still in progress. Although several core standard Request for Comments (RFCs) have been finalized, there are still others in the state of Internet Drafts.

We can easily imagine a situation in the foreseeable future when digital signatures are used for legal contracts and other documents in daily business. Such contracts may be signed between business partners, private persons, government agencies, or between any of these. Digitally signed documents may also be used across country borders, in which case legal aspects may become even more important and we have to expect some social, governmental, or legal establishment. Furthermore, for the deployment of the seamless PKI system all over the world, we have to be sensitive about export regulation issues. In any case, PKI not only includes public key technology, but must also include a supporting infrastructure that takes care of any operational, legal, possible cultural, and other issues. The following section focuses on some systematic aspects of the infrastructure supporting a PKI.

2.3.1 Directory exploitation

Although most standards (RFCs and Internet Drafts) define internal functions of a PKI only, the importance of a directory-based approach cannot be stressed enough. The PKI standards, however, define how to access such a directory, including directory schema, using the LDAP protocol: RFC 2587 defines the use of LDAP Version 2, and an Internet Draft proposes the use of some enhancements in Version 3 (see Chapter 5, “PKI-related standards” on page 101).

Most PKI-related literature (and sometimes even the RFCs) do not specify how certificates and certificate revocation lists are to be published in practice. In fact, there is not a single solution for this, but the recommended way is to use an LDAP directory. LDAP is a fast evolving standard for large scale, general purpose directories that most organizations are about to exploit for various reasons. A PKI is just one of many applications that benefit largely from an LDAP directory.

Using an LDAP directory, certificates, and Certificate Revocation Lists can be published very easily in such a way that they can be retrieved by all (or most) clients using a relatively simple protocol.

For some general information about LDAP, please read Appendix A, “Introduction to LDAP” on page 211.

2.3.2 The system environment

Two critical issues in a PKI system to guarantee the system trustworthiness must be considered. One is how to securely manage and operate the CA system (with its critical private key) as part of the whole infrastructure. Part of this is a secure auditing system as explained in 2.2.3, “Auditing” on page 34.

Another issue is how to build a reliable process of CRL distribution. As described previously, a Certificate Repository (CR) can offer the most suitable mechanism for it, and furthermore, a LDAP directory is the best choice for storing X.509 certificates. This is the reason why LDAP server deployment is strongly desired. LDAP is being exploited in a number of other areas, such as people directories, distributed system management repositories, network quality of service (QOS) repositories, and so on.

Finally, the planning of a system environment must take into consideration any backup and recovery solutions. Necessary means for availability and scalability must be in place. Special attention must be put on system backup solutions. System backup tapes or backup operations over a network are relatively easy to access if security measures are not in place.

2.3.3 Cryptographic hardware

Cryptographic hardware has two major advantages over software cryptography. The first one is a performance advantage, because specialized hardware can normally perform cryptographic operations much faster (and in parallel to other application operations) than a software solution running on a general purpose processor. Another advantage over software cryptography is that cryptographic hardware increases the overall security by storing critical items away from shared disk or other storage.

In hardware cryptography implementations, like the IBM 4758 PCI Cryptographic Coprocessor, a master key of the cryptographic processor is normally used for encryption of all other user keys. This master key can be stored in a tamper detecting, battery back-up, special register of the processor. All user keys, including the CA’s private key, are encrypted in auxiliary storage or even on processor storage. While this provides for a maximum of security, it bears a risk when the hardware fails for any reason. Should this happen, all keys are lost, unless the master key is stored somewhere outside the processor.

As described in 2.3.2, “The system environment” on page 35, a CA system’s trustworthiness is a critical issue. Cryptographic hardware can be considered to increase the safety of a CA system.

2.3.4 Smartcards

A *Smartcard* (or, more generally, a *security token*) is a device that houses some processing power for cryptographic and other processing and some secure storage area for the storage of keys, certificates, and the like. Smartcards are considered a reliable method for the storage of an End-Entity’s public-private key pairs, trusted CA’s root certificates, or other information. Smartcards have some advantages over simple files on a system’s mass storage media:

- A Smartcard can provide a user some independence from the PC hardware. A private key and associated certificate information are usually stored on a PC's hard disk as a file, which binds the user to that particular PC. It is not normally a trivial operation, such as a file copy operation, to move a private key and its associated certificate to another PC. In an office computing environment, a PC is often a company's asset, and in some cases, one PC can be shared by several people. It is, therefore, desirable to store security-related data on a more personal device, such as a Smartcard. This way, people can carry their private key with them without worry of the key being compromised or copied from a hard drive.

- Because Smartcards have memory and processing logic, some advanced function become possible, such as key-pair generation, CRL caching, repository search request application, logging, and so on.

Some security applications support the concept of *virtual Smartcards*. A virtual Smartcard provides the same functionality as a physical Smartcard, but the functionality is implemented in software and file storage on a user's workstation. Since virtual Smartcards normally support the same standardized software interface (API) that real Smartcards do, the replacement of a virtual Smartcard by a real (physical) Smartcard can be relatively simple.

2.3.5 Applications

Applications' exploitation of PKIX standards is vital for the deployment of a PKI. Applications include high-level (real) applications, such as Lotus Notes groupware, or some low-level security enablers, such as SSL or SET. Some everyday applications, such as the popular Web browsers from Netscape Communications Corp. or Microsoft Corp., already support a PKI to some extent. For example, a Web browser supports client certificate authentication using either built-in certificate storage or external Smartcard support. Popular mail client software supports signing and encrypting e-mail messages through the use of PKI features. It is easy to imagine that many new applications will soon exploit the PKIX standard.

As a matter of fact, by using existing shrink-wrapped software for Web access and e-mail, organizations can make use of a PKI as of today. By using certificate authentication for application clients running in a Web browser and secure e-mail, many (if not most) of today's business processes can already be incorporated into a PKI. What needs to be in place, however, is the infrastructure that issues and manages the server and user certificates. Organizations may choose to outsource this function by having external providers for these duties. Another option is to establish an organization's own PKI with the required basic functions, such as a CA that issues the necessary certificates.

2.3.6 Policies and processes

A PKI must be operated in accordance with defined policies. Deployment of a PKI system in an organization requires the development of security policies and processes for that organization.

The standardization effort has been made to involve security policies into a PKI framework systematically, as outlined in RFC 2527, *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*.

According to X.509, a certificate policy is “a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.” A more detailed description of the practices followed by a CA in issuing and otherwise managing certificates may be contained in a Certification Practice Statement (CPS) published or referenced by a CA.

A certificate policies extension contains a sequence of one or more policy information terms, each of which consists of a registered object ID (OID) and optional qualifiers. Applications with specific policy requirements will have to recognize the OID meaning at least in the same security domain. If the required policy’s OID is not contained in the certificate extension field, or if any existing critical OIDs are not understood by the application, the application has to reject the clients request.

Security policies also result in *processes* that have to be in place and subsequently enforced. Processes describe (and/or mandate) the way an infrastructure is utilized by its administrators and users. Processes may include elements, such as:

- The certificate requesting, issuance, distribution, and revocation processes.
- The use of certificates for client authentication.
- The use of certificates for securing e-mail communication.
- The use of certificates for inter-organization communication.
- Procedures to follow when security violations are suspected.
- Handling guidelines for private keys and certificates.
- Application development guidelines for PKI exploitation (such as user authentication using certificates).

Chapter 3. Principles of operation

Public key cryptography is regarded as an enabling technology. It is a key technology that helps to achieve business objectives in a practical manner by maintaining a high level of security. Businesses want to operate over the Internet in a secure manner. In other words, they want to be sure they know who they are conducting business with and they want to be sure that whoever they are dealing with can be held accountable. The nature of any network is effectively a distributed environment. Users may not be part of the same network. Also, the identity of a user cannot be verified unless special conditions are in place.

Chapter 1, “Introduction to security systems” on page 3, introduced a number of security concepts. There are five fundamental security concepts that a PKI enables: data integrity, data confidentiality, authentication, non-repudiation, and key management. Allowing the components introduced in Chapter 2, “PKI overview” on page 21, to work together will provide a convenient and transparent environment for the implementation of these security concepts. This chapter shows the relationships the above components share and how these relationships can provide a set of useful services known as a *public key infrastructure* (PKI).

Further, a PKI, based on open standards, can provide the following:

- Standardization of networked applications. This provides secure communications over the Internet or any organizational network.
- Support and enablement of electronic commerce, for example, with the use of Secure Electronic Transaction (SET) payment protocols or the Secure Sockets Layer (SSL) security protocol.
- A secure and scaleable security environment.
- Key management and certificate management across multiple applications.
- A security solutions infrastructure that has a very good chance of being recognized and accepted by business partners and even government authorities.

3.1 Data encryption and decryption

The basis of all cryptographic operations involves either an encryption or a decryption process. The type of cryptographic environment that an enterprise can implement can be either secret key based or a public key based. A security infrastructure can use both types. For instance, Pretty Good Privacy (PGP) uses RSA (a public key algorithm) and IDEA (a symmetric key algorithm) for the signing of messages and encryption. Public key cryptography can be used to establish a secure session between two parties. Thereafter, secret key cryptography can be used for bulk encryption between authenticated parties (remember that secret-key cryptography uses considerably less computing power and will, therefore, work faster).

Data encryption services within a PKI can be provided as part of an application program interface (API) module. This can be either a software or hardware implementation, or both. Typical services can include data encryption, decryption, symmetric and asymmetric key generation, digital signature generation and verification, hashing functions, data format operations, key storage operations, and financial services operations (for example, building a SET-compliant data block). An API determines how applications will use the PKI.

The Common Data Security Architecture (CDSA), developed by Intel Corp. and now adapted and supported by The Open Group, provides a PKI API interface. IBM, Entrust, Netscape, and the PKIX working group have committed themselves to the architecture. CDSA is a software security framework for consistently providing security-related services to applications. The services that CDSA provides are known as security modules. CDSA can be used to provide these services independent of the implementation of these security modules. Figure 12 on page 41 illustrates the components of the CDSA. The Common Security Services Manager (CSSM) is the kernel of the CDSA. It provides a top-level API (CSSM API) to applications and services that require PKI services as well as managing add-on modules.

There are four service provider modules that integrate into the CDSA environment:

- *Cryptographic Service Providers* (CSP) modules perform all cryptographic operations, such as bulk encryption and hashing. In addition, they provide secure key storage for storing private keys.
- *Trust Policy* (TP) modules are used to implement policies. They provide a framework to allow developers to manage security policies and trust operations by creating policy-specific trust modules. For example, determining the trustworthiness of a certificate holder to perform an action could be such a function.
- *Certificate Library* (CL) modules allow operations on digital certificates, including CRLs. Modules need to be created for different certificate formats.

- *Data Storage Library* (DL) modules provide an API for secure storage of certificates, CRLs, and other security objects. The API provides operations, such as search and select objects, as well as querying each data store.

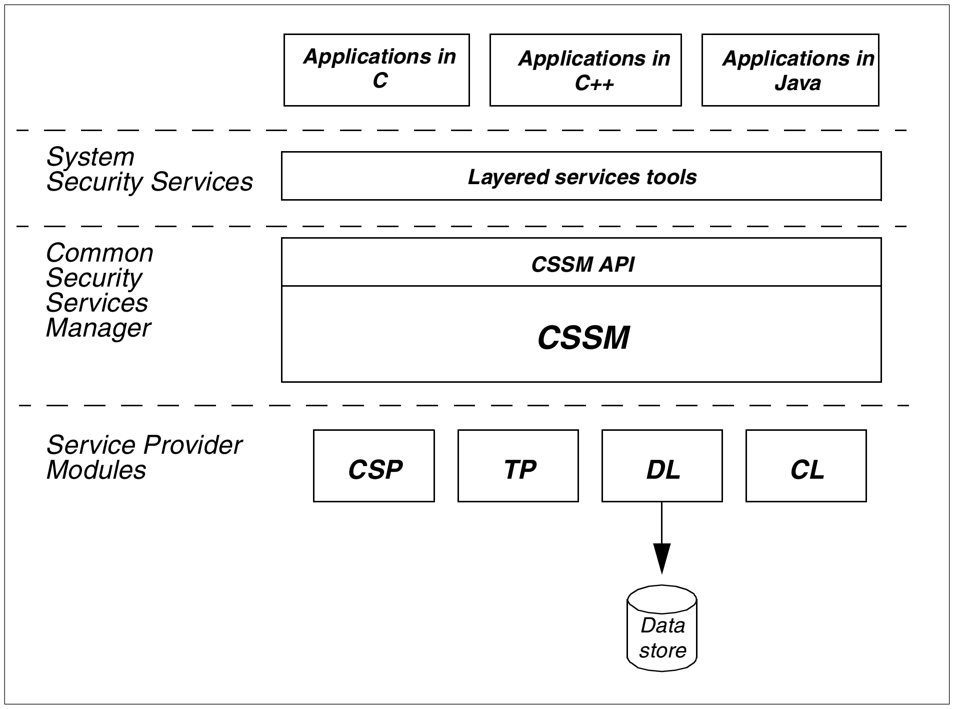


Figure 12. The CDSA architecture

An independent software or hardware vendor can create service provider modules. For example, a hardware implementation of a security module can be the IBM 4758 PCI Cryptographic Coprocessor, providing cryptographic operations. The modules can be elected depending on the implementation of the PKI. Developers at the application level need not concern themselves about low-level cryptographic operations. Because the design is modular and standards-based, applications created with this CSSM API can be easily upgraded to take advantage of new technologies as they become available.

More information about the CDSA architecture can be found, for example, at:

```
http://www.opengroup.org/security/cdsa
http://www.cdsasecurity.com
```

Encryption services can also be implemented at the session protocol level. For example, IP security (IPSec) provides security at the IP or transport layer, whereas Secure Sockets Layer (SSL) operates between applications and the transport layer. Most session security level protocols implement a variation of the following:

1. Decide on security parameters.
2. Establish a session key (or variation) to encrypt/decrypt further communications.
3. Authenticate previous communication.

SSL provides end-to-end encryption for applications, such as HTTP, FTP, Telnet, and so on. SSL provides different encryption algorithms depending on the application required. The integrity of encrypted messages is provided with the use of *message authentication codes* (MAC). Server and client authentication capabilities are provided with SSL.

During a PKI deployment, data encryption requirements at the session protocol level can vary from organization to organization. Encryption of data can be implemented for communications between the End-Entity (EE) and the RA. An End-Entity requesting a certification of its public key will send sensitive personal information to the RA which may need to be encrypted. Other information, such as requesting status information, may not need to be encrypted.

Other data encryption issues that need to be considered are:

- The CA keeps a local copy of all the certificates and the certificate-related information they issue, such as a certificate holder’s personal information. An organization will need to determine what part of this information should be stored in an encrypted format.
- Certificate Revocation Requests sent by a client to a CA can be encrypted if the implementation requires it.
- The private keys of all users may need to be stored in a central place. How the private keys will be encrypted, the key used for encryption, back up of the encrypting key, and so on need to be addressed.

Fortunately enough, the provider of a CA/RA solution product, such as the IBM Trust Authority (discussed later in this book), selects the best solution for communication and/or storage encryption wherever appropriate.

3.2 Digital signatures

Chapter 1, “Introduction to security systems” on page 3, described the significance of public key cryptography for digital signatures. Using RSA encryption and hashing algorithms, for example, a digital signature is generated. Once a user’s public key is certified, data integrity and non-repudiation can be achieved by using digital signatures. Data integrity is achieved because a verified message is compared to the original message. Non-repudiation is achieved because only the person who originated the message owns the key required to generate the digital signature.

The use of digital signatures can have major business implications. Digital signatures allow the speed of transactions to increase. This has important implications for electronic commerce as digital signature technology gains acceptance. Digital signatures are increasingly being accepted as valid forms of user identification. Traditionally, parties meet face-to-face to conclude a business transaction using handwritten signatures. However, digital signature technology allows automation of this process. Digital signatures allow individuals and organizations to conduct transactions beyond their usual boundaries. Users worldwide are able to transact.

Although a digital signature is a powerful tool for providing identification, there are two issues that must be understood when using digital signatures:

- Public key algorithms are not efficient in enciphering large amounts of data. They are processing intensive and are slow when compared to secret key algorithms. Some public key algorithms may be up to 1000 times slower than secret key algorithms.

  In order to overcome this problem, the concept of hashing is introduced. Hashing reduces the size of the original message before enciphering. Refer to 1.3.2, “Digital signatures” on page 10 for a more detailed discussion about hashing.

- There must be a way of ensuring that the public key of a sender truly belongs to the sender, and not to somebody else, someone pretending to be the sender.

  This problem is addressed using certificates and Certificate Authorities (CAs). Refer to section 3.3, “Authentication using certificates” on page 45.

These two facts are considered in Figure 13 on page 44.

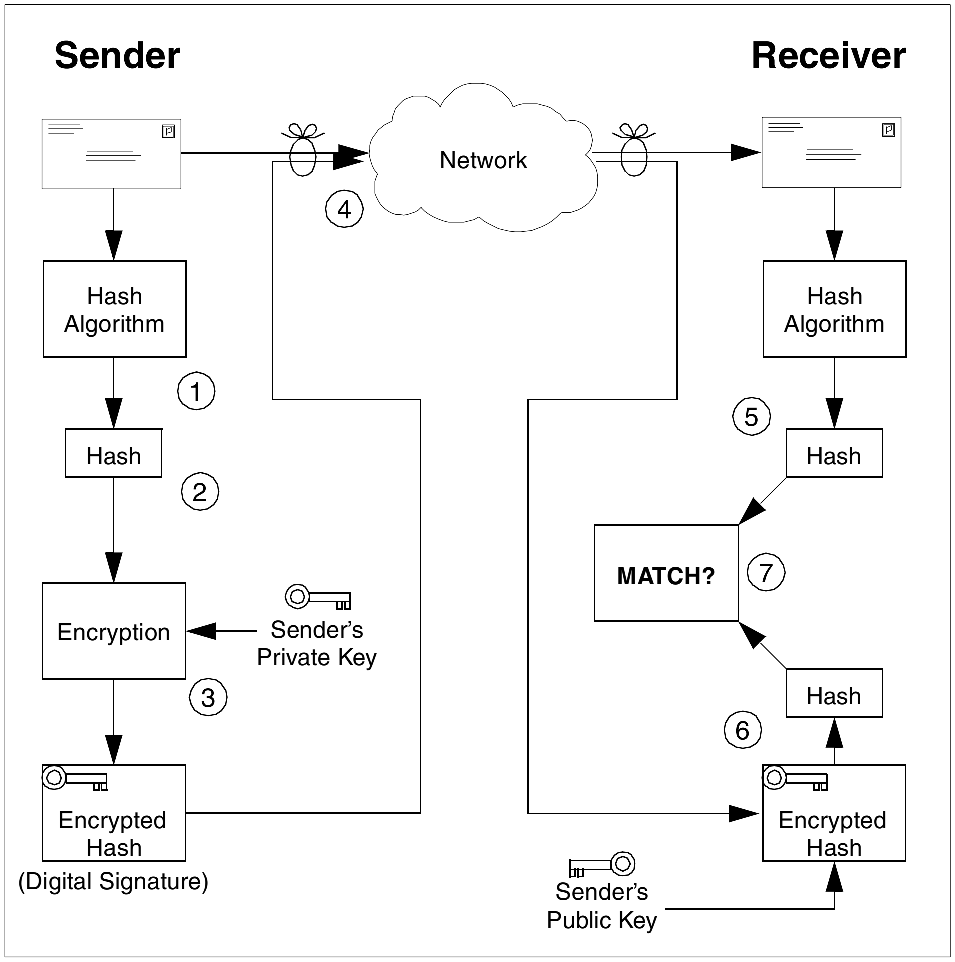


Figure 13. Digitally signing a message

Figure 13 describes how a digital signature is used in practice. The following steps describe the process (the numbers correspond to the numbers in the figure):

1. The sender hashes the message to be sent. Common hashing algorithms produce either a 128-bit or 168-bit hash.
2. The sender then needs to format the hash according to the relevant industry specification (either PKCS #1 or ISO 9796).
3. The hash value is encrypted using the sender's private key. The result is an encrypted hash value, known as a digital signature.

4. The sender transmits the original message together with the digital signature (encrypted hash value) to the recipient.
5. The recipient hashes the message received using the same hashing algorithm used by the sender. The result is a hash value.
6. The recipient uses the sender's public key to decrypt the hash value received along with the message.
7. The recipient performs a comparison of the two values to determine if the original message was altered or not and to verify that the signature is authentic, that is, if it is from the authentic sender. The first value is the decrypted hash value obtained in the previous step. This is compared to the calculated hash value obtained in Step 5. If the values match, and the recipient is reasonably sure that the public key used in step 6 belongs to the sender and is still valid, then the recipient can be sure that the message did in fact originate from that sender and that it has not been altered while in transition.

Digital signatures are not only used by ordinary users on a network. A CA, for example, uses its private key to sign an applicant's public key in order to create a certificate, thereby verifying the authenticity of the public key. Another example is, as we have seen in 2.2.1, "Certificate Revocation Lists (CRLs)" on page 29, when the CA signs a CRL as a means for clients to verify its authenticity.

Two important facts that are true for most PKI operations are:

- The receiver must rely on the fact that only the sender (and nobody else) is in possession of the private key. This is addressed by the confidentiality requirement for the storage of any private key.
- The receiver must have the correct public key belonging to the actual (authentic) sender. This is commonly addressed by the use of digital certificates, as further explained in the next section.

3.3 Authentication using certificates

Section 3.2, "Digital signatures" on page 43, mentions that when public key cryptography is used, there must be a way of knowing that a public key does, in fact, represent the entity it is supposed to represent. It will also be useful for a user of a public key to know the status of the public key. For example, the public key may have been valid at one time but has now expired, or has been revoked. Distributing public keys in a different format, as digital certificates, helps to address these issues. A digital certificate issued by a CA confirms the validity of a public key. Signing a public key and distributing the certificate to the public key owner and to a public directory service announces the status of the public key being certified.

The size, responsibilities, and implementations of CAs can vary greatly. Obtaining a client or a server certificate from a CA involves a variation of the following steps:

1. The user requiring certification generates a key pair.
2. The user signs its own public key and any other information required by the CA. Signing the public key demonstrates that the user does, in fact, hold the private key corresponding to the public key.
3. The signed information is communicated to the CA. The private key remains with the client and should be stored securely. For instance, the private key could be stored in an encrypted form on a Smartcard, or on the user's computer.
4. The CA verifies the that the user does own the private key of the public key presented.
5. The CA (or optionally an RA) needs to verify the user's identity. This can be done using out-of-band methods. For example, through the use of e-mail, telephone, or face-to-face communication. A CA (or RA) can use its own record system or another organization's record system for verifying the user's identity. For example, information from a credit bureau or a government agency would be able to confirm the user's personal details.
6. Upon a positive identity check, the CA creates a certificate by signing the public key of the user, thereby associating a user to a public key. The certificate will be forwarded to the RA for distribution to the user.
7. Finally, the CA, RA (if implemented), or the user can optionally publish the certificate to a public directory. For the CA, there is an ongoing process of checking and publishing certificate status information.

The authentication as described above when the CA verifies a subject's identity in order to sign its public key is a one-time authentication for the purpose of certificate issuance. This can be compared to the process when a government authority issues a passport to an individual. The passport then serves as an authentication mechanism when this individual travels to foreign countries. Just like passports, digital certificates can subsequently be used in daily operations for authenticating subjects to other parties that require authentication. A user connecting to a Web server with his/her browser, for example, may want to be sure that the Web server is the one it claims to be. Especially when business is being conducted over the network that involves funds transfers and legal obligations, each participating party must be able to verify its counterpart's identity.

A PKI offers means for such day-to-day authentication using CA-signed certificates. This works as follows:

Assume X needs to be authenticated by Y in order to exchange messages or conduct business. In the traditional world of paper exchange, X would sign a piece of paper with his/her handwritten signature. Using digital signatures, the process is very similar: X signs his/her messages using the private key that nobody else knows. Y can verify that signature using X's public key (see 3.2, "Digital signatures" on page 43). If the test is positive, Y can be assured that only X could have sent this message because only X is in possession of the private key that was used for signing the message.

A valid question may arise at this point: How does Y know that the public key used in this check was really X's public key? If this cannot be assured, then anyone could have sent this message using another private key and the test would still be positive if Y had been given the corresponding public key. This is where certificates come into play. Remember that a certificate represents a public key that is signed by a trusted CA. Remember also that there is no practical way to falsify such a certificate. Having this in mind, if Y is in possession of X's certificate, Y can retrieve X's public key from that certificate. Provided that Y trusts the CA that certified (signed) the public key, Y can be sure that the public key is actually X's public key and no one else's. This way, Y can securely validate X's signature.

The important point here is that Y must trust the signing CA. In other words, Y must be in possession of the CA's public key (or its certificate that contains the public key). It is, therefore, an important point when deploying a PKI to make sure that the involved parties have a valid copy of each CAs certificate. Refer to 3.5, "Certificate validation" on page 49 for more details about this.

To see in an example of how that is achieved in practice, let's have a look at a Web browser. For secure communication with Web servers, the SSL protocol is widely used. SSL uses public key cryptography for authentication and message encryption. When a secure communication with a Web server is being established, a Web browser receives the server's certificate in order to authenticate this server. Most Web servers use certificates that have been issued by one of the well-known CAs, such as VeriSign Inc. or Entrust Technologies Inc. Most Web browsers, on the other hand, already come with a list of trusted CAs. Therefore, when a Web browser is installed on a computer, this list of trusted CAs is installed automatically as part of the browser. Because the browser has this list already built in, it can then verify any certificate that has been signed by any of the well-known CAs. If a browser receives a certificate that was signed by another CA, not known to the browser, it will refuse the connection and/or ask the user through a pop-up window for further actions to be taken.

3.4 Non-repudiation

As explained in 3.2, “Digital signatures” on page 43, digital signatures can function like (and, therefore, even replace) handwritten signatures and they can serve as a means for non-repudiation. In summary, a digital signature confirms that the signer was in possession of the private key when he or she signed the document. This is, however, not enough for a reliable non-repudiation solution because an attacker could have signed the document later with a stolen private key. Thus, the date and time when the document was signed becomes an important issue, just like handwritten signatures that are almost always used in conjunction with a date. Digital signatures may need to be verified even after a certificate (representing the signer’s public key) has expired or even revoked. Depending on the corporate security policy, a revoked certificate may still be used to verify a digital signature as long as there is a secure way to prove that the signature was created before the certificate was revoked.

A signer’s local time, that is, for example, the local clock in the signer’s personal computer, is not suitable for the non-repudiation digital signatures, because the stolen private key could be used on a malicious person’s computer system that has the time intensionally set behind. Therefore, a reliable time service must be established as a Trusted Third Party (TTP). Two Internet Drafts address this, one describing a *Time Stamp Protocol*, which includes a *Time Stamp Authority* (TSA) and a *Temporal Data Authority* (TDA), and another describing a *Data Certification Server Protocol*, which includes a *Data Certification Server* (DCS). See 5.3, “Internet Drafts” on page 104 for more details about these draft standards.

A TSA replies with a *time stamp token* upon a client’s request. The client should verify the TSA response containing the time stamp token. For example, the client will check the response status, compare the time stamp value with local time, or verify the time stamp token’s digital signature. Optionally, a Temporal Data Authority (TDA) is proposed. A TDA is a TTP that creates a *temporal data token*. This temporal data token associates a datum with a particular event and provides supplementary evidence for the time included in the time stamp token.

A DCS also returns the *data certification token* upon a client’s request. While a TSA only returns the correct time to a client, a DCS verifies several things for non-repudiability evidence. A DCS verifies the digital signature value contained in the request, checks the revocation status of the End-Entity, and also checks the full certification path from the signing entity to a trusted point, such as the DCS CA, or the root CA in a hierarchy. Therefore, a DCS token is more reliable than a TSA/TDA token. The presence of a DCS token has two meanings:

- It provides evidence that a signature or public key certificate was valid at the time indicated in the token. The token can be used even after the corresponding public key certificate expires and its revocation information is no longer available on CRLs.
- The production of a DCS token in response to a signed request for certification of another signature or public key certificate also provides evidence that due diligence was performed by the requester in validating the signature or public key certificate.

A time stamp in a DCS token comes from either a TSA or another DCS. Having all this in mind, it becomes obvious that a time certification service must be a highly trusted service, much like a Certificate Authority (CA).

We must also think about another potential problem: How can a digital signature be verified until the end of its validity period, which may be several decades? This may be necessary, for instance, if a digital signature is used to sign a long-lasting contract. In the period after the initial sign, the DCS key length or the signature algorithm might be no longer enough to prove the DCS token's validity. The extension technique for DCS tokens is also described in an Internet Draft. In the Draft, a *non-repudiation token storage format* is proposed, and the original DCS token is put as a root node in the storage. This works as follows: a client requests the verification of signed data, which is done by a DCS service. The DCS server, in turn, then returns a new DCS token to the client. The new token is added as the leaf node in the non-repudiation token storage. An application with a requirement for verification of digital signatures can then accomplish its purpose by verifying the leaf DCS token's digital signature.

3.5 Certificate validation

Users of public key certificates need to have a way of validating certificates and obtaining certificates if necessary. Web browsers, for example, do not need to obtain many CA certificates (root-level certificates) because they are already hard-wired into the application; it is implicitly trusted that those root-level certificates belong to their respective owners. However, a situation may arise where an application may need to verify a user's certificate where it does not have a trusted copy of the CAs certificate who signed it. In such a situation, the application may need an additional certificate in order to get that CAs certificate. In RFC 2459, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, the use of multiple certificates (or chains) is described, which may be needed to validate a certificate. These chains may consist of the public key owner's certificate signed by a CA, and optionally, one or more CA certificates signed by other CAs. These chains are known as *certification paths*. The purpose of certificate path validation is to make certain that a user's public key in a certificate is valid (according to the basic and policy constraints extensions) and that it does indeed belong to the subject in that certificate.

RFC 2459, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, provides a description and algorithm for validating certification paths. The RFC does not require other implementations to use the algorithm, as long as the implementation is "functionally equivalent" in terms of the resulting output. The RFC distinguishes between two paths: the *basic path validation* and the *extending path validation*. The basic path validation assumes that there is a single most-trusted CA and that all path processing begin with certificates issued by this CA. Extending path validation extends the basic validation algorithm. This validation path introduces multiple CAs and situations where processing needs to conform to PEM (RFC 1422) specifications.

In any case, verifying the validity of a certificate may not be a trivial task. A full certificate validation is, in fact, considered a difficult problem in a PKI that is addressed in several RFCs and Internet Drafts. The Internet Draft titled *Simple Certificate Validation Protocol (SCVP)* proposes a protocol that clients can use to offload the validation task to a specialized certificate validation server. But why is certificate validation considered difficult? For an idea of what is involved, consider the following list, which may not even be complete:

- Every certificate involved in a validation process must itself be validated.
- Every certificate involved in a validation process must be checked against a valid CRL.
- Any certificate extension that is marked as critical must be understood and checked.
- Any CRL extension that is marked as critical must be understood and properly handled.
- Any other party involved in the validation process must be trusted.

In practice, many solutions simply check whether a certificate signer's certificate is available and, at the time of checking, still valid. This is what Web browsers do; they do not consult a CRL to verify either a server's certificate or a CA (signer) certificate.

3.6 The certificate life cycle

An organization's overall security policy needs to determine the life cycle of the certificates it uses. The type of application a certificate will be used for will also determine its life cycle. For example, a CA providing certification services can distinguish between personal certificates issued to regular users and certificates issued to Web (or other) servers. A certificate life cycle can also be governed by any subscriber agreement that the user enters into with a CA. For example, if a user provides false information about his or her identity, a CA will be entitled to revoke that user's certificate. The security policy of an organization will also determine the backup procedures needed. Once a user's private key has been compromised or lost, the corresponding public key certificate should be revoked. Chapter 4, "PKI deployment" on page 59, explores important security policy issues and certificate and key management issues for an organization wishing to deploy a PKI. The following subsections describe the fundamentals of the certificate life cycle: creation revocation, expiration, and suspension.

3.6.1 Certificate generation process

A user will need to be positively identified before a certificate is allowed to be generated and issued to that user. The extent of user authentication depends on the type of certificate being requested and on any security policies. Authentication for an e-mail certificate may require no more than verifying if a user can receive e-mail at the given e-mail address, whereas a certificate for a merchant Web server may require official documentation, for example. The CA component is responsible for generating and issuing a certificate. Once a user's details have been verified by the RA component (assuming an RA has been implemented), a request will be sent by the RA to a CA for a certificate. The CA can receive the request in the form of a template that includes the public key to be certified. The CA can then apply its appropriate signing key to the certificate, effectively signing the public key. The relevant fields in the template can be updated by the CA, and the certificate is then forwarded to the RA. The CA can also keep a local copy of the certificate it generated. A certificate, once issued, can optionally be published to a public directory.

3.6.2 Certificate revocation process

Certificate Revocation Lists (CRL) are one of the most important elements that provide security and integrity of a PKI. Certain situations may cause a certificate to become invalid before its expiration date, for example, when a user's private key is lost or compromised. Both, a user (EE) and the CA can initiate a revocation process because both may have reasons to do so. Upon such a revocation process, a CA updates its internal records and any CRL with the required certificate information and timestamp. A revoked certificate is identified in a CRL by its certificate serial number. To take effect, the CA signs the CRL and places it in a public repository. Other applications using certificates can access this repository in order to determine the status of a certificate.

3.6.3 Certificate expiration process

Every certificate that a CA issues should have an expiration date. A CA can issue certificates with a standard validity period. However, the validity period may depend on the circumstances in which the certificate is being used. After the expiration date, the certificate may not be used any more for any kind of authentication and/or validation. A user can be reminded by the issuing RA (if implemented) of the upcoming expiration of a certificate. Then, a user will be required to follow a renewal process. If this process is successful, it will result in the user being issued a new certificate that has a new expiration date. Note that this process does not normally include the generation of a new key pair; the old key pair can still be used.

3.6.4 Certificate suspension process

In addition to a certificate revocation, a certificate can also be put on hold, or be suspended. This may occur for a variety of reasons, as, for example, when an employee is going on vacation. During the period of the vacation, the user may want to be sure his or her certificate cannot be used for any fraud authentication process until he or she returns. On the employee’s return, the suspension can either be withdrawn or the certificate can be revoked. Another example of temporary certificate suspension may be if certificates are preregistered and pregenerated by a security authority before they are distributed to and used by the users.

3.7 Administration

So far, we have shown how public key cryptography can be used to greatly enhance security across a distributed environment. This section describes what administration activities need to be addressed when deploying or operating a PKI. It is also shown that the successful operation of a PKI relies on effective key and certificate management.

Key pairs and certificates are generated and used at different times and stored at different places throughout a network. Multiple key pairs can be generated for different users. One key pair could be issued for encryption purposes, while another one is issued for signing purposes. Key pairs could also be issued depending on the applications they will be used for and the levels of trust associated with the application. Some users might require a certificate for securing their e-mail messages, as opposed to a banking institution that may require a server certificate in order to provide server authentication services to its clients. In the case of a CA generating keys for an End-Entity, a secure means of distribution of these keys must be introduced. In other words, if a private key is generated for the user, it must be distributed to the user in such a way as to prevent eavesdroppers from obtaining the clear key value of the private key.

Administration of a PKI also involves all backup and recovery procedures. An organization will have to decide what data needs to be backed up and how often according to the organizations policies. An organization that distributes key pairs to its employees may want to store a copy of the key pairs in a secure area in order to be able to decrypt certain information later if required. The key pairs need to be stored securely and an organization can consider using cryptographic hardware to store these keys.

Administration of key updates for a PKI is another important task required. Organizations have to decide when to generate and distribute new key pairs to users. They also have to decide who will be required to have their keys updated and how. An update may be designed to work semi-automatically, or it may require a user to manually initiate the process. Finally, some keys may be used for sensitive transactions and so may require updating more frequently than others.

Certificate management is the administration of certificates. This activity includes certificate generation, certificate revocation, administering a CRL, and the creation and implementation of a trust model for the specific organization. During the certificate management process, the correct management of a CRL becomes very important, because it determines whether a key (or certificate) should be trusted or not.

Certain applications, for example, financial transactions, may require online certificate validation. A stockbrokerage firm receiving a request from a client will need immediate verification that the certificate presented is valid. If the certificate presented had been revoked 30 minutes ago, and the CRLs next update is in an hour, the certificate could still be regarded as valid by the firm. Issues regarding when a CRL should be updated or whether an online CRL is required by the organization needs to be examined and defined in security policies.

Additional administration involves defining the roles of security personnel administering the PKI and the skills required from them. For example, an organization could require an overall administrator who will be responsible for generating keys and updating the CRL if necessary. Other security staff will only be responsible for doing investigative work that involves verifying the identity of a user applying for a certificate and updating information of a user in the relevant database.

Chapter 4, “PKI deployment” on page 59, provides a detailed discussion about administrative issues to consider when implementing a PKI.

3.8 Administration domains and hierarchies

Generally, a CA will provide certificates to a fixed group of users. An environment where users share the same CA is sometimes referred to as a *security domain*. For example, a large corporation can act as a CA, issuing certificates to its employees. Users who share the same security domain will find it easy to trust each other’s certificates since all users trust the CA who signed those certificates. When user X receives a certificate from user Y sharing the same security domain, user X will compare the digital signature in the certificate presented to the one stored locally. If they match, X knows that Y is trustworthy, and a secure session can be started. Certain applications may require that people in different security domains communicate with each other. For example, X, an employee belonging to an organization with its own CA, may want to send signed and encrypted mail to Y, who is outside the organization, and who belongs to a different CA. In order to do so, the following should be considered:

- X and Y need a way of obtaining each other’s public key certificates.
- Y needs to be sure that he or she can trust X’s CA. X needs to be sure that he or she can trust Y’s CA.

People and applications from different security domains will need to trust each others certificates. Therefore, different CAs need a way of trusting each other, so the roles and relationships between CAs has to be defined. The purpose of creating such trust relationships is to eventually achieve a global, interoperable PKI.

The implementation of a trust model between organizations, for example, will be largely influenced by the business policies and practices of these organizations. The implementation of a trust model can vary from organization to organization and can become complex. Refer to Chapter 4, “PKI deployment” on page 59 for a discussion of factors to consider during implementation.

One implementation allows for CAs to cross-certify each other. In such a situation, the CAs certify each other's certificates. Consider the diagram shown in Figure 14.

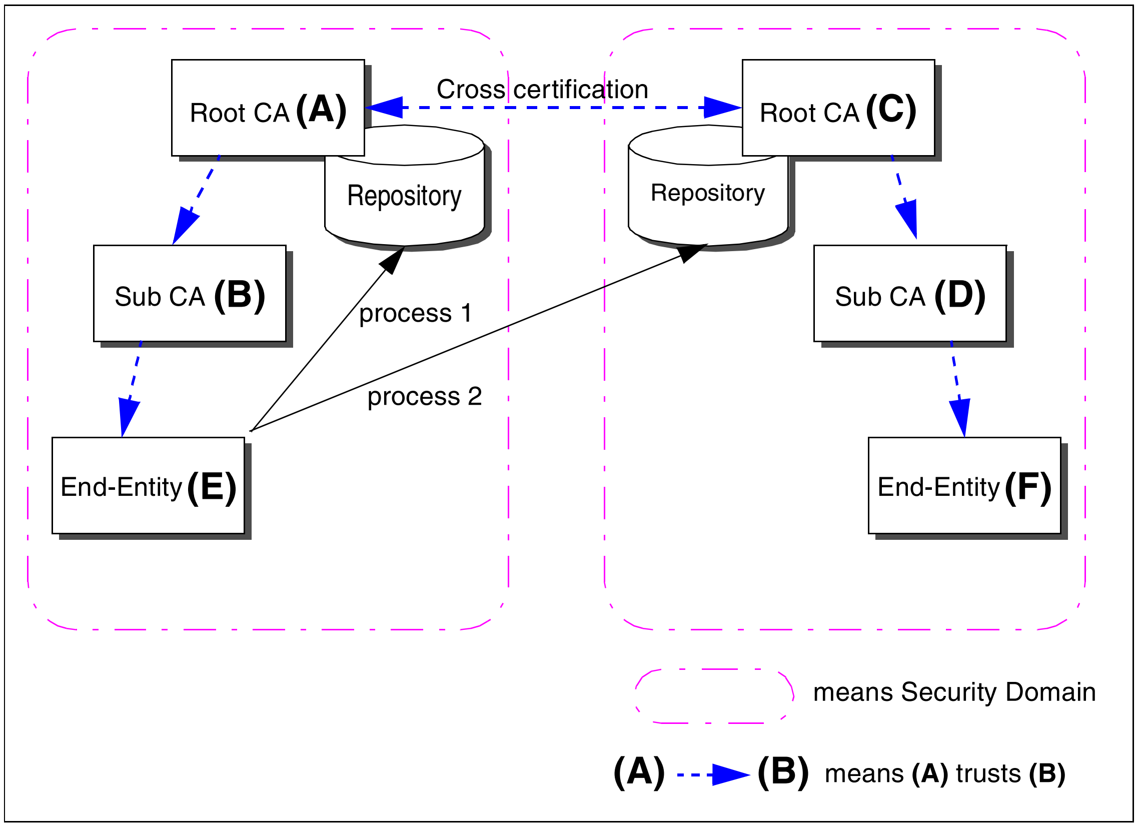


Figure 14. Cross-certification between two security domains

Figure 14 shows two security domains. An End-Entity (EE) can represent a user, or an application. For our purposes, we will assume that the End-Entity represents a person. User (E) relies on CA (A) for certificates. User (F) relies on his or her own CA (C) for certificates. The subordinate CA servers (B) and (D) are optional sub-CAs of a PKI, and certain management tasks can be delegated to them by the CA. (A) provides a repository for users to obtain certificates of other users belonging to the same security domain (process 1). (C) does the same for users belonging to its security domain.

In the configuration shown in Figure 14, users in the security domain governed by (A) can obtain public key certificates belonging to users of another security domain (process 2). (C) does the same for users belonging to it security domain. The root CAs (C) and (A) cross-certify each other, which, in this implementation, means that the public key certificate of (C) will be signed by (A) and stored in (A)s repository, and the public key certificate of (A) will be signed by (C) and stored in (C)s repository. Now suppose that (F) wants to communicate securely with (E). (F) could send its public key certificate to (E). (F)s certificate is signed by (C). (E) will receive the certificate and access a repository to confirm the validity of the certificate (process 1). Because the public key certificate of (C) has been signed by (A) whom (E) trusts, (E) will trust the certificate presented to him by (F). The following trust relationship exists:

- A trusts C, and C trusts A. They have certified each other’s certificates.
- E trusts A and F trusts C. A has certified E’s public key certificate, and C has certified F’s public key certificate.
- Therefore, E will trust F’s certificate, and F will trust E’s certificate.

Cross-certification is believed to be beneficial to organizations that share business relationships. Many organizations may feel more comfortable following a cross-certification path initially because it may reflect current business relationships. For instance, business partners or organizations involved in mergers may decide to cross-certify in order to provide easier communication between employees. Cross-certification can also be seen by many organizations as a chance to define business relationships and, therefore, may also be a reason for early adoption. However, cross-certification is believed to be impractical for establishing trust between a large number of CAs because the infrastructure may become difficult to manage as the number of CAs grows.

A more indirect implementation of a trust relationship is the hierarchy model. In such a relationship, a number of CAs create vertical relationships with each other. This allows users that do not share the same CA to review the hierarchy until they find a CA that they both trust. Hierarchical structures consisting of multiple CAs have been suggested by standards authorities, such as X.509, in order to address issues like scalability.

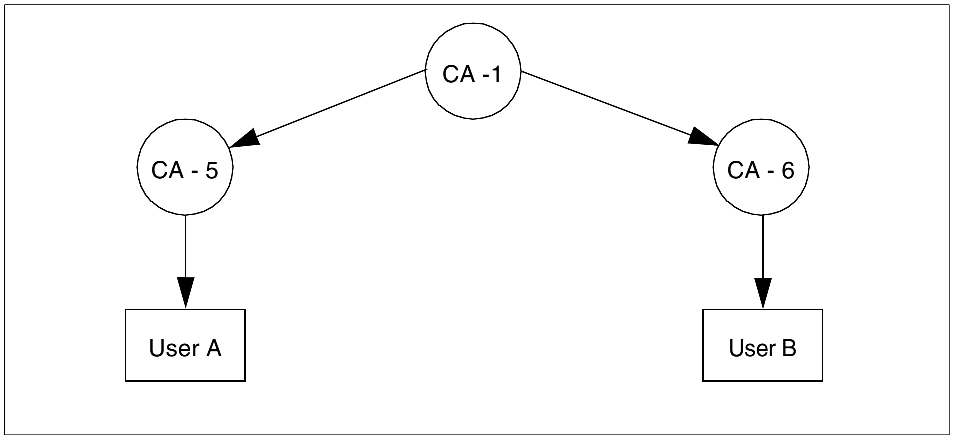


Figure 15. A simple hierarchical structure

Figure 15 is an example of a simple hierarchy. In this example, CA-1 is a root Certificate Authority that signs (certifies) the public keys of CA-5 and CA-6. CA-5 is a Certificate Authority that issues certificates to a user, user A. CA-6 is a Certificate Authority that issues certificates to user B. Suppose that user A wanted to communicate with user B. User A could send his or her certificate to user B. However, user B does not trust user A's CA directly. In order to establish trust, the users can find a CA that they both trust, namely CA-1. This is the essence of a hierarchy; CAs certify each other up and down a hierarchy, building a chain of relationships. This is sometimes referred to as a *chain of trust*. Therefore, user A and user B can scan the hierarchy until they find a CA that they both trust. Users can then authenticate each others public keys using the CA certificates issued by the trusted CA (CA-1). This process is sometimes referred to as *certification path processing*.

Organizations will want to and should maintain the management and administration authority of their security domains. In the above diagram, CA-5 and CA-6 are still in control of the administration of their respective security domains. If, however, administration of a security domain is transferred to a central authority, for instance, then administration of the domains could become very complex as the hierarchy grows. Local administration makes certificate management easier.

There are a number of hierarchy structures that have been proposed. These range from strict hierarchies requiring that everybody needing a certificate eventually trust one authority at the root of the hierarchy (PEM, RFC 1422) to proposals by the IETF to eliminate hierarchies altogether (SPKI). Currently, CAs are cross-certifying each other in order to gain interoperability in the marketplace. As the number of CAs increases, a more hierarchical model could be introduced in order to address scalability concerns.

Chapter 4. PKI deployment

The previous chapters introduce and explain the building blocks that make up a PKI. It has been pointed out that the surrounding infrastructure, besides the pure public key technology, is of significant importance for a successful PKI deployment. While some basic PKI operational principles are defined in RFC standards and Internet Drafts (see Chapter 5, “PKI-related standards” on page 101), much remains to be elected and defined individually by the organization that deploys a PKI.

This chapter gives you a more detailed view of what steps are, or might be, involved when a PKI is being introduced. It is very important to notice, however, that a practical implementation depends very much upon an organization’s individual policies, goals, and other determining factors. It is, therefore, not the goal of this chapter to provide bullet-proof solutions, but to list, explain, and discuss work items and potential issues that an organization needs to work out according to its individual business practices and requirements. Yet other items, such as cost of implementation and operation, or project management issues, are important factors, too, but they are beyond the scope of this book and, therefore, not further elaborated.

It should be mentioned at this point that IBM Global Services offers consultation and implementation services for organizations that seek professional help for a successful PKI deployment, including:

- Current security assessment
- Work out general security and specific PKI requirements
- Design and implement security and PKI solutions
- Other specialized tasks as per request

You can read more about IBM security service offerings at:

`http://www.ibm.com/security/services/index.html`

4.1 Defining security policies

One of the first steps an organization has to consider when comprehensive security solutions are to be introduced is to define a feasible set of security policies. In the first place, this has little to do with a PKI because security policies need to be in place for any kind of I/T infrastructure. Only when the deployment of a PKI has been decided, some additional benefits and issues come up that need to be defined within security policies. The following subsections discuss security policies that primarily relate to a PKI.

4.1.1 Scope

Defining the scope of an organization's security policy may be discouraging for those required to do so. Digital certificates are used as the basis for providing confidence that communication between parties is trusted and secure. Therefore, a security policy needs to address all possible security-related issues in that environment. An inadequate or unclear security policy that leads to an error in the implementation of the security system could affect the integrity of the entire PKI. Generally, your organization should consider the following areas when defining a suitable security policy:

- If an RA will be part of the operation of a PKI, additional policies may have to be defined specifically for this component.
- Creation of one or more certificate policies and a certificate practice statement (CPS). These policies have to be coordinated with any other security polices of the organization.
- The means by which the CA keys will be protected.
- The process of how a CA key compromise will be handled.
- The means by which PKI repositories will be protected.
- The process of how users and CAs will be verified.
- The range of PKI operation.
- The process of how the PKI will be audited.
- The process of how PKI policies will be approved, implemented, and maintained.

4.1.2 Trust model

A fundamental aspect of PKI is the fact that public keys, normally in the form of signed certificates, must be trusted. If this cannot be ensured to the level required, the whole PKI suffers from a serious "weak link in the chain." This is normally achieved by trusted CAs that sign public keys. This acceptance forms trust relationships between the various parties involved in a PKI and, therefore, allows users to regard their transactions as trusted transactions.

A large-scale deployment of a PKI may involve multiple CAs, and thus, multiple security domains. In order for your organization's implementation to be interoperable, relationships between these CAs have to be defined. In addition, a CA could issue multiple types of certificates to users, creating multiple trust relationships. The form of such relationships will vary and will depend on factors, such as the scope of your organization's deployment, users of the certificates, the type of applications used, the type of business operated, and so on.

Relationships between interoperable CAs create one or more certification path. As described in 3.5, “Certificate validation” on page 49, a certificate path validation is the process of how a certificate will be trusted (or verified) by another party. Depending on the structure between CAs, the certification path will follow a number of connected nodes between the holder of a certificate and the root-level certificate. Remember that a root-level certificate has its public key trusted by a certificate-using application. Once an application is presented with a certificate from a user, it can validate that certificate using a certification path that maps to a root-level certificate. In many cases, like with most Web browser SSL connections, this will be just a one-level validation path where the browser already has the signer’s certificate.

Organizations have to decide on the certification path they will follow. In other words, how they will go about trusting a public key. In a simple situation, your organization issues certificates to its own employees. All certificate holders trust the same CA, so the certification path only requires one certificate. Your organization will need to determine whether they will only trust certificates from users or applications in their own domain, or from other domains, too. Your organization may decide to only trust a finite set of domains. In such a situation, a cross domain certificate could be issued to participating domains, thereby introducing your trust model to other domains. If your organization wants to provide a global secure e-mail application, for example, a more complicated structure will be required where CAs will be required to cross-certify each other. This will allow a certification path between any two certificate holders from any domain. An organization considering deploying a PKI will need to define how to handle certificate chains. If your organization receives a certificate chain, how will this be validated? Will a root key, imbedded in the user application, be used for validation? Or will your organization’s public key be sufficient? Or perhaps, your public key will be sufficient to validate the certificate chain if your organization operates its own “in-house” CA.

A trust model is not only important for defining relationships with external parties, but also plays a significant role in defining relationships and processes between parties within an organization. For example, corporations in certain countries traditionally have steep corporate hierarchies. In such a situation, the organization may want to have a central CA, with subordinate CAs representing divisions or departments of the organization. An electronic trust model for that organization can then be based on the traditional relationships and business rules shared between departments in the organization.

Finally, an organization’s trust model can form the basis for subscription and Chained CA Certificate Practices and Liability Agreements. Your organization needs to consider the following questions. To what extent will the organization be liable for certificates issued by the CA it cross-certified? How will dispute resolution be solved? Under what circumstances can a user hold a CA liable? The level of trust that should be associated with the certificates that are being used should be defined by the CA. For example, a CA can document the degree of its liability when issuing certificates, thus helping the user to decide on the degree of trust in a certificate signed by that CA. In fact, all systems operations that affect the level of trust should be defined and documented in the organizations IT security policy. A certificate policy or Certificate Practice Statement (CPS) can be used to help users decide whether or not to trust the certificate presented to them. They are the foundations of trust in a PKI. For a detailed discussion, refer to 4.1.6, “Handling guidelines” on page 73.

4.1.3 Registration process

The registration process is the process of an End-Entity, such as a user, requesting a certificate and being declared eligible for public key certification by a CA. The registration process involves two critical steps:

- Subject enlisting
- Subject authentication

Subject enlisting is the process of a certificate user applying for a certificate. Depending on the implementation, a user may be required to fill in an online application form. Typical questions may be asked in order to establish the identity of the user, such as name and personal address of the applicant. The level of information required from the applicant will depend on the type or class of certificate that is being applied for. As we recall, a certificate binds a public key to a subject; in other words, the CA certifies that the public key contained in the certificate belongs to the subject (owner) of the certificate. In some cases, a certificate may only bind a public key to a subject identified by his or her e-mail address. In other cases, the CA certifies that a public key really belongs to a specific individual, like a personal passport issued by a government agency. In the latter case, the CA (or optionally an RA) may require an applicant to apply in person with some legal documents, such as a birth certificate.

If an organization intends acting as a CA to its employees, the applicant may not be required to supply as many personal details since this information is already on record. In such a situation, only a justification for the certificate may be required by the user, or nothing at all if that organization issues a certificate to each employee.

In addition to the information required from the applicant, a public/private key pair needs to be generated and the public key needs to be sent together with the applicant information to the CA (or RA if this component is in use). Where the key pair is generated will be an essential implementation issue for an organization. One option is for the user to generate his or her key pair on their workstation and keep the private key stored securely. Alternatively, an organization may require that the key pair be generated by a central point in the organization. There are two reasons for this. First, it enables the organization to keep a backup copy of the user's private key. This may be important for encryption keys. A user losing his or her private key and, thus, being unable to decrypt information may be costly to an organization. Secondly, the organization may want to use specialized hardware for generation of higher quality keys. Whenever a private key is generated in a central place, a secure means of transporting the key to the user must be in place.

Once the CA or RA receives the subject's certificate request, the process of *subject authentication* must take place. The CA or RA needs to confirm the applicant's details before the public key is bound to the applicant. If a CA issues a certificate to an imposter, binding somebody else's public key to that person, the imposter can pretend to be that person and may be able to carry out transactions in that person's name. As previously mentioned, the degree of identity checking done by the CA or RA will depend on the class of certificate requested.

Generally, commercial CAs require an applicant to agree to a subscriber agreement. The subscriber agreement is a series of steps required between the applicant and the CA and it may be detailed in the CPS of the CA (refer to 4.1.6, "Handling guidelines" on page 73). A CA can also offer additional plans that offer greater warranty protection and other extended assurances for a fee or included in the price of the certificate. For example, the CA could detail the amount it would pay to a subscriber in the event of the CA breaching one of the warranties.

4.1.4 Certificate revocation process

Certificate management involves administration of public key certificates and their CRLs. The entire integrity of a PKI relies on the use of CRLs. Without CRLs, there is no way for certificate verifiers to know whether the certificates they are using are valid or not (besides checking for the expiration date). Therefore, correct management of CRLs is extremely important to the operation of a PKI.

An organization wishing to deploy a PKI has to determine who it will accept certificate revocation requests from. Usually, an owner of a certificate will request a certificate revocation, for example, if the user's private key has been lost, or if the user suspects that his or her private key has been compromised. The CA or RA (if used in the implementation of the PKI) could ask the user for a challenge phrase that will provide authentication of the user. In certain situations, a person other than the owner of the certificate may ask for a certificate to be revoked. For example, an employee leaving an organization may require his or her manager to request revocation. In other situations, the CA that issued the certificate, or perhaps another cross-certified CA, may request a certificate revocation if it is discovered, for example, that the owner of the certificate contravened CA policies. In situations where a person other than the certificate owner requests a certificate revocation, there must be a means of identifying the requestor and determining whether the requestor has authority to request revocation. This will be determined by the security policy of the organization. Once a revocation request has been received and authenticated, the CA will be responsible for updating its internal records and publishing the CRL.

Generally, a PKI provides some capability to allow expired certificates to be placed on CRLs. Publishing revoked certificates to a CRL can be straightfoward if the management of certificates is confined to a reasonably small organization. Applications can access the CRL in the background, verifying whether the certificate they are about to use is valid or not. Some applications may have the latest CRL cached since this will increase performance and allow for the application to operate offline. However, once an organization manages certificates from other domains and uses the Internet, for example, management of a CRL may become difficult. Specifically, performance and storage requirements become major issues the larger the CRL grows.

In addition, the organization needs to decide how to publish a CRL—what distribution points to use. The type of CRL to be used also needs to be decided. For example, an organization could publish different lists depending on the severity of the revocation. One list could represent keys that have been compromised, while another list could represent certificates details that have changed.

Deploying a PKI requires an organization to decide how often to publish a CRL. The method used by a CA to publish CRLs should be clearly defined in an organization's security policy. There are three ways a CA can publish a CRL:

- Pull method. This method relies on the certificate-using application to periodically check recognized repositories for the latest CRL update. The CRL can then be “pulled” down for use by the application. The application can check certificate validity depending on the next scheduled update that will be published in the CRL. A CRL will be signed by the appropriate CA before being placed in the repository.
- Push method. This method relies on the CA broadcasting a CRL to certificate-using applications every time a certificate has been revoked. The CRL is “pushed” to applications.
- Online verification. This method requires the certificate-using application to execute an online query with a particular CA before validating a certificate.

Each method has its advantages and disadvantages. An organization will have to decide which method to follow, depending on its particular implementation. The nature of the transactions that will be used across the PKI and the level or risk the organization is willing to accept will determine the method used.

The *pull method* allows for a period whereby the certificate may have been revoked, but the CRL may not have been updated to reflect this yet. This situation allows for the revoked certificate to still be regarded as valid by others until it has been published in the CRL. Organizations using sensitive applications may not feel comfortable using this method, even if the scheduled updates of the CRL are performed quite frequently. Nevertheless, the pull method is the method of choice for most solutions unless other reason dictate another method.

The *push method* may be suitable for small organizations operating their own PKI, using a small number of PKI applications. This method may not be suitable for a PKI servicing a large community of users using multiple certificate applications. Determining which applications to broadcast to and distributing the CRLs incurs other problems, such as the synchronization issue if applications are not available while a CRL is being distributed. Also, a CA needs to know the applications to which it is supposed to push the CRL, which creates an administrative overhead.

The *online verification* method provides a number of useful advantages. First, it allows for a timely delivery of revocation information. This method will also be advantageous to applications that require certificate verification before a transaction can take place. The disadvantages of this method is that the CA will need to provide a secure server at all times and will have to ensure that each query is digitally signed, which may create processing bottlenecks.

A possible solution for an organization is to distinguish between certificate usage. A CA could then provide online verification services for certificate classes that are used for sensitive applications, such as secure electronic commerce and other legal transactions, and provide the pull method for less critical certificates, for example, those used for everyday e-mail communication.

Finally, the IT security industry could move to repository-based infrastructure, where legislation would require (one or more) official and registered repositories to hold certificates and CRLs. The CA could then outsource such activities to the repository administration. In the USA, for example, the state of Utah has already set up laws to promote such activities.

4.1.5 Certificate and key lifetimes

Implementing a PKI involves significant certificate and key management issues. An important point to consider is at what point in time certificates and keys become valid and how long they remain valid before they need to be replaced or reissued. Different considerations apply for certificate and key lifetimes as discussed below.

4.1.5.1 Certificates

We start by defining when a certificate becomes valid. This is an important question because of the potential legal implications of digital signatures. An organization may have authenticated a user requesting a certificate for his or her public key. However, a CA signing and issuing a certificate does not necessarily make a certificate valid. It seems logical that just as a certificate becomes invalid when listed in a CRL and published in a public repository, a certificate should become valid when published in a public repository by the issuing CA. Some CAs require that the user of a certificate first accept certain subscription or policy agreements before the CA publishes the certificate. These CAs issue “operational certificates” to users that requested certification. In practice, the certificate may be sent to the user with a subscriber agreement requesting the user not to use the certificate until he or she formally accepts the agreement. Once the CA receives acceptance from the user, the CA publishes the certificate in a public repository, thereby making it valid. Therefore, an organization should define clearly when a certificate becomes valid.

Once an organization determines that a certificate is valid, the certificate and the certified key will be available for general use. Usually, an organization will issue multiple certificates to a user. In many practical situations, a user has at least two key pairs—one key pair for encryption and another key pair for digital signatures. Users will need to hold different certificates to represent the different relationships they have in the real world. This will mean different security levels, expiration times, and additional certificate management. If an organization intends to act as a CA to its employees, it will have to determine what type of certificates it will issue and the lifetime of those certificates. These specifications need to be detailed in the appropriate security policies.

There is no technical reason for a limited certificate lifetime, and thus, certificates can remain valid for extended periods of time. For practical reasons, however, most certificates have a defined expiration date (see also 2.1.5, “X.509 certificates” on page 25). A short certificate lifetime can statistically lessen the chance of misuse, but adds the additional burden involved for frequent re-issuance. Most personal certificates in practical PKI deployments are valid for a period of one or two years after issuance, while server certificates typically remain valid for two years or more. Specialized certificates may be valid for extended periods of time. Such long-lasting certificates may be handy for archival and long-term encryption purposes.

Figure 16 on page 68 illustrates a possible certificate life cycle. The solid arrows indicate the normal life cycle of a certificate, where the certificate life cycle ends as expected. The dashed arrows indicate where a CA or RA intervenes in the life cycle of the certificate.

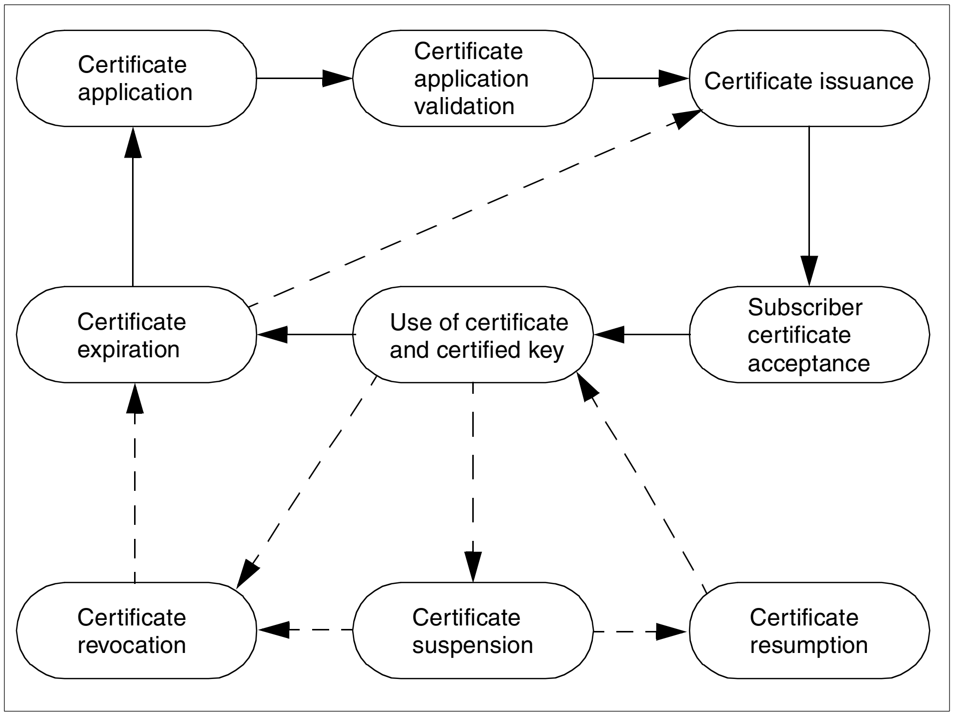


Figure 16. The certificate life cycle

Certificate expiration is inevitable. The certificate lifetime needs to be described in the security policies. The security policies will have to detail the type or class of the certificate being used. For example, as previously mentioned, a certificate issued for personal use may have a different lifetime than a server certificate used for Web server authentication. The security policy needs to describe under what conditions the CA (or RA) can intervene in the normal life cycle of a certificate. A CA will have different tools for managing the lifetime of a certificate. For instance, Figure 16 shows the certificate revocation and certificate suspension management tools available to a CA. An employee leaving an organization will need to have his or her certificate revoked, since that certificate no longer represents that user correctly. When an employee forgets the password to access his or her private key, or loses the private key altogether, the corresponding certificate may have to be revoked, depending on the organization's implementation of the PKI. An organization will also have to determine under what circumstances a certificate life cycle should be put on hold. For example, an organization may want to suspend a certificate issued to an employee who is under investigation, or who is on leave. Resumption of the certificate validity can continue once an employee is cleared from investigation or has returned from leave. An organization may want to issue certificates with different lifetimes to its employees depending on their status. For example, contractors to an organization could have a certificate valid for the period of their scheduled contract work, whereas a permanent employee may have a certificate renewable every 12 months.

An organization wishing to deploy a PKI has to consider the type of users that will be using the certificates, the application that the certificates will be used for, and the business requirements of the organization. Details of certificate usage need to be detailed in a certificate policy and Certificate Practice Statement (CPS). This will form part of the organization's IT security policy.

4.1.5.2 Keys

As a general rule, a key pair can remain valid for a long period of time, normally much longer than a certificate. However, there are some reasons to limit the lifetime of key pairs, such as:

- The longer a key is used, the greater the chance that it could be compromised. This is especially true for keys that are used for encryption of highly sensitive data, such as government secrets. As a matter of fact, in theory, the more encrypted samples a potential attacker has, the easier it is to compute the key.
- If the key is compromised, the potential loss becomes greater the longer a key was used. If an encryption key was used for a extended period of time, a large number of documents were possibly encrypted, and thus, a large number of documents is at risk of being disclosed to an attacker.
- Technology advances may render a key less secure in the future than what it was considered when it was generated. A DES 56-bit encryption key, for example, was considered very secure when DES was adopted in 1976. However, recently, it has been demonstrated that DES may not be secure enough against more advanced technology.

While it is relatively easy to have a certificate resigned with a new expiration date, a key pair update may involve much more. Depending on security policies, a new key pair may require that all previously encrypted documents must be decrypted and encrypted again with the new keys. Any documents signed with the old, expired private key may need to be signed again with the new key. This can have a large impact when key pairs are being updated.

Alternatively, an organization could request a user to keep his or her old private key secure in order to verify documents that were signed during that period, and to use his or her new private key to sign new documents. Additional key management policies will be required in such a situation.

For these reasons, a key pair could remain valid for a period of typically five years or even more, while some critical key pairs could have a lifetime of as little as a few months or even less. Security policies need to specify what actions are to be taken when key pairs are replaced.

Some organizations may even choose to not specify a specific key lifetime but replace keys as it becomes necessary, for example, due to a loss of a private key. In such cases, it is good practice to reevaluate the level of security with the key pairs being used after, for example, five years or after new algorithms or other security issues or technical advances become available or known.

A key's life cycle can also depend on whether an organization implements key recovery facilities or not. An organizational security policy needs to specify under what circumstances a key can be recovered. It has to be defined who will be responsible for the key facility and what type of key pairs will be recovered. Key recovery, just like key escrow (to be explained later), is a sensitive topic to be considered and implemented very carefully, since it may be misused to compromise security.

No matter for how long keys are being used, it must be pointed out that security policies must include rules on how to handle expired keys. Though the keys might have been compromised, there might still be many documents and signatures around that can only be decrypted or verified with the expired keys. Thus, even if they are expired, it might be wise to keep a copy of these keys in an archive. Refer to section 4.1.8, "Key management, storage, and archiving" on page 75.

4.1.5.3 Example scenarios

In order to get a practical picture of a PKI system as it relates to key and certificate lifetimes, examples are provided below with the following certificate policy assumed:

- The private key lifetime is defined as 10 years.
- A digital signature is valid for 25 years, since it is signed.
- A certificate must be renewed every year.

In the first example illustrated in Figure 17 on page 71, a private key is used for the digital signature in a business contract.

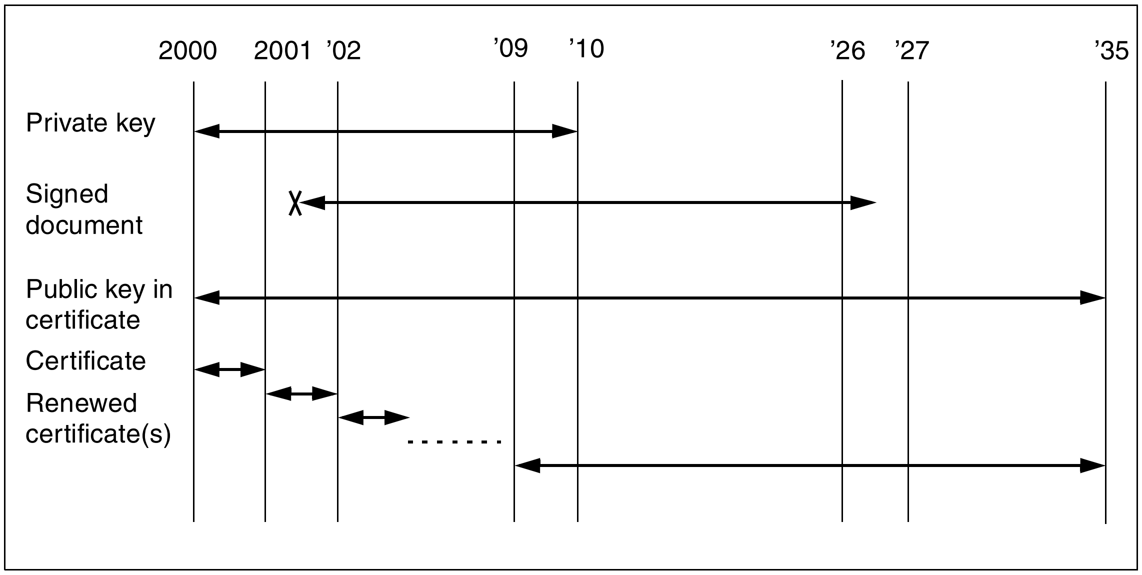


Figure 17. Example 1: Digital signature

Since the private key's lifetime is 10 years and it was created at the beginning of year 2000, the key must be kept at least until the beginning of 2010. Assuming a document is signed in the middle of 2001 (marked with an X in Figure 17), it will be valid until the middle of 2026, because its validity duration is defined as 25 years. As it can be seen in Figure 17, the signature remains valid beyond the lifetime of the private key used to create that signature.

The public key should, therefore, be kept longer than the private key, because it is used for verification of the digital signature even after the private key's lifetime ends. This fact is hard to circumvent, considering the following: there is a chance that another document will be signed at the end of 2009, just before the private key expires. Therefore, the public key should even be kept at least until 2035.

There are other solutions to this. A trusted system designed to work like a notary system could certify the validity of a digital signature at a given time by signing it with its own signature. From then on, only the notary system's public key is required, which could be imbedded in a long-living certificate. The original signer's private or public key is not required any more since the notary system has, at a certain time, certified that the signature was valid.

Back to the example shown in Figure 17—it is shown that the certificate is renewed every year. Each renewed certificate replaces the former one. At the beginning of 2009, the last certificate (10th certificate) for the public key will be issued, which should be kept until the beginning of 2035. In this example, the last certificate's expiration would be set to the beginning of 2035 because it might be required to validate a digital signature up to 25 years after it has been placed on an electronic document. Between 2010 and 2035, the private key cannot be compromised any more if safely destroyed or stored, and thus, a longer certificate expiration period is less of a concern.

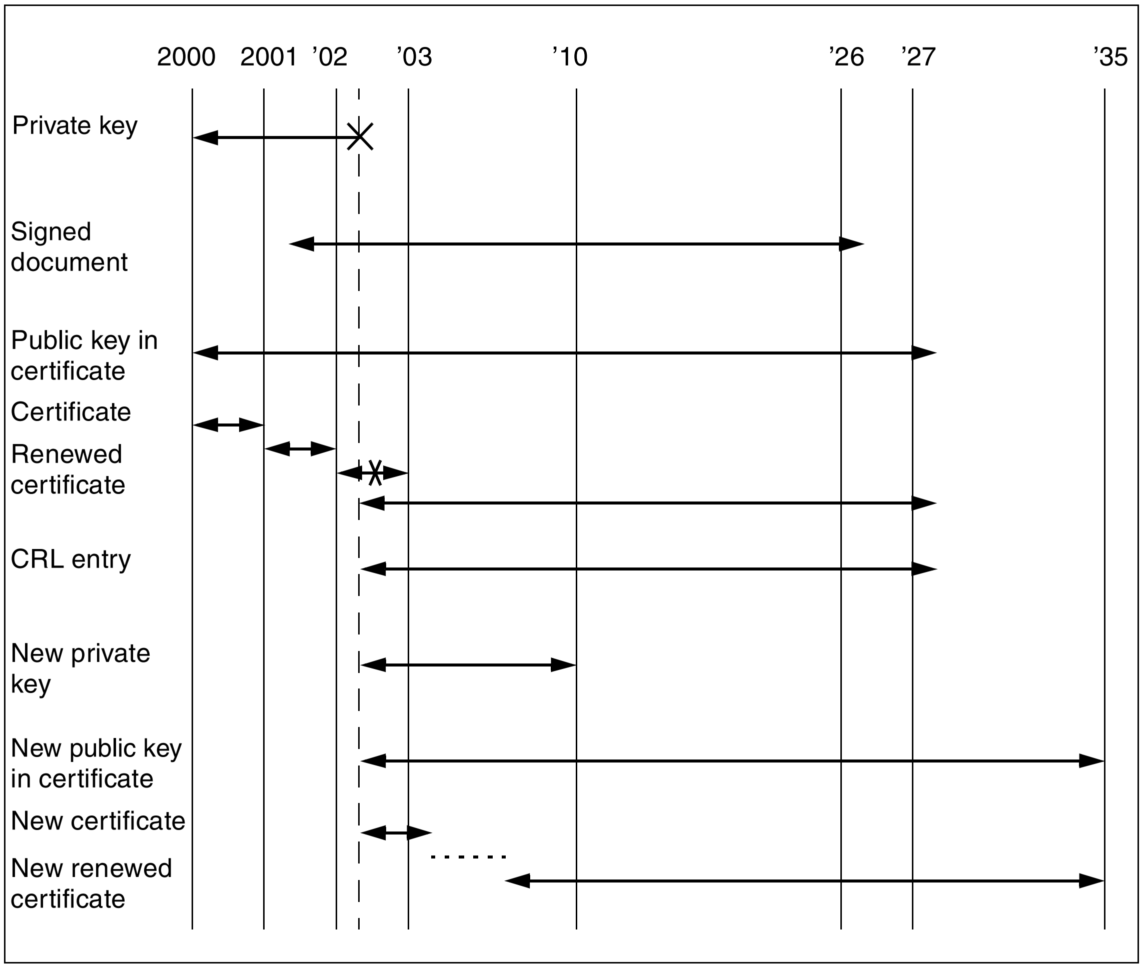


Figure 18. Example 2: Digital signature with compromised private key

Figure 18 depicts another example that is more complicated than the previous case. The assumption in this example is that the private key is being compromised in early 2002.

The private key is no longer valid and the CA puts the certificate on the CRL. But in many cases, and depending on the security policies, the signed document, which was signed before the private key was compromised, can still be trusted by the verification process with the (old) public key. It is up to the security policies and its implementation whether the last, now revoked, certificate can still be used for such validation purposes, or if a new certificate with a long-lasting lifetime will have to be created for that purpose.

If the last document was signed in early 2002, the public key should be available until early 2027 (2002 + 25). So, a certificate has to remain available through early 2027 although it possibly is listed in the CRL.

The signature validation algorithm will have to understand whether the digital signature was created before the certificate's revocation or not.

A new key pair will be generated and the new corresponding certificate will be issued for the subsequent signing process. The serial number of the new certificate should be different from the original. The subject name in the new certificate should be identical to the original.

4.1.6 Handling guidelines

The policies that an organization defines and implements will determine how the PKI operates within that organization. Therefore, the policies act as prerequisites to many PKI activities and implementation details. Policies also define how an organization interacts with the outside world. Therefore, it is important to look at some policy tools available for a PKI and what factors to consider when constructing one.

Generally, there are two types of PKI policy tools:

- *A certificate policy*. A CA certificate policy determines *what* the certificate will be used for.
- *A Certificate Practice Statement* (CPS). A CA CPS is a description of *how* the certificate policy will be implemented in a specific organization.

A certificate policy is defined by X.509 as a *named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements*. A certificate policy will be used by a user in deciding whether to trust a certificate for a specific purpose or not. For example, a certificate policy of your organization may require that the certificate issued to you may only be used for authentication on your organization’s intranet. Or, a CA policy can issue certificates specifically for online shopping. The certificate policy must be recognized by the user of the certificate and the issuer (CA). There also must be a clear method for identifying the certificate policy. Standards bodies (X.509) have provided such a method, using object identifiers (OIDs) and policy fields in the certificate template, so a CAs policy can be identified. A CAs certificate policy also forms the basis for any CA-to-CA trust relationship. CAs considering cross-certification can use each other’s certificate policies as a starting point in defining their relationship.

A Certificate Practice Statement (CPS) is defined by the American Bar Association (ABA) as “A statement of the practices which a certification authority employs in issuing certificates.” This is a document that explains what exactly is being certified and what degree of trust can be placed in the certifying party (CA). It allows the user of a certificate to make a decision about whether to trust the certificate issued by that CA or not. A list of topics that need to be covered in a certificate policy and certificate practice statement can be found in RFC 2527 (see also 5.2, “Internet RFCs” on page 102).

It is a good idea for an organization intending to deploy a PKI to have a predefined PKI strategy in place. This strategy will drive the PKI policies required by that organization. An organization’s security policy should provide details on how a PKI will be designed, constructed, operated, utilized, and managed. An organizations security policy should contain a certificate policy and a Certificate Practice Statement (CPS) as previously mentioned. The deliverable detailing the policies will be known as the PKI security policy recommendations document.

Finally, it is important to remember that a PKI implementation can also be a privately managed system. For example, the Secure Electronic Transaction (SET) protocols have their own predefined set of rules describing the services required by each party. In such a situation, a CPS will not be required, since the rules of engagement have already been defined by the protocol.

4.1.7 Centralized or individual responsibilities

The components introduced in Chapter 2, “PKI overview” on page 21, allow for an organization wishing to deploy a PKI to determine to a large extent the distribution of responsibilities across their environment. For example, when deploying a PKI, a security policy will have to address where key generation will be performed. It is possible for key generation to be performed at the End-Entity, CA, or RA. If an organization intends to generate keys centrally, it will have to find a secure means of transporting those keys to other components of the PKI. A CA will be responsible for publishing CRLs to a repository. Certificates can be published by an RA, CA, or end user to a certificate repository. Policy guidelines will have to detail the use of such repositories. For example, there can be one central directory service providing certificates to PKI users in that security domain, or there can be multiple distribution points.

For practical reasons, an organization may want to separate authentication services from certificate management services. In such a situation, the organization could request an external party to supply certificates. The organization could then act as an RA, authenticating employees. The requests could then be passed on requests to the third-party CA, while at the same time, maintaining the privacy of its employee database. In this way, the organization can distribute PKI activities and minimize administration activities for itself. The organization could also request the third party CA to issue certificates directly to the end users who requested them. Any management activities, such as key updates or certificate renewals, could be handled between the third-party CA and the end user directly. Requests, such as certificate revocation requests and loss of private key, could be processed via the employee's organization in order to provide authentication.

4.1.8 Key management, storage, and archiving

Key management is an essential aspect for the successful deployment of a PKI. Key management may be difficult because it includes each individual participating in the PKI. In a large organization with potentially thousands of users, this can be demanding. Key management can be divided into three areas:

- Generation and distribution of keys
- Usage, updating, and destruction of keys
- Storage, backup/recovery, and archiving of keys

Key management also includes *key escrow* and *key recovery*, which are discussed later in this chapter. The three primary key management areas above are described in the following sections.

4.1.8.1 Key generation and distribution

To achieve a high level of security, it is required that the key generation procedure is good enough. A full discussion of key generation techniques would be beyond the scope of this book. However, there are a number of issues that need to be kept in mind. A public and private key pair generation uses mathematical calculations along with some random number generators.

A “good” key is a key that is based on a high quality random number generator. An organization concerned about key-generation procedures can consider hardware-based, pseudo-random number generators that allow key generation within a secure enclosure.

Then, an organization may want to consider the key length for its symmetric and public key encryption algorithms. The key length chosen by an organization depends on the type of application used and the computing power available now and in the future (which may be impossible to predict). If computing power doubles every 18 months, it might be necessary to be conservative and choose longer keys. Common lengths for keys used in a PKI are 512, 768, and 1024 bits, sometimes also called *low*, *medium*, and *high grade* keys. While writing this book, it has been proven in at least one example by a team of scientists that 512 bit keys can be broken, but only with a considerable amount of computing power that is not (yet) easily available to most organizations. However, if strong security is a requirement, medium or high grade keys should be selected. Low grade keys have the advantage of consuming less computing power (thus working faster) and might still be good enough for several years for less sensitive information, such as e-mail.

Finally, an organization has to consider whether key pairs are being created by the users (End-Entities) or by a central authority. In most cases, key pairs will be created by the End-Entities because of the additional level of security. If keys are to be created centrally, key distribution becomes a critical issue. Central generation may have the advantage of better control over the algorithms used and to enforce other relevant standards.

If an organization intends to generate a private key for its users, it needs a safe way to distribute the key to the owner. It must also be ensured that no unauthorized copies can be made at the place of generation. In a small organization, there could be a requirement for users to physically collect their key or key parts. Then, the organization would have to provide a procedure for users to securely enter their private key or key parts into their computer or Smartcard. An organization will need to define how it intends to publish certificates.

4.1.8.2 Key usage, update, and destruction

As mentioned in 4.1.5.2, “Keys” on page 69, key pairs have a limited lifetime. An organization’s security policies have to define how often its users will require new key pairs. Certain key pairs may be used for more sensitive applications and so may require more frequent updates. Typically, an organization should stagger key updates over time, allowing users to hold certificates with different expiry dates, thus avoiding possible problems occurring when a large number of keys expire at the same date. An organization's security policies should define the processes for key updates and recertification.

Once a key pair is replaced, the security policies should include how the old keys will be destroyed. Even though an old key may have expired, it could still be used to decrypt information. In addition, a user's private key is usually stored on disk, where it can be copied to multiple locations. Also, if software cryptography is involved (for example, a Web browser), the operating system could write a private key to disk while swapping applications in and out of memory. There is no way of knowing whether the operating system successfully erased the private key from the disk during its memory management operations or not. This has to be taken into account when applications are being developed.

In 4.4.2, "Smartcards and readers" on page 87, the possibility of using Smartcards for storing a user's private key was discussed. There are many ways that an organization can store cryptographic keys, and it will eventually depend on the implementation of the specific organization. Once again, the procedures of how and where to store keys will need to be outlined in the relevant security policy. In addition to Smartcards (security tokens), keys can be stored on a magnetic stripe card, or a *ROM key* (a plastic key with an embedded ROM chip). Some implementations divide a key into two parts, one could be stored in a ROM key and the other in the workstation. This lessens the chance for an attacker to get unauthorized access to the full key. A loss of one of the key parts does not compromise the key itself, and so a new key can be regenerated. Finally, keys will need to be stored in an encrypted format. For example, a key could be encrypted using a user-defined password, or it could be encrypted with a master key that is securely stored in a cryptographic hardware module.

4.1.8.3 Key storage, backup, and archiving

Key backup and recovery procedures are very important aspects of key management. In an extreme situation, if a CAs signing key accidentally gets lost without a backup, the CA could not sign any CRL, CR, or recertify certificates. In other words, the whole security domain is basically lost. Disaster recovery and backup processes and procedures need to be covered in an organization's security policy. The following items should be covered in a backup and recovery security policy: audit logs, archive material, CA private key(s), RA private key(s), End-Entity's private key(s), and the personnel responsible for the backup and recovery procedures. The format in which the backups will be made (encrypted, plain text, or key parts) must be defined, too. A correctly implemented backup and recovery policy is essential in order to avoid the consequences of a key compromise situation.

A key or other information may be archived by an organization in order to be used at a later date. Whatever an organization archived, there has to be a policy detailing the archiving procedures. Issues to consider when drawing a policy for archiving are the following:

- Objects to be archived, since an organization may not only want to archive public keys.
- Audit logs detailing specific events, such as certificate requests, key compromise notification, and so, on may be helpful to dispute resolutions later on.
- Procedures detailing time-stamping, the retention period of archives, and who will be responsible for the archive needs to be documented.
- In addition, an organization's policy has to address issues, such as who will be allowed to view the archive and how archive information will be retrieved.
- Finally, because of the nature of the material that will be archived, procedures need to address how the archive will be protected against tampering.

It must be pointed out that any archiving or backup process is essentially a copy operation. Policies must address the issue that the same strict rules must be applied to any copy of the data as are incurred to the original. Bear in mind that a copy operation may include a data transfer over a network where the data could be eavesdropped during that operation.

4.1.9 Legal considerations

Although the particular details are beyond the scope of this book, it needs to be mentioned that an increasing number of countries have (or are in the process of introducing) some legal regulations on the use of encryption, digital signatures, and conducting business using these electronic means. For example, the use of digital signatures, instead of handwritten signatures, may be legally accepted in some countries. In any case, it can be expected that government regulations will be in place in the future that will affect an organization's PKI deployment. For example, under current French law, any electronic transaction over FFr 5000 requires a signature on paper. However, in September 1999, the French government drafted legislation that would legitimize digital signatures on Internet-based financial documents.

4.2 Defining a deployment goal

If an organization desires a PKI deployment solution, it has to set up and define some deployment goals so as to fulfil the security and business needs of the organization. Once the goals and objectives are set up, every activity or task has to be tied up with them. Some examples of business needs may be: an organization wants to start an e-business service by hosting a machine on the Internet that uses PKI for user authentication; a corporate strategy tries to standardize and improve user authentication in all systems; a particular department or application needs to use PKI for its specific business needs; or a business-to-business transaction, which needs PKI for formal business communications, and so on. On the other hand, organizations may want to deploy PKI due to their security needs, for example: an organization wants to improve the overall security level, especially for systems connecting to the Internet; the government or some authorities has announced some security guidelines for companies to follow; or an organization has to fulfil its customers' high security expectation on its system.

Therefore, an organization has to understand its needs thoroughly and then define the deployment goals and objectives accordingly, no matter whether the needs are classified as business or security. Goals can be defined from several perspectives, but please bear in mind the business and security needs all the time.

4.2.1 Define the deployment scope

PKI deployment is in fact a large-scale project and its scope has to be clearly defined. Geographically, if the PKI is deployed worldwide, which countries or geographies should be covered and who should be the prime facilitating geography? Is this PKI used by and within one organization, or it is for business-to-business purposes? Is this PKI going to interoperate with other established PKIs? Will there be any interoperability issues? Should this be allowed and accepted? Is the deployment intended for the whole organization, or just for a few departments and business functions? These factors determine what parties will be involved in the deployment process and who is the driving party. For parties not inside the scope, they may not need PKI, or they may have PKI at a later stage or in the next phase.

From an application point-of-view, is it necessary to enable all applications to use PKI, or only some? If applications have to be rewritten or modified so as to utilize the PKI, what will be the scope of change? Will new applications or systems be introduced and what is the scope? For existing legacy systems or systems on other platforms, can they use the PKI as well? If so, what has to be done? Is it necessary to standardize the security control on all platforms?

These questions may help an organization to evaluate the efforts needed for the PKI deployment and to confine the scope.

A PKI may come in many flavors. For example, a government may want to set up a country-wide PKI so that every citizen is an End-Entity and all its interfaces with the government will be made through the certificates or digital signatures. A company may set up a PKI so that its staff can use their staff badges (which are Smartcards) to access rooms and log on to computer systems and applications. An online shopping mall may set up a PKI for customers to buy things using some prerequested certificates to represent their identities. A bank may set up a PKI for corporate customers to use its corporate Internet banking system for functions, such as remittence and letter of credit (LC). From the examples above, it is found that the End-Entities (or users) of a PKI have to be carefully defined. The users may be citizens, customers, I/T staff, internal staff, business partners, or other companies. Individuals or organizations not defined in the scope should not be included in the security domain of the PKI.

A well-defined scope is a prerequisite for a successful PKI design and deployment. The scope may (and most likely will) be reconsidered later on as new types of users have to be included.

4.2.2 Define the security expectations and approaches

The discussion of PKI deployment cannot be completely separated from the technology it uses. The advancement of technology is closely related to the security level of the PKI. Just to give an example, people trust a certain cryptographic algorithm because it is almost impossible for existing computers to guess or calculate the keys. However, as computing power keeps on growing, some secure algorithms today may be cracked easily in the foreseeable future. Therefore, with the state of the art in technology, organizations have to define their expected level of security. There are many cryptographic techniques or algorithms available, and organizations have to choose the ones that suit their needs.

As introduced in 1.1, “Recap: Security basics” on page 3, PKI provides five security capabilities. Some organizations may emphasize the user authentication part. If this is the case, they need to decide how and when to authenticate an identity. In general, certificates will be used for user authentication. Organizations may have to consider whether to put the certificates on Smartcards, or some other key storage media. If their emphasis is on data confidentiality, organizations may just choose a cryptographic algorithm for the encryption and decryption of data. If the emphasis is on data integrity, digital signatures may be used. If non-repudiation has to be enforced, certificates with public key cryptography may be used. The above discussion is much simplified. Since different customers have different security expectations and applications, the final security solution may be a combination of several techniques and cryptographic algorithms.

Another important thing for consideration is when to implement what. That is, will all the security controls be introduced at once, or will they be introduced phase-by-phase, or will only some minimal controls be implemented at the beginning? This has to be planned and defined so that all involved parties can get a feeling of how the project is to be run and how large the scale will be. This can set different parties' expectations of this PKI deployment. Resources can also be planned to align with the whole deployment process.

Therefore, by putting the security expectations and approaches into the deployment goals, this ensures the security level will be met and resources can be committed for the project.

4.3 Defining a deployment road map

With the deployment goals in mind, organizations can start their planning of the deployment process. Up until this point, readers may realize that the amount of work and planning involved is not that simple. So, by defining a deployment road map, the deployment tasks can be listed in a logical and organized manner.

A road map is a global picture of the deployment over time. The picture can be composed of the following elements: time frame, milestones, and deployment phases. In fact, these are also the basic elements in every project plan. A time frame can be defined by specifying the start date, end date, and the length of the project. If the project takes long time, for example, several years, then some major milestones have to be identified so that the project is trackable. In general, a PKI deployment process may have the following high-level milestones:

- Solution workshop
- Readiness assessment
- Deployment planning
- Prototyping
- Pilot run
- Production

A milestone is usually at the beginning or the end of a phase. Milestones help executives, project managers, and users to track the progress of the deployment. A milestone is a intermediate goal for respective parties to complete their work. Usually, financial planning is also closely related to the milestones. This helps the financial department to budget enough funds at different stages of the project. As a rule, a delayed project will incur extra costs and resources. When planning the milestones, it is important to think about process dependencies.

Therefore, a good road map serves to be an overview of the project. It also assists different parties to perform their part of the whole work and this helps to keep the project costs under control. The following subsections align some tasks with each deployment phase.

4.3.1 Pre-engagement

An organization may know about PKI through exhibitions or seminars organized by vendors. Afterwards, a vendor may organize individual workshops for the organization, which is a two-way communication between the vendor and the organization. The vendor may describe its PKI solutions and offerings. The organization may explore the opportunities and risks brought by a PKI. The vendor will also guide the organization to think about its strategy and business needs, and how the PKI solution brings benefits. This is called the pre-engagement stage because the organization has not decided which vendor to choose from (if at all), nor have any details been defined.

4.3.2 Assessment

Now, the organization is in a position of having some interests in a PKI deployment. Before deciding to deploy PKI, an organization must assess its own status at the time. First, will the PKI bring value to the existing business? Some values that PKI may bring are strong security, automated processes, centralized security control, and in particular for the case of VPN, PKI helps to save money. What will be the initial cost and operating cost? Is it worth it to implement a PKI if the cost seems high? If the cost is not justified, it is hard to persuade the executives to support such a deployment. Organizations also have to check their budget for the few coming years. Does the budget allow such a large-scale deployment or does it need to be readjusted? If not, organizations have to source some funds. During this process, not only the business executives or technical people are involved, the financial department should also be invited to the discussion. If everything is ready, this brings us to the next step: planning.

4.3.3 Planning

The planning phase may take a long time, since many perspectives have to be considered. Organizations need to define their business and security requirements. As introduced in 4.1, “Defining security policies” on page 59, the PKI policies have to be defined during the planning stage. The site and facilities for hosting servers and for development have to be considered. What will the PKI architecture be like? The components, protocols, services, and standards have to be chosen and defined. Some organizations may even consider outsourcing the deployment or operation of the PKI to some other security companies (which will also be discussed in 4.5, “Management and administration” on page 89). Since the PKI will cause some changes to the existing organization, new security processes may be introduced and re-engineering of some business processes may be required. A good planning is a good start for the whole project.

4.3.4 Prototyping, customization, and application development

A PKI may affect the whole organization. However, not every party in the organization can understand the concept of a PKI and how will it affect the existing systems. Therefore, a prototype environment may be needed as a *proof-of-concept*. Before setting up and developing the prototype, the organization has to investigate the available PKI products and choose one that is suitable. The criteria of choice may be based on the compliance of the product to PKI standards, comprehensiveness of the product, price, scalability, support, integration with other security solutions, ease-of-use, and so on. The prototype can be implemented by using the chosen product or a small selection of products. At the same time, the organization may start to compile the implementation plan, which is not only restricted to the product and applications, but the overall implementation tasks for that organization.

Once the prototype is accepted by the involved parties, application design and development can start. Here, application means several things, including the applications that need to be rewritten or modified so that they can use the PKI, some product customization code for the organization’s specific needs, and other new applications that may be needed to support the PKI. The development team should work closely with the users of the system to get their requirements. Some certificate practices have to be decided upon and these will affect how the applications need to be developed and how the products are to be customized.

4.3.5 Pilot run

When the applications and products are ready for use, including the successful completion of all unit tests, system integration tests, and user acceptance tests, a pilot test run will most likely be the next step. A pilot run usually means putting the system to a real-life, but small-scale environment for running over a certain period of time. Organizations may choose a few branches, departments, or a group of people to test run the system. This pilot run should not affect the existing daily operation of those pilot users. Organizations may consider having the old and new systems running in parallel so that if the pilot run is not satisfactory, they can go back to use the original system. This will make sure that the existing system availability and service level are not affected.

4.3.6 Production

If everything goes fine during the pilot run, organizations can proceed to launch the system for production purposes. Before production, it is a good idea to have the system assessed by different parties, including the end users. The auditing department may want to assess the security level of the system and check if all security processes have been carried out. Otherwise, if the system is not secure but put to production, this will expose the whole organization's security to unnecessary risks. The legal department may need to verify if the system adheres to any government legislation. Also, as many license agreement, policies, terms, and conditions are defined in the system, legal expertise is required to check and finalize all these. The CPS described in 2.3.6, "Policies and processes" on page 38 and some business-to-business contracts have to be established. These will bind the users of the PKI concerning the usage of certificates, digital signatures, or levels of encryption. If the system is to be rolled out to more than one country, the legislation of all other countries has to be checked as well.

Similar to the pilot run phase, some parallel-run and fallback planning may be needed. Also, further planning is needed on how to roll out to the entire organization. To have a smooth production start, *education* should be offered to all users beforehand. Also, the organization should set up a *help desk* to answer general queries from the users. Some staff should be trained to become technical support staff, and they are responsible for problem determination and escalation whenever problems appear.

These are some of the considerations that an organization may need to think of. Of course, different organizations may base on their own application introduction life-cycle policies and procedures for the production or their individual application introduction.

4.3.7 Scalability considerations

A PKI has the potential of affecting many resources of an organization, including servers, applications, and employees, in one way or another. A PKI may also be desired to securely conduct business with an unknown number of clients outside the organization.

In either case, scalability of the overall PKI solution becomes an important consideration that needs to be designed into the deployment and roll out process. Exact machine sizing and network load figures and tools are not (and will not be) available with new products, and even if they were, they can only predict the behavior of a solution in a very specific reference environment. Most, if not all practical environments differ in their specific implementation.

Because of the above, this book cannot give sizing guidelines and calculation methods. However, the following are the most important points to consider in terms of scalability:

Key pair generation – Key pair generation requires a significant amount of computing power. If this is to be done centrally, say in the machine that represents the CA, the corresponding computer will only be able to calculate a limited number of key pairs in a given time interval. In addition, the CA will have to sign the certificates and do some additional processing, such as cryptographic processing and storing data in a local database. For this and the reason of confidentiality, it is a good idea to create key pairs on the clients and let the CA sign and process the certificates.

Certificate directory – The importance of a generally accessible directory for storing certificates cannot be stressed enough. Centralized, universal directories based on LDAP are being deployed throughout most organizations and certificates are just one of the objects served by such directory services. Because they are universal, such directories can easily become large in size and heavily utilized, and thus slow in performance. In addition, the growth rate might be difficult to anticipate in the beginning. The IBM SecureWay Directory product provides special means for scalability. These include:

- Support for *different hardware and operating system platforms*, from small Windows NT servers up to System/390 mainframes.
- Support for *split namespaces* for manageability and scalability purposes. A single server then only manages a part of the whole directory namespace.
- Support for *replication*. Replicated directory servers share their workload and increase availability.

In order to decrease directory workload, applications might cache certificates and other information. While caching is generally a very effective way to improve performance and scalability, there has to be policies (and yet other methods) in place that define how long certificates in a cache may be considered valid.

Certificate Revocation Lists – CRLs are critical in PKI exploitation. Just as with certificates in a directory, an application must have a means to check whether or not a certificate has been revoked. The same basic considerations apply as for certificate directories, but due to the fact that CRLs are more critical, online verification might be desirable. This, of course, rises the performance requirements of the underlaying directory service significantly. If online verification is to be done with the CA, the CA system(s) must be able to handle the additional load.

Lifetime management – Although it might be desirable to introduce and deploy a PKI at a given date in time, it can be disastrous for an organization when all the certificates and/or keys expire on the same day. It is, therefore, highly desirable to generate and renew keys and certificates on an ongoing basis rather than corporate-wide fixed dates. This may not be possible in certain environments, such as universities, for example, that have a high enrollment demand only for a relatively short period of time. Daily peak hours are another point to consider. Long-term storage of keys and certificates may not contribute much to performance bottlenecks, but needs to be taken into account for database and system sizing.

4.4 The infrastructure

The success of a PKI deployment not only relies on a PKI product, but also on the surrounding and supporting infrastructure. Here, the word *infrastructure* means the basic facilities, equipment, and features needed for the functioning of the PKI.

4.4.1 Server machines and cryptographic hardware

As discussed in 2.1, “The core PKI components” on page 21, a PKI usually has a CA, CR, RA, EE, and digital certificates as its core components. Among these five components, only the CA, CR, and RA are physical server machines, and among these three, the CR and RA are optional. So, organizations have to decide whether to put all three components on the same server machine, or separate the workload to different server machines. This separation of tasks may not solely be desirable due to performance considerations, but it helps organizations segregate the authorities among several personnel or departments. When organizations have to choose what server machines to install, there is always a balance between performance and cost. In general, PKI-related servers demand high performance, speed, system resources, and availability. Also, certain sizing on the CPU usage, disk space usage (or data volume growth), and memory usage should be estimated. In general, it is better to choose hardware with a scalable design, such as an SMP system, since the PKI will have foreseeable growth in usages every year.

The CAs private key is one of the most important parts in the whole PKI. It is used, for example, by the CA to sign every issued certificate. Thus, it is especially subject to attacks from hackers (may be internal personnel or Internet users). Therefore, it is most important to protect this key. This key can be encrypted for storage. However, whenever the CA has to sign a certificate, this key has to be decrypted back to its clear form for signing, so a more secure storage method is needed. It might be considered to use a hardware cryptographic card for the key storage. Taking the IBM SecureWay 4758 PCI Cryptographic Coprocessor as an example, it is a special hardware that can store encrypted keys within the card. This on-board coprocessor can handle extensive encryption and decryption. The hardware is enclosed, tamper-sensing, tamper-responding, and thus can provide a high degree of safety. It has also been certified under the Federal Information Processing Standard (FIPS) 140-1 at Level 4, that is, the first product to achieve this level of certification. For more information, please refer to:

`http://www.ibm.com/security/cryptocards`

4.4.2 Smartcards and readers

In the deployment, each End-Entity will have at least one of its own private key and certificate. These should be kept safely, since they represent that End-Entity. If the key and/or certificate is lost or accidentally disclosed, others can fake this End-Entity and do some unauthorized operations. Smartcards, or security tokens, are portable cryptographic devices that can be used to store keys and certificates. The chips on the card can perform cryptographic operations, such as signing without disclosing the private key from the card. Therefore, Smartcards may be deployed to store End-Entities' keys and certificates. Its portability allows the card to be used at personal computers, network computers, kiosks, access badge readers, and so on, and thus aligning with the applications that use PKI. Organizations may need to investigate the feasibility of using Smartcards in the deployment. There are many Smartcard products on the market with different features. Organizations should choose one that suits them most by considering the functionality, standards-adherence, cost, and other factors.

Most current computers do not support Smartcard reading natively. Smartcard readers are extra peripheral devices adding on to the computers. Over time, it can be expected that more computer manufacturers are going to include the Smartcard reader as a standard hardware feature to their computers (both desktop workstations and notebook computers). In addition, some Smartcard readers may be integrated into keyboards. The latter has the advantage, if properly designed, that a user's password to access the Smartcard cannot be intercepted on the keyboard cord because it is not transmitted over that cable to the computer unit. The above discussion suggests that organizations have several options to choose from. Add-on devices may be cheaper than replacing existing hardware. User friendliness also has to be considered.

Some PKI products also allow users to use *virtual Smartcards* for storing keys and certificates. Virtual Smartcards are files stored on a local disk with some application logic that behaves like physical Smartcards. This convenient feature allows users without a Smartcard reader to access the PKI. However, the users need to handle the virtual Smartcards more carefully and should only keep one copy of the card.

4.4.3 Physical environment

The physical environment of a system is also considered a part of the infrastructure. If the servers are kept in an open room, no matter how secure the applications are designed or how complex the cryptographic algorithms are, the server services can easily be interrupted, such as being powered off, or otherwise tampered with. Therefore, physical access should be controlled and it is part of the security policies that need to be defined.

The CA servers should be located in a robust, dedicated, and locked room. All accesses should be logged and controlled so that only CA-related personnel can go in. The power supply to the servers should never be interrupted. This means an uninterruptable power supply (UPS) must be used. However, a UPS may still run out of electricity after a prolonged period. In such a case, the servers should be able to automatically backup the data and properly shutdown. The room's entrance can also be monitored to check who has accessed the room.

For maximum security, the network segment where the PKI-sensitive server machines are installed should be physically and logically separated from the rest of the network. Ideally, the separation is done through a firewall that is transparent only for PKI-related traffic. Normally, PKI traffic is reduced to using only a few TCP/IP ports.

4.4.4 Skills required

A solution without skilled people to implement, maintain, and support it is not useful at all. The involved personnel can be considered part of the infrastructure. In 4.5.1, “Responsibilities (roles)” on page 89, some typical roles are defined and they are responsible for supporting the whole PKI. This section lists some skills that are expected or have to be acquired by the personnel. The I/T department may need these skills: product installation, configuration, system administration, PKI theory and practice, public key cryptography, and I/T security. The system support or operations department may need skills, such as problem determination and management, basic knowledge in PKI theory, and usages of the system. An authorization department may need to know PKI concepts and system administration. An audit department may have to take care of the policies, liabilities, legal aspects, and understand I/T security. Formal training and education classes can also be set up as a part of the infrastructure to facilitate and accelerate personnel skills acquisition.

4.5 Management and administration

This section provides an overview of certain issues that need to considered for the successful operation of a PKI. It also provides a list of responsibilities and skills that will be required from various security personnel during the different phases of the PKI deployment. This section is intended only as a guideline and some of the items listed here may not be relevant to your organization.

4.5.1 Responsibilities (roles)

The procedures for each administrative and management role in a PKI must be documented clearly. If necessary, the correct training should also be provided to security personnel. Below is a list of possible roles that will be required in an organization implementing a PKI. Depending on the specific environment, one person may assume more than one role. For example, the CA administrator can also be responsible for key generation and revocation requests. The following list provides a guideline of what roles may be required:

- System administrator
- System operator
- CA administrator
- RA administrator
- Directory administrator

- Help desk professional
- Policy manager
- Security auditor or supervisor

The *system administrator* oversees the overall running of the security system. The system administrator should be involved early during the PKI deployment phases. Typically, a system administrator will participate during the planning phase and will generally be involved during the implementation phases. Refer to 4.2, “Defining a deployment goal” on page 79, for a discussion of the various phases. The system administrator will be especially valuable during the planning phase since he or she will be able to provide time estimates of various implementation activities, which will affect the overall planning of the project. If an organization is planning on operating its own CA, the system administrator will be responsible for the planning, installation, and configuration of the necessary software. Any updated software components that need to be installed and configured for the PKI will be done by the system administrator. The existing software components are also maintained by the system administrator. Assigning roles and profiles to the users of the system will be another task required by the administrator. Password maintenance will be one of the day-to-day operations required from this role. Users often forget their passwords. The system administrator will have to follow the processes and procedures laid down by the organization to reissue passwords.

Basically, *operators* will be overseeing the operation of the PKI. They must be able to respond to error conditions and must be able to follow documented procedures. Depending on the size of the PKI, one or more system operators will be required for daily operations. Additional tasks may include backing up and maintaining the DB2 tables and maintaining the documentation relevant to the PKI.

The *CA administrator* is responsible for all tasks related to the Certificate Authority and certificate generation. The CA administrator has privileges to generate keys and sign certificates. The CA administrator also has privileges that allow him or her to authorize or reject cross-certification requests. For instance, before a cross-certification is processed, the CA administrator may be required to enter a PIN or password. The CA administrator will also be responsible for authorizing any key recovery services that may be required.

An *RA administrator* will be required if an organization implements a PKI with the optional RA component. The RA administrator is a person authorized to run the registration authority utility. This activity requires processing requests for certificates and maintaining certificates that have already been issued.

The RA administrator has the responsibility to evaluate a user's certificate request and has privileges to approve or reject such requests. Please refer to 4.1.3, "Registration process" on page 62, for more details about the registration process.

The *directory administrator* is responsible for the maintenance of the directory (LDAP) containing information about certificates. The directory administrator is responsible for implementing naming conventions used in the directory as required by industry-related or organization standards. Access rights to the directory will be governed by the administrator. For instance, most roles within the organization may have only read access to the directory, while external users may have no access at all.

Whether an organization needs personnel dedicated to the *help desk* will again depend on the specific implementation. Help desk personnel will have to be supplied with detailed documentation describing procedures to follow when receiving requests either from customers or fellow employees. Refer to 4.3.6, "Production" on page 84 for more information.

Security policies may need to be reviewed on a regular basis in order to maintain a secure and efficient PKI. Where an organization's policies need to be amended, a *policy manger* can approve this. In practice, however, a steering committee could act as the policy manager.

The *security auditor or supervisor* is responsible for auditing the entire security system. This person may be required to submit reports to a steering committee or a team of reviewers. The security auditor will need to possess certain specialized skills. The person needs to understand cryptography and understand when and where it is required. An understanding of what a secure cryptographic system is and what procedures are required to ensure that the PKI stays that way is essential for this role. The responsibilities of the security auditor or supervisor include:

- Implementing the enterprise IT security policy (including certificate policy, Certificate Practice Statement, and key management policy).
- Approve and manage planning of backup and disaster-recovery procedures.
- Define and/or approve management and administration procedures for the security system.
- Define and/or approve procedures that third parties will have to follow, such as when the organization outsources their PKI services.
- Ensures all policies and procedures relating to IT security are documented.

- Regular and/or unscheduled reviews of audit logs.
- Ensures that all the above items are monitored regularly for compliance to enterprise and industry standards.

4.5.2 Key escrow

Key escrow means allowing a third party, such the employer or the government, to keep a copy of someone's private key. This may be desirable, for example, for law enforcement purposes. It allows that third party under certain circumstances to decrypt messages. Key escrow has been receiving much publicity in the recent past because of the privacy issues it raises. Issues concerning an individual's privacy versus interests of national security are outside the scope of this book.

However, key escrow techniques can be considered by an organization wishing to deploy a PKI as part of its backup and recovery plans. A commercial escrow agency could be an independent agency operating external to an organization, or an independent internal agency. One possible way of utilizing such a service while at the same time maintaining a high level of security is to encrypt private keys with the agency's public key and then store them locally. Thus, the third party does not actually receive any keys. Only when a private key needs to be recovered will the encrypted key be handed to the agency for decryption (by the agency's private key). An alternative escrow technique is to divide the private keys into two parts and encrypt each key part with the public key of a different security officer, for example. The encrypted key parts can, again, be stored on a local storage media under control of the owner or local authority. However, in this example, the keys are escrowed to individuals in an organization instead of an agency. Also, the escrow agent/agency only becomes involved in key management when it is necessary, for example, to recover a key. Finally, escrow techniques could be used with Smartcards. A user's signing key could be split into two key parts and escrowed to Smartcards.

Whether an organization uses escrow techniques is a business and security issue that needs to be carefully considered. Some organizations might find managing this aspect of the PKI too complex and costly and would prefer using a third party for this service. Issues, such as the integrity of the third party's staff, their security policies, and so on, have to be evaluated. Finally, in the event of using an external organization, the extent of their liability coverage in the event of key loss or other disasters has to be negotiated before it becomes a costly exercise for an organization.

4.5.3 Key recovery

Organizations that introduce a PKI also need to consider methods for *key recovery*. Key recovery lets an authority recover (rebuild) an encryption or decryption key if it becomes necessary. As an alternative, *information recovery* provides the means to recover encrypted information without exposing the actual decryption key. There are several techniques for key recovery, the simplest being key escrow (see 4.5.2, “Key escrow” on page 92). Key escrow as a means for key recovery, however, is refused by many organizations because its use (or misuse) is too simple and tempting, and it could, therefore, introduce an unnecessary security risk. To make it more safe, parts of a key can be encrypted and individually distributed to multiple places such that several people (or authorities) must agree to bring together the pieces when key recovery needs to be performed. But still, the process of breaking up a key is a single point of failure, and thus, a potential weak link in the security chain. Scalability issues are another limiting factor for key escrow because possibly large numbers of keys need to be stored for a potentially long period of time.

More sophisticated methods add *recovery information* to encrypted messages (encapsulation) that allow specialized software to later recover the key. No central storage and/or management of keys or parts of keys is required, thus scalability is not an issue. There are a number of provisions added such that the recovery of a key (or of the information) is not simply a matter of running a piece of software. The key recovery software will require some other pieces of information from different parties. This way, a high level of safety and protection from misuse can be maintained. The *IBM KeyWorks* product, for example, contains the *Key Recovery Server* (KRS) and the *Key Recovery Service Provider* (KRSP) that provide the functionality to implement key recovery. Using KeyWorks, the sending part involved in message encryption adds encrypted pieces of recovery information to the encrypted message. These pieces are individually encrypted using the public keys of one or more KRSPs. If key recovery becomes necessary, all KRSPs must be involved with their private keys to decrypt the fragments of the recovery information. This recovery information is required (but not necessarily sufficient) to then recover the key.

An organization may decide to implement key recovery techniques to protect itself from possible loss of intellectual capital should an important key be lost, just like a copy of a physical door key or a locksmith ensures that no door will be shut and locked forever. Government law or law enforcement agencies may also require implementation of key recovery in certain countries or in particular circumstances.

Security policies may need to include a statement if key recovery is incorporated into PKI. Also, the precise circumstances when key recovery is to be applied should be described, as well as the methods that are in place.

4.5.4 Processes and procedures

All processes and procedures for managing a PKI must be documented, including the administrative role required for each operation. Section 4.1, “Defining security policies” on page 59 discusses a wide variety of topics that need to be addressed by an organization’s security policies when deploying a PKI. This section provides guidelines for some administrative tasks that need to be followed during the normal operation of a PKI.

Some administrative tasks will occur frequently, so an established procedure on how to process such situations will be useful. Users will inevitably forget their private key passwords, for example, and so the necessary procedures to deal with this needs to be in place. Allowing password recovery services will depend on the implementation of the PKI. It may be a good idea to provide password recovery services, if the additional administration of generating a new key pair each time a user forgets their password becomes burdensome to the organization. A user that forgets his or her password could request a new password by filling in a standardized electronic form for password recovery and submitting it to an RA. Details should be asked about the password. For example, if it was written down somewhere, does the user suspect that anybody knows the password, is the user still in possession of the private key, and so on. The RA administrator should then assess the form and decide whether a password recovery is warranted or whether a new key pair should be generated. If password recovery is approved, the user needs to prove to the RA that he or she is indeed the owner of the private key. One way of doing this is to use a questionnaire. This questionnaire is a list of personal questions that the user chose and answered when he or she first registered for a certificate. For example: What is your time of birth? These questions should be of such a nature that only the user knows the answers to all of the questions, thereby identifying himself or herself. The user will then return the completed questionnaire to the RA. The RA application could then securely compare the answers received with the correct answers stored in the RAs database. If the number of correctly answered questions reaches a predetermined level, the user should be sent his or her password. Using the questionnaire method could be a useful form of identification for a large-scale PKI where user identification is difficult. In a smaller organization, the CA could require their employees to physically identify themselves before a password is recovered.

An organization should describe processes and procedures for other important operations, for example, private key compromise. A similar procedure could be followed as above where a user could identify himself or herself through the questionnaire method, alternatively an out-of-band method may be required.

One important mandatory operation that needs to be followed by all CAs is the CA private key rollover procedure or update. RFC 2510, *The Internet X.509 Public Key Infrastructure Certificate Management Protocols*, provides procedures to follow when a CAs key needs to be changed. Provisions are made to assist users that hold the CAs existing certificate to use the new CA certificate securely, as well as assisting users that hold the CAs new certificate to be able to use the CA's old certificate in order to verify existing data. The procedure requires the CA to publish an additional three self-signed certificates. RFC 2510 suggests the following operator actions:

1. The CA generates the new key pair.
2. The CA creates and publishes a certificate containing its old public key that is signed by its new private key.
3. The CA creates and publishes a certificate containing its new public key that is signed with its old private key.
4. The CA creates and publishes a certificate containing its new public key that is signed with its new private key.

The CAs old public key will be kept for non-repudiation purposes. The CAs old private key will no longer be required.

4.6 Migration considerations

An implementation of a PKI normally does not start from scratch. There will be applications and other services in place that need to be migrated to a PKI. This section overviews some of the issues that may arise when existing services and applications need to be migrated to exploit a PKI.

4.6.1 PKI enablement of existing applications and services

It is clear from the discussion that a PKI without applications and/or services utilizing it is of little use. However, many existing applications and services are not PKI-enabled.

Some of existing applications and services may already support a PKI and do not need to be changed. Modern Web browsers, for example, usually support client authentication using X.509v3 certificates. Authentication procedures may also already have some built-in functions for certificate authentication.

Organizations should be clear of how this enablement can be done. The most basic question is, are there APIs for my applications to interface with the PKI so as to utilize the functions provided by the infrastructure? The PKI solution must provide an API library to support the functions required to build an End-Entity interface. As mentioned in 2.1.1, “End-Entity (EE)” on page 22, an application program is an End-Entity. Application developers can use the APIs for the End-Entity to access and manipulate the functions provided by the PKI. Please refer to Figure 8 on page 22 to see how an End-Entity can utilize the PKI functions.

Therefore, when an organization chooses a PKI solution, it has to evaluate each PKI solution to see if the APIs can be called by the existing applications. Most common APIs are written in C, C++, or Java. If the existing applications cannot communicate with the programming language used in writing the APIs, the particular PKI solution may not be suitable, or another application programming layer has to be written to bridge the communication gap between the existing applications and the APIs. This may require extra skills, testing, and funding. Some PKI solutions provide high-level APIs for common applications, such as virtual private network (VPN) applications, secure e-mail S/MIME applications, SSL Web browser-based applications, and Smartcard-based authentication applications. This improves the ease of use of the PKI solutions and the deployment time may be shortened.

4.6.2 Integration of current uses

We have discussed some current uses of Public Key Infrastructures, such as SET and SSL, in 1.5, “Current use of Public Key Infrastructures” on page 18. Readers may have an interest to know if these two protocols can be integrated into the PKI that we have been discussing so far.

4.6.2.1 Secure Electronic Transaction (SET)

SET is an open technical standard for the commerce industry. Figure 7 on page 18 shows that SET is a protocol at the application layer. In the SET specifications, X.509 Version 3 certificates are used as the format of SET certificates. However, SET has defined some private extensions that are not included in the standard X.509 extensions. Actually, SET can be considered a custom-PKI, which is an industry-specific standard for particular electronic commerce applications. So, imagine a company which has implemented SET (as a SET CA issuing SET certificates to card holders), if it later wants to introduce a corporate PKI, a business-to-business PKI, or a customer-to-business PKI, in these circumstances, the SET custom PKI may not fit with the non-SET PKI. The format of the certificates may not be interoperable, and SET has its own procedures and operations. At the time of writing, many organizations are hosting both SET CAs and non-SET CAs if they really need both electronic commerce and other PKI-related applications. This segregation of PKI may be good in the sense that the PKI trust model is less complex and the electronic commerce part (using SET) already has many sophisticated products or solutions on the market. In addition, an organization may have two departments handling the SET system and the non-SET PKI aligning with other business functions.

However, a feasible approach is to implement a PKI solution that is able to issue both SET-compliant certificates and other certificates for other PKI applications, such as VPN, secure e-mail, and corporate PKI solutions. This sounds attractive since the PKI is generic enough for all applications, but organizations have to note that more customization and testing may be needed when using the generic PKI to issue SET compliant certificates.

More information about SET can be found, for example, at:

`http://www.setco.org/set_specifications.html`

4.6.2.2 Secure Sockets Layer (SSL)

The SSL protocol is widely deployed in Web servers and browsers and in merchant server software. It helps to provide a basic level of security for people to conduct business over insecure networks, such as the Internet. As mentioned in 1.5, “Current use of Public Key Infrastructures” on page 18, SSL is application-independent and the data is encrypted during the SSL-enabled communication. Unless properly used, SSL may not be sufficient for user authentication. When a user buys something over the Internet using SSL, the user normally only types in some personal and payment information, but this information is not verified by some trusted third party. The merchant cannot be sure of the user’s identify. SSL does allow digital certificates for mutual authentication. However, these certificates are optional and there is no common root CA for all SSL certificates. In most cases, SSL is only used for server authentication, which does not include any personal authentication. SSL server authentication certificates are normally issued and signed by different organizations, and the user or End-Entity registration process may not be comprehensive and may vary.

However, SSL is more powerful than most current uses. SSL allows for client certificate authentication. If client certificate authentication is configured on a Web server, the browser prompts the user for either a certificate (stored, for example, on a Smartcard) or for the password of a local certificate store. This way, the certificate is bound to an individual and can, therefore, be used for reliable user authentication. In order for this to work, the Web server and the browser must be in possession of the corresponding CAs root certificate to be able to accept a certificate. If this is not the case, a Web browser, for example, may present the user a series of warnings and dialogs, requesting the user to accept an unverified certificate.

In addition to authentication, SSL also encrypts all traffic between the two communicating parties.

Thus, SSL integration is relatively easy as long as the CA (RA) software supports the respective certificate extensions required by SSL.

Part 2. Standards

Chapter 5. PKI-related standards

Standards, be they formal or de facto, are a prerequisite for interoperability between different venders' product implementations. There is a number of related PKI standards and proposals (Internet Drafts) that define a PKI in close relationship to the X.509 certificate standard.

A first X.509 standardization effort for PKI has been made in RFC 1422, *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*, in 1993. PEM was an Internet Engineering Task Force (IETF) standard based on X.509 V1 certificates and its purpose was secure e-mail. Although today's PKIX (*PKI* for *X*.509 certificates) standards are based on the PEM standard, they have significantly modified and enhanced the PEM standard and added flexibility for certificate management. The PKIX effort focuses on the exploitation of X.509 V3 certificates and it aims at making PKI system usable for general purposes, rather than only for a specific application.

This chapter lists and briefly explains the most important standards and drafts that relate to a PKI.

5.1 ITU-T recommendations

The following standards are known as the *X series* of the International Telecommunications Union - Telecommunications (ITU-T), formerly known as the CCITT. The purpose of the X series is the standardization of data networks and open system communication. X.509 was defined as a part of the X.500 series recommendations. The X.500 series recommendations generally define directory services, which are used in an Open Systems Interconnection (OSI) environments. X.509, which was originally defined in 1988, and the latest Version 3, defined in 1996, is specifically defined as the authentication framework for using a directory service. After that, the certificate format, which was defined as a portion of X.509, has become focused on PKI standardization. The following are X series standards that relate to PKI:

X.208 *Specification of Abstract Syntax Notation 1 (ASN.1)*
ASN.1 is an abstract language used to describe data structures in a machine- and implementation-independent way. X.509 certificate data structure, for example, is defined using ASN.1.

X.209 *Specification of Basic Encoding Rules for Abstract Syntax Notation 1 (ASN.1)*
X.209 describes how to encode ASN.1-formatted data structure into

binary data, and this is called basic encoding rules (BER). The distinguished encoding rules (DER) is a subset of BER. DER is used in the X.509 certificate data structure encoding process.

X.500	*Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services* This is the top-level recommendation for directory services.
X.509	*Information technology - Open Systems Interconnection - The Directory: Authentication framework* X.509 defines the authentication framework for the X.500 directories. As part of X.590, certificates are defined that play an important part in most of the succeeding PKI drafts and standards.

5.2 Internet RFCs

Internet Request for Comments (RFCs) are the formal and informal standards defined by the Internet Engineering Task Force (IETF). *PKIX*, which is an acronym for Public-Key Infrastructure (X.509), defines a PKI using X.509 digital certificates. The PKIX working group within the IETF has been working on the implementation of an Internet-based PKI standard, and their RFCs are considered to be widely accepted in the industry. The following RFCs relate to PKIX:

RFC 2459	*Internet X.509 Public Key Infrastructure Certificate and CRL Profile* RFC 2459 is the body of the PKIX standard and contains detailed information about X.509 certificate and CRL formats.
RFC 2510	*Internet X.509 Public Key Infrastructure Certificate Management Protocols* This RFC describes the administrative task mechanism called Certificate Management Protocols (CMP) for PKI (see also 2.2.2, “Protocols” on page 31).
RFC 2511	*Internet X.509 Public Key Infrastructure Request Message Format* This RFC defines the message format that is used in CMP defined in RFC 2510 protocol.
RFC 2527	*Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework* RFC 2527 describes two important PKI policy structures, the Certificate Policy and the Certification Practices Statement (CPS), along with their relationship.
RFC 2528	*Internet X.509 Public Key Infrastructure Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key*

	Infrastructure Certificate RFC 2528 describes the format and semantics of fields in X.509 V3 certificates containing KEA keys, which is defined by the Computer Security Resource Clearinghouse (CSRC) of the National Institute of Standards and Technology (NIST).
RFC 2559	*Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2* This RFC, along with RFC 2585 and RFC 2587, describes the methods of accessing the Certificate Repository (CR). This RFC defines the access based on the LDAP protocol. (Note: There is also an Internet Draft describing the use of the LDAPv3 protocol. See 5.3, “Internet Drafts” on page 104.)
RFC 2560	*X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP* RFC 2560 describes the Online Certificate Status Protocol (OCSP) that provides a means for getting certificate revocation status information in a timely manner without the use of a CRL.
RFC 2585	*Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP* RFCs 2559, 2585, and 2587 describe the methods of accessing the Certificate Repository (CR), and this one (RFC 2585) in particular defines access by the use of the FTP and HTTP protocols, which can be used when the CR is not implemented in an X.500/LDAP directory.
RFC 2587	*Internet X.509 Public Key Infrastructure LDAPv2 Schema* RFCs 2559, 2585, and 2587 describe the methods of accessing the Certificate Repository (CR). RFC 2587 defines attributes and object classes (schema) for use in LDAP for the CR. This RFC closely relates to RFC 2559.

Every RFC is associated with a status; it may be *historical* or *informational* only, while others might be actual *standards*. Some other statuses exist, too. To find out about the status of an RFC, or to read RFCs, the IETF Web site is a good source:

```
http://www.ietf.org
```

There is another working group in IETF in regard to PKI, the *Simple Public Key Infrastructure* (SPKI) working group. This group is working for a simpler PKI standard compared to the PKIX effort. From the SPKI working group, two RFCs have evolved, that are, at the time of writing, in the *experimental* status:

RFC 2692	SPKI Requirement This RFC describes the basic ideas behind the SPKI effort to create a simplified PKI.
RFC 2693	SPKI Certificate Theory RFC 2693 defines a new standard form of simplified certificates that are primarily geared for use in authorization processes, rather than authentication.

You can access the PKIX and SPKI group charters at:

```
http://www.ietf.org/html.charters/pkix-charter.html
http://www.ietf.org/html.charters/spki-charter.html
```

SPKI was merged with Simple Distributed Security Infrastructure (SDSI) that is described in 5.5, “Other standardization efforts” on page 107.

5.3 Internet Drafts

PKI standardization is, at the time of writing, still under work. Because of this, some work is available only as Internet Drafts, rather than RFCs. There are some Internet Drafts that relate to PKI as follows:

Internet X.509 Public Key Infrastructure PKIX Roadmap: Describes in an easy-to-understand way the current status and the proposed future of the PKIX standard. The reading of this draft is recommended because it gives a broad and clear overview of the PKIX roadmap.

Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates: Describes the use of the ECDSA for keys and signatures in certificates.

Certificate Management Messages over CMS: Defines the Certificate Management Protocol using Cryptographic Message Syntax (CMS), which is a superset of PKCS #7 (see following paragraph). This Internet Draft addresses two immediate needs within the Internet PKI community, one is an interface to public key certification products and services based on CMS and PKCS #10, and the other is a SMIMEV3 for a certificate enrollment protocol for DSA-signed certificates with Diffie-Hellman public keys.

Time Stamp Protocol: Describes the Time Stamp Authority (TSA) that proves that a datum, like a digital signature, existed at a particular time. This is considered effective and essential for reliable non-repudiation service.

Additionally in this Internet Draft, a Temporal Data Authority (TDA) is defined that can add supplemental evidence data to the TSA response.

Data Certification Server Protocols: Describes Data Certification Server (DCS) Protocols that can offer non-repudiation service for the request. A DCS server will do more verification process than a TSA and it will return a data certification token.

Internet X.509 Public Key Infrastructure Qualified Certificates Profile: Expands RFC 2459 by adding a qualified certificate profile for certificates that represent human entities for legal purposes.

Diffie-Hellman Proof-of-Possession Algorithms: Originally, Diffie-Hellman was a key agreement algorithm that could not be directly used for signing or encryption. This Internet Draft describes two new signing algorithms using the Diffie-Hellman key agreement process designed to provide a proof-of-possession rather than general purpose signing process.

An Internet Attribute Certificate Profile for Authorization: An Attribute Certificate (AC) is a new type X.509 certificate, other than the X.509 Public Key Certificate (PKC). Although a PKC is mainly used authentication, data integrity, and confidentiality, an AC will be used for authorization service. An AC does not contain public key information, but other certificate owner's attribute information, such as the group membership, role, clearance, and other access control information, and it will be used in conjunction with PKC.

Basic Event Representation Token: A Basic Event Representation Token (BERT) defines a token structure that represents events in time.

Extending Trust in Non-repudiation Tokens in Time: This proposal describes a method of maintaining the trust in a token issued by a non-repudiation trusted third party (may be either of DCS, TSA, or TDA) after the key initially used to establish trust in the token expires. The document describes a general format for storage of DCS/TSA/TDA tokens for this purpose, which establishes a chain of custody for the data.

Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv3: This proposal advances the use of an LDAP directory to leverage some of the advantages of the LDAPv3 protocol, such as better authentication methods, for retrieving certificates from a directory (see also RFC 2559).

Simple Certificate Validation Protocol (SCVP): This interesting proposal defines a protocol that allows clients to offload certificate validation to servers. Certificate validation can be very complicated and, thus, too costly for small and uncritical applications (see also 3.5, “Certificate validation” on page 49).

Using HTTP as a Transport Protocol for CMP: Describes a proposal on how to use the HTTP protocol to carry Certificate Management Protocol (CMP) messages.

5.4 PKCS

Public-Key Cryptography Standards (PKCS) is a numbered set of standards defined by RSA since 1991. Although they are informal standards, today they are widely accepted in the industry, based on the dominant RSA public key algorithm. Some PKCS standards have also been published as informational RFCs. Currently, PKCS has #1 to #15 standards as follows:

PKCS #1	*RSA Encryption Standard* PKCS #1 defines the encryption and digital signature format using RSA public key algorithm.
PKCS #2	Incorporated into PKCS #1.
PKCS #3	*Diffie-Hellman Key Agreement Standard* PKCS #3 defines the secret key exchange protocol using Diffie-Hellman algorithm.
PKCS #4	Incorporated into PKCS #1.
PKCS #5	*Password-Based Encryption Standard* PKCS #5 defines the encryption method by secret key derived from a password that is intended primarily for private key encryption when its is transferred.
PKCS #6	*Extended-Certificate Syntax Standard* PKCS #6 defines the X.509 certificate extension format that was reflected on X.509 V3.
PKCS #7	*Cryptographic Message Syntax Standard* PKCS #7 defines general message syntax includes digital signature and encryption.
PKCS #8	*Private-Key Information Syntax Standard* PKCS #8 defines the syntax of private keys, which includes the private key itself and a set of attributes.
PKCS #9	*Selected Attribute Types* PKCS #9 defines the attribute types that are used in data formats defined in PKCS #6, PKCS #7, PKCS #8, and PKCS #10.

PKCS #10 *Certification Request Syntax Standard*
PKCS #10 defines the syntax of a request format in case of a certificate issue request.

PKCS #11 *Cryptographic Token Interface Standard*
PKCS #11 defines the technology-independent device interface, called *Cryptoki*, that is used for security tokens, such as Smartcards.

PKCS #12 *Personal Information Exchange Syntax Standard*
PKCS #12 defines the portable format for storing and transporting a user's private keys or certificate.

PKCS #13 *Elliptic Curve Cryptography Standard*
PKCS #13 defines the elliptic curve cryptography algorithm for use in a PKI. The elliptic curve cryptography algorithm was developed by RSA.

PKCS #15 *Cryptographic Token Information Format Standard*
PKCS #15 defines the standard about the interpretability of cryptographic tokens, which were originally defined in PKCS #11.

RSA PKCS information can be accessed on their Web site at:

`http://www.rsa.com/rsalabs/pubs/PKCS`

5.5 Other standardization efforts

Other than the previously mentioned standardization activity, a number of organizations or groups has been (and still are) working on security standardization.

The *Simple Distributed Security Infrastructure* (SDSI) has been developed mainly by Ronald L. Rivest, who is one of the RSA founders. It outlines a simple PKI design. Because the SDSI efforts are very similar to the IETF SPKI goals, the two efforts have been merged. The SDSI/SPKI approach is a contrast to PKIX. SDSI/SPKI employs local name spaces rather than X.500/X.509 hierarchical global name space, and *S-expression* notation is used for certificate description rather than ASN.1. Further information about SDSI/SPKI can be found, for example, at:

`http://theory.lcs.mit.edu/~cis/sdsi.html`

Intel Corp. has developed the *Common Data Security Architecture* (CDSA), which is widely accepted in the industry. The CDSA has been adopted by The Open Group and it provides a set of layered architecture that covers overall security services, and its implementation is geared toward PKI (see also Figure 12 on page 41). You can access CDSA information at:

`http://www.opengroup.org/public/tech/security/cdsa/index.htm`

Other than the PKI standardization activity, there is a remarkable trend toward the cryptography issue. In 1997, the National Institute of Science and Technology (NIST) requested submissions for a symmetric key algorithm known as the Advanced Encryption Standard (AES). AES is to become the successor of Data Encryption Standard (DES), which appeared in 1976 and is likely to be the most widely-used symmetric key algorithm today. Its cryptographic strength, however, is becoming insufficient in today's computer technology. At the time writing this book, there are five candidates for the AES, which are MARS (IBM), RC6 (RSA), Rijndael (team of Belgian cryptographers), Serpent (Ross Anderson, Eli Biham and Knudsen), and Twofish (Counterpane Systems). The NIST will again ask for comments and will choose the finalist around May 2000. The algorithm will become a Federal Information Processing Standard and is expected to become the standard encryption algorithm used worldwide. The AES will be a 128-bit block cipher with either a 128-bit, 192-bit, or 256-bit key size. For more information, please refer to the following Web site:

`http://csrc.nist.gov/encryption/aes/aes_home.htm`

Part 3. IBM products supporting a PKI

Chapter 6. IBM SecureWay Trust Authority

In addition to professional consulting services, IBM offers comprehensive products for a PKI deployment and the supporting components that enable customers to build PKI solutions. Besides powerful hardware products, a number of security-related software products are offered by IBM of which two are of special interest and need to be pointed out in relation to a PKI:

- The IBM SecureWay Directory
- The IBM SecureWay Trust Authority

The former, the IBM SecureWay Directory, is a no-charge offering that incorporates a highly-scalable LDAP directory available for Microsoft Windows NT, IBM AIX, Sun Solaris, IBM OS/400, and IBM OS/390. An LDAP directory, as we have seen in the previous chapters, is an essential backbone service for a PKI as a public store for certificates and CRLs. However, a thorough discussion of LDAP is beyond the scope of this book. A brief introduction is included in Appendix A, “Introduction to LDAP” on page 211. More detailed information about LDAP and the IBM SecureWay Directory can be found in the IBM Redbooks listed in C.1, “IBM Redbooks” on page 221, or at:

`http://www.ibm.com/software/enetwork/directory`

The IBM SecureWay Trust Authority product also constitutes an integrated building block of the *IBM SecureWay FirstSecure* offering, which is a total security solution from IBM that includes a Security Policy Director, Secure Boundary Services, Intrusion Detection, PKI, and a comprehensive set of development tools, called the SecureWay Toolbox.

This chapter provides an overview of IBM SecureWay Trust Authority. The individual components of Trust Authority will be introduced as well as the components that support it. A discussion about installation and configuration issues will be presented, including how to go about planning an installation of Trust Authority.

In order to remain updated with the Trust Authority product, please visit the IBM Secureway Trust Authority Web site at the following URL:

`http://www.ibm.com/software/security/trust`

In particular, the Trust Authority online documentation can be found at the Trust Authority library site:

`http://www.ibm.com/software/security/trust/library`

6.1 Product overview

The IBM SecureWay Trust Authority (TA) offering ensures secure, trusted communications for applications and users. Trust Authority does not only provide certificate services to applications, it enables an organization to deploy a total PKI solution. The Trust Authority product provides some important features that will be discussed.

Trust Authority is more than just a certificate server since it supports the entire life cycle of a certificate. This includes requesting, validation, issuing, publishing, and administration of certificates. Trust Authority manages certificate life cycles using cryptographic standards allowing different products to interoperate. It supports several enrollment protocols and certificate types, which allow certificates to be issued through multiple protocols and used for multiple applications and purposes. Using Trust Authority, certificates can be issued for use with VPN applications, e-mail applications using S/MIME, PKIX client applications, Web applications, and others.

Figure 19 on page 113 illustrates the components that make up a Trust Authority system. The *Registration Authority* (RA), *Certificate Authority* (CA), and *Audit Server* components are separate logical components of TA. In addition, a Java-based TA Client (shipped with the TA product) and standard Web browsers can be used to utilize the functions provided by TA.

TA provides a Java applet, which presents a graphical interface to RA administrators, called the *RA Desktop*. The RA Desktop is accessed by authorized RA administrators and it allows for the management of certificates and other requests. Multiple RA administrators can be defined and authorized to use the RA Desktop.

Web browser access to the RA component is supported for user certificate requests and issuance of certificates. The *TA Client* is a Java application provided with TA, which allows users to obtain and use PKIX compliant certificates without using a Web browser.

The RA and CA components have audit clients incorporated that are used to communicate audit events to the Audit Server. The Audit Server then writes the events to an audit log, which is typically a DB2 database.

The RA, CA, Audit Server, and TA Client components use local data stores. A data store stores information about transactions in progress and status information about those transactions. The DB2 databases are also used to store persistent information, such as certificates issued, user details, and so on, on the RA, CA, and Audit Server components.

TA utilizes an LDAP directory for publishing certificates and CRLs.

Details of the core components are discussed in the following section, 6.2, “The Trust Authority building blocks in detail” on page 115.

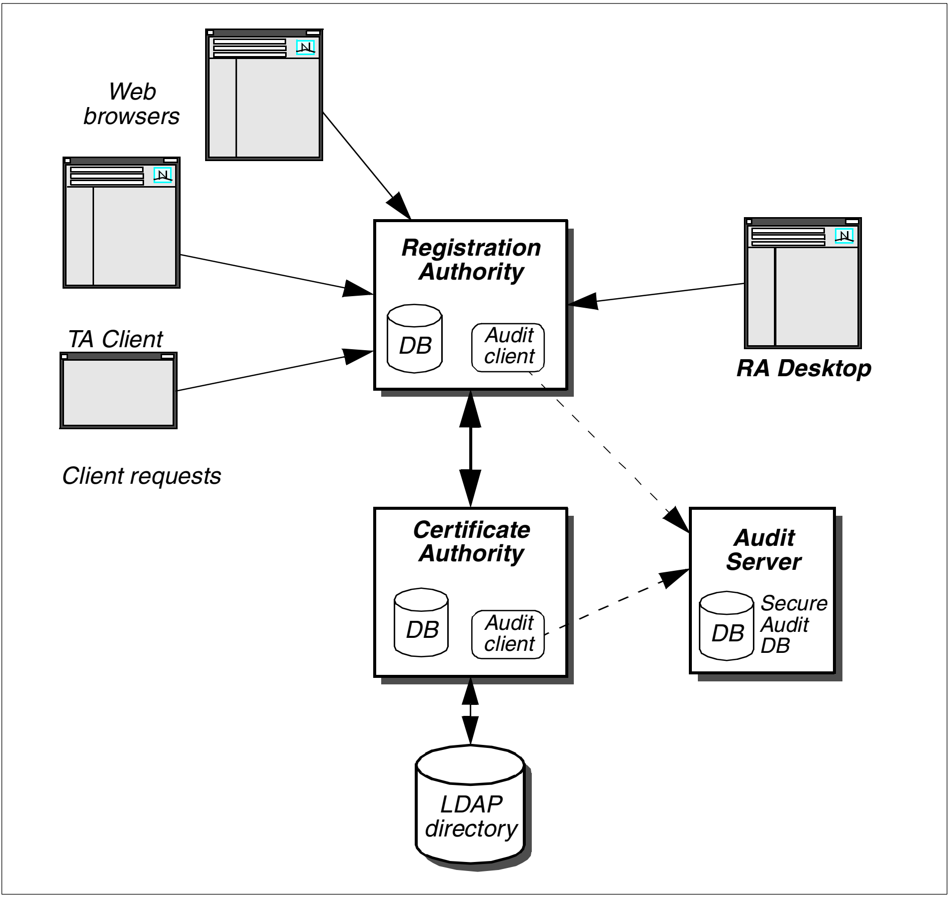


Figure 19. The Trust Authority building blocks

All the Trust Authority components can be installed on a single machine. However, an organization may want to distribute the processing across multiple machines, depending on the size of the organization, performance considerations, anticipated workload, availability, and other implementation issues. For example, an organization may already have an LDAP service in place and may want to continue having it on separate machines. Whatever an organization's implementation might be, there are certain Trust Authority components that must be installed on the same machine. Figure 20 shows a summary of the functional dependencies of the Trust Authority components, that is, the functions that must reside within the same machine(s).

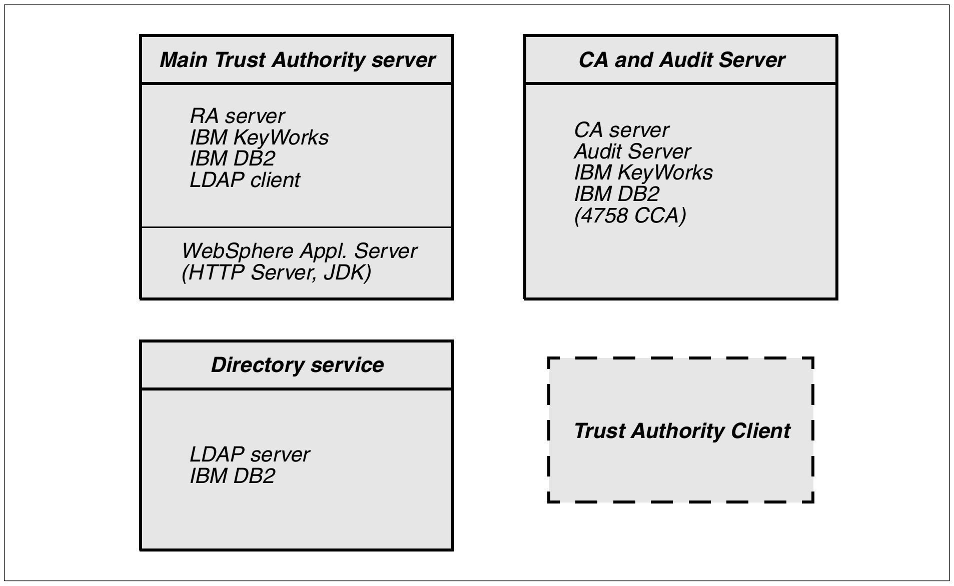


Figure 20. The Trust Authority functional dependencies

IBM KeyWorks is an underlying technology that must be present on all components. It is installed automatically as part of Trust Authority and has been included here for completeness. Each of the following combinations of components must reside on one single machine (see Figure 20):

- The directory server and related database (LDAP server and IBM DB2).
- The CA server and the Audit Server and their related databases (IBM DB2).
- The RA server, the HTTP Server, and WebSphere Application Server, all administrative programs and configuration and registration databases (IBM DB2). These components make up the component known as the *Main Trust Authority Server*.
- The TA Client runs on a Windows 95/98/NT machine.

For a list of configurations showing how server components can be distributed, see the *Trust Authority Up and Running* available with the product and online on the Trust Authority library Web site.

6.2 The Trust Authority building blocks in detail

The following is a description of each of the building blocks of Trust Authority as introduced in the previous section.

6.2.1 The Certificate Authority (CA)

The Trust Authority CA is responsible for signing and issuing certificates. It is also responsible for creating and updating CRLs. CRLs need to be signed by the CA before they are published. A CA configuration file controls the settings for the CRL creation schedule and the lifetime of the CRL.

The Trust Authority CA component processes requests it receives from the RA. Such requests may be for issuing, renewing, or revoking of certificates.

The CA generates unique serial numbers for all new and renewed certificates. A list of all certificates issued by the CA is maintained by the CA and stored in its local database. The list is called an *Issued Certificate List* (ICL). The ICL includes information such as certificate status, certificate serial number, and CRL information. The ICL is used to create the CRL that is published to a directory. The CA generates a message authentication code (MAC) for each record stored in its database for integrity checking purposes. An administrator will have the option of performing an integrity check on the database or not. If integrity checking is set, a new MAC value is calculated on the database and compared to the MAC value generated when the database was last saved. If the values do not match, the administrator will be alerted that the database may have been tampered with since its last update.

The Trust Authority CA stores its own keys and certificates in a special Trust Authority Keystore area. However, for maximum protection of CA keys, the Trust Authority CA can be integrated with the IBM SecureWay 4758 PCI Cryptographic Coprocessor. In this situation, the CA signing key will be triple-encrypted by the master key of the cryptographic hardware. The master key is stored securely in the co-processor. At the time of writing, the IBM SecureWay 4758 PCI Cryptographic Coprocessor was supported on AIX only. For more information about the cryptographic coprocessor, visit the IBM SecureWay 4758 PCI Cryptographic Coprocessor Web site at:

`http://www.ibm.com/security/cryptocards`

The Trust Authority CA can be configured to operate in a variety of trust models. The CA is able to produce self-signed certificates if required to operate in a single administrative domain. The Trust Authority CA also supports cross-certification and CA hierarchies. If the CA is required to operate with other CAs, a number of options are available. A Trust Authority CA can act as a root CA and sign other CAs certificates, resulting in a hierarchy. Alternatively, the CA can participate in the hierarchy, by having other CAs sign its certificate and becoming a subordinate CA. The Trust Authority CA also supports one-way cross-certification. In order to allow for two-way cross-certification, each CA needs to obtain a cross-certificate from the other CA.

The Trust Authority CA supports most of the fields and extensions specified in the X.509v3 certificate format. Standard, common, and private certificate extensions can be used with the Trust Authority CA. Additionally, the CA supports generation and validation of certificates with user-defined extensions.

Finally, the CA generates audit records. These records are passed on to the Audit Server, which then stores the records in a DB2 database.

6.2.2 The Registration Authority (RA)

To recap from the previous chapters, the primary function of the RA is to verify the identity of the requesting party and to ensure that he or she is authorized to receive the type of certificate requested. The Trust Authority RA component offers a broad range of features for such activities, enabling an RA administrator to carry out enrollment, administration, and customization activities. Some of the Trust Authority RA component enrollment features include:

- A set of Java based *enrollment forms* for use with requests through Web browsers. Using this method allows for authentication of registering parties and encryption of data between them.
- A set of *certificate profiles*. A certificate profile defines the type of a certificate and what it can be used for. Using a certificate profile will make it easier to issue certificates to users. Certificate profiles can be edited to suit specific solution requirements.
- A *preregistration process*. This allows one user, typically a security administrator, to request a certificate for another user. In such a situation, a manager could request a certificate on behalf of an employee. The employee could then obtain the certificate using the Trust Authority client application.
- The use of *registration domains*. During configuration of Trust Authority, a registration domain is created. A registration domain is the administrative unit used by Trust Authority to govern registration activities. It has a domain name that is a subset of the URL used to run the registration

application. There is a single registration domain per installation of an RA. An organization will customize a number of files that will eventually represent its business and certificate policies as well as control its resources within such a domain.

- A number of *configurable files* are provided to enable customization of the enrollment forms and the registration process. Configuration files allow for the customization of certain runtime variables. HTML and Java Server Pages (JSP) template files are provided for easy customization. Sample notification letters are provided for e-mail notification of users.
- A *policy exit* allows an organization to execute their own programs for additional checking and processing when certain events occur.
- The *RA Desktop* is an applet that allows administrators to manage certificates. Using the RA Desktop, RA administrators have the ability to search for, list, and sort certificates in their various states. RA administrators can then approve certificate requests, revoke certificates, and perform other tasks on certificates.

Subsequent sections in this chapter explain these features in more detail.

6.2.3 The Audit Server (AS)

All events processed by the CA and RA are forwarded to the audit subsystem and recorded in the audit log. The Audit Server allows for the audit clients to mask certain events. Although some events will always be logged, using the masking facility allows an organization to filter any events it does not need.

Once the Audit Server receives the event data, it records it in an audit log, which is stored in a DB2 database.

Event data stored in the audit logs are stored as records. A message authentication code (MAC) is computed for each audit record. A MAC is used to determine whether an audit record within the audit log has been tampered with since its last update. The Audit Server also provides tools for integrity checking of the entire database. The Audit Integrity Check utility is used to check the integrity of the Audit Server database and archive files.

It usually is a good idea to archive audit data and store it in a secure area. The Audit Server makes archiving of audit data easy by providing the Audit Archive and Sign utility for archiving and signing audit log files.

6.2.4 The TA Client

The TA Client is a Java application that users can use to manage their certificates. The TA Client can be used instead of the provided Web interface to do client-side certificate management, such as requesting and downloading certificates. The TA Client allows a user to perform actions that are not normally possible through the Web interface, such as some handling capabilities of certificates other than e-mail or Web browser certificates.

The TA Client uses standard PKIX/CMP protocols to communicate with the RA (see also 2.2.2, “Protocols” on page 31).

6.2.5 The supporting components

Besides the core building blocks as described in the previous sections, additional components support and complement Trust Authority. This section overviews these supporting components.

6.2.5.1 The Directory Server

IBM SecureWay Trust Authority can use the *IBM SecureWay Directory* (or basically any other suitable LDAP directory) as a Certificate Repository (CR). The RA server publishes information to the CR, including:

- Certificates, which are used for encryption and digital signature
- CRLs
- Information about the CA and the RA and their certificates

In order to store the above information in a CR, the directory server has to have a schema which defines the directory’s distinguished name (DN) syntax. Each certificate’s subject name is represented by the DN (for example *cn=John Smith,ou=Security Department,o=IBM,c=US*), and other certificate items are represented as directory attributes and their values.

The PKI-related schema file, along with attribute files and object files, are supplied in the Trust Authority package.

6.2.5.2 The WebSphere Application Server

Any browser-based interaction with Trust Authority is processed by the *IBM WebSphere Application Server* in conjunction with a Web server (HTTP server) and Trust Authority application logic. The IBM WebSphere Application Server and the *IBM HTTP Server* are packaged in the IBM SecureWay Trust Authority media for easier installation. The Web interfaces are the user interface and the RA Desktop. Through the user interface, users can request certificates, inquire about approval status information, and download certificates. Through the RA Desktop Web pages, the RA administrator can manage certificates and certificate requests. The Web interfaces are provided by Java Server Pages (JSP) including Java applets. For secure communication, the following three types of methods are prepared:

HTTP	Non-SSL, or public requests (default port 80)
HTTPS	SSL without client authentication (default port 443)
HTTPS	SSL with client authentication (default port 1443)

When installing and configuring Trust Authority, the configuration of the Web server is automatically done for the above ports.

6.2.5.3 The DB2 database

As depicted in Figure 19 on page 113, the RA, CA, and the Audit Server store some information in a local database. *IBM DB2* is used as a reliable and scalable repository to store several kinds of information. The following default databases are prepared during Trust Authority installation and configuration:

cfgdb	Trust Authority configuration database
ibmdb	CA database
pkrfdb	Registration database
adtdb	Audit database
ldapdb	LDAP directory database

These databases can reside on one system if all components of Trust Authority are installed on one single system, or they may reside on up to three separate systems (see Figure 20 on page 114).

6.2.5.4 IBM KeyWorks

IBM KeyWorks is a comprehensive set of layered security services forming one of the most mature CDSA implementations (see also Figure 12 on page 41). Trust Authority uses IBM KeyWorks as an underlaying security framework. IBM KeyWorks can be exploited in the following ways:

- As middleware embedded in applications
- As a security toolkit for applications directly accessing security services

IBM KeyWorks security services include:

- Cryptography
- Data/key recovery
- Trust policy
- Data storage
- Certificate library

During the installation of IBM SecureWay Trust Authority, KeyWorks is automatically installed and does not normally require any separate or special attention.

6.2.5.5 The security model within Trust Authority

Within a Trust Authority system, a number of security functions are used to protect the PKI resources and to maintain the integrity of the system.

Critical messages between the components of Trust Authority and between clients (users) and the RA server use encrypting protocols to protect against eavesdropping and data tampering.

The Audit Server receives audit events from the audit clients incorporated into the RA and CA and then computes a message authentication code (MAC) for each audit record to keep its integrity. At the same time, the masking can be used for certain audit events to prevent other non-relevant events from being reported.

A CA server also computes a MAC for each record, which is governed and written to the DB2 by the CA, to protect against data tampering. In the AIX version of Trust Authority, a CA server can utilize additional security features provided by the IBM 4758 PCI Cryptographic Coprocessor.

The entries in a CR can be properly controlled for its access by LDAP server capability. For example, clients can retrieve CRL information from the directory, but they cannot alter it.

6.3 Installation planning

In this section, the Trust Authority (TA) installation planning steps are discussed to make sure the requirements for a TA system are met. TA is a client/server type application and both clients and servers are covered. You should also refer to the *Trust Authority Up and Running* guide (available online on the Trust Authority library Web site) as a source of additional information about installation planning.

6.3.1 Server environment

Trust Authority supports AIX and Windows NT as server platforms. The following are the minimum requirements for operating systems:

- IBM AIX/6000 (AIX), Version 4.3.2
- Microsoft Windows NT, Version 4.0 with Service Pack 5 (at the time of writing, only Service Pack 5 was supported)

Trust Authority gives you some flexibility when selecting the system environment, since not all components must reside in one single system (see also Figure 20 on page 114). The following basic configurations are possible candidates, while other options also exist:

- All components (CA, RA, Audit Server, and LDAP server) can run on one single machine. This is likely to be the case in a typical test, evaluation, and education environment.
- The RA and the CA can run on separate machines, and the LDAP server may be running on yet another machine or in one of the TA machines.

Bear in mind when planning for a server environment that the Audit Server cannot be installed separate from the CA server, and that the RA server must also be running the WebSphere Application Server and the HTTP Server. One of the purposes of a test and verification scenario will be to evaluate the load on each component in order to derive a suitable machine configuration.

Before Trust Authority can be installed, the following required supporting components must be installed; they are provided in their required release levels with the Trust Authority product package:

- *IBM SecureWay Directory* – You can install it on the same machine as Trust Authority, or on a remote machine. For small environments or for test and education purposes, it is advantageous to install the IBM SecureWay Directory on the same machine as the CA and RA server components because its configuration will then be done automatically. The IBM SecureWay Directory server, in turn, requires IBM DB2. If installed on the same machine as the CA or RA, the same DB2 installation can be shared.
- *IBM WebSphere Application Server (WAS)* – The WAS, Standard Edition 2.0.3, includes the IBM HTTP Server and the Java Development Kit (JDK). The WAS must be installed on the same machine as the RA.
- *IBM DB2 Universal Database* – DB2, Enterprise Edition Version 5.2 with FixPak 6 and 10, must be installed on all machines with Trust Authority server functions (CA, RA, and Audit Server).
- *IBM 4758 PCI Cryptographic Coprocessor* – As an optional CA feature, the IBM 4758 PCI Cryptographic Coprocessor can be used on AIX. If used, it must be installed prior to installing the TA CA component, along with the *sway.cca* fileset support of IBM KeyWorks. Please check the media package or the Trust Authority Web site for the sway.cca fileset.

6.3.2 Client environment

For the client environment, the following four different client types can be identified and distinguished:

- End users who connect to the Trust Authority server(s) through a *standard Web browser*. The minimum supported browser level is either Netscape Navigator or Communicator, Version 3.0, or Microsoft Internet Explorer, Version 4.0. The browsers must have Java enabled.
- End users who connect to the Trust Authority server(s) through the *TA Client* program. The client application can be installed from the Trust Authority media package. The TA Client is supported on Microsoft Windows 95/98/NT.
- A Registration Authority (RA) administrator, who uses the *RA Desktop* applet. An operating system is one of the Microsoft Windows 95/98/NT. The minimum browser level is either Netscape Navigator or Communicator, Version 4.51, or Microsoft Internet Explorer, Version 4.01 (with service pack 1). With Internet Explorer, you must have the Java Virtual Machine (JVM), Release 5.00, build 3167 or later.
- During customization of the Trust Authority server, a *Setup Wizard* needs to be run in a Web browser on the server or a remote machine (recommended). This is a one-time step during the customization of TA. Supported operating systems are either IBM AIX or Microsoft Windows 95/98/NT. A JDK 1.1-based, applet-enabled Web browser is needed, and the minimum level of browser is either Netscape Navigator or Communicator, Version 4.05, or Microsoft Internet Explorer, Version 4.01. The Java swing class comes with Trust Authority and must be installed on the machine where the Setup Wizard runs.

6.4 Installing Trust Authority

As with any other non-trivial application, it is recommended to start with a separate exploring and training installation of Trust Authority. The following is a step-by-step overview description of the installation and basic configuration process for a single-system environment. It is important to refer to the *Up and Running Guide*, the *Installing Trust Authority on AIX* or the *Installing Trust Authority on Windows NT*, and the *README* documentation for additional information and for details about setting up Trust Authority in a multi-system environment. These documents can be found on the Trust Authority library Web site (see introduction to this chapter), in the media package, or through instructions contained in the media package. The two *Installing* documents for AIX and NT, respectively, are the most pertinent documents for installing Trust Authority, and the *README* document contains last minute information. The documents also list the exact prerequisite software levels at the time of installation pertinent to the maintenance level of Trust Authority as of installation time.

In the following, the documents listed above are simply referred to as *installation documentation*. Please refer to C.3, “Other resources” on page 221, for a list of other documentation available for Trust Authority.

6.4.1 Installation and basic configuration

The following steps summarize the installation and configuration steps for planning purposes.

Hardware and software requirements – Ensure the hardware and operating system prerequisites are in place. Make sure that TCP/IP communication is set up and working, including host name resolution. On AIX, make sure the correct level of xlC.rte is installed. On both platforms, ensure JDK is installed and at the level included with the Trust Authority media package. If you choose to create separate file systems on AIX for the four databases as listed in the product documentation, make sure they have at least the following free space available (the installation documentation recommends higher values that include some assumptions on production data):

/local	100 MB
/dbfsibm	40 MB
/dbfspkrf	40 MB
/dbfsadt	40 MB

During the configuration of Trust Authority, files are being created in and copied into the /usr file system, so there should be at least 16 MB of free space in /usr.

LDAP server – Install the IBM SecureWay Directory Server. If you choose to install it on the same machine as the Trust Authority servers, do not do any basic configuration, such as specifying an administrator account, password, or default database. The Trust Authority configuration process will do this later.

DB2 – Install DB2 Enterprise Edition and Client Application Enabler with the appropriate fix pack. The correct release of DB2 is shipped with Trust Authority for both platforms with appropriate installation scripts. DB2 must be set up to support UTF-8 code set. On Windows, this is achieved by an environment variable *DB2CODEPAGE*, which needs to be set to *1208*. Do not create either an administrative instance or a client instance and deselect automatic start of any components at boot time.

User ID on Windows NT – On Windows NT, create a new user account (for example, *cfguser*) with administrative privileges. Define a password for this user account and make sure that **User Must Change Password At Next**

Logon is *not* selected. On AIX, such a user account is created automatically, thus there is no need to create it manually.

HTTP Server – Install the IBM HTTP Server. The IBM HTTP Server is included with the Trust Authority media package. On Windows NT, make the user account, defined in the previous step, the owning user of the HTTP Server service. On AIX, make sure to also install the HTTP Server SSL Module.

WebSphere Application Server – Install WebSphere Application Server. Choose HTTP Server 1.3.3 as the Web server in the configuration. Ensure that the HTTP Server is not started automatically at system startup. Trust Authority starts and stops the HTTP Server as needed. On Windows NT, change the *start* property in the Services folder of the Control Panel. On AIX, remove any entries in /etc/inittab that would start the HTTP Server at system restart.

Acquire installation user credentials – On Windows NT, log on as the newly created user (see above).

Install Trust Authority – On AIX, installation of Trust Authority is most conveniently done through SMIT. On Windows, the installation is done with an InstallShield guided process. (At the time of writing, known error or warning messages may show up on AIX, which are not critical. See the installation documentation.) On AIX, select the server components to install (most likely all except 4758 support). Do not install 4758 support unless such a cryptocard is physically installed in the system. On Windows, InstallShield gives you the option to select different servers. You may choose to not install the client and the RA Desktop if you want to run them on another machine.

After the software installation, three more steps need to be performed to achieve a basic configuration of Trust Authority: a post-installation configuration process, a Setup Wizard for customization, and a customization program.

Run the post-installation process – A post-installation process must be run after all software components have been installed. This configures and initializes the database and the Trust Authority components. Make sure that the environment variable %TEMP% is set on Windows. Run the `CfgPostInstall` program (on Windows, this can be done through **Start** -> **Programs** -> **IBM SecureWay Trust Authority** -> **Post Installation Configuration**). Follow the configuration messages. Although there might be warnings or even errors shown, you should watch for a “successful” indication at the end.

Run the Setup Wizard – The *Setup Wizard* is an applet that runs in a Web browser. For performance reasons, it is recommended that this should be run on another machine. To run the Setup Wizard, point the browser to:

```
https://<TA server name>:81
```

(Notice the http*s*, rather than http.) Carefully follow the instructions on that page to download and install the swingall.jar file. Then, on the same page, click **CfgSetupWizard.html**. Be patient, it may take a while to download the Setup Wizard. The Setup Wizard

guides you through a series of dialogs:

- The Trust Authority server name, port numbers, and DN. The information given in the DN will be used later as identifying information contained in the self-signed CA certificate (root certificate). The ports should be left as default unless there are strong reasons not to accept them.
- Selection of encryption defaults and key lengths. Unless there are reasons not to do so, you should accept the defaults. A key length of less than maximum (1024) should be considered carefully.
- Information about the LDAP directory server. This includes directory root and administrator information and passwords. If the LDAP directory is on another machine, and/or the directory is already configured, refer to the document *Using the SecureWay Directory With Trust Authority*, available at the Web site mentioned in the introduction to this section.
- A *registration domain name* that is the name for your security domain along with registration domain language. The registration domain name must be unique and must not contain any blanks or other special characters (see the installation documentation for details). The registration domain name will become part of the URL that users will need to access this Trust Authority server.
- Information (name and port) about the Web server running on the system to access the various Trust Authority functions. It is recommended to accept the defaults unless there are strong reasons not to do so.
- A port number for TA Client accesses.

Upon completion, the Setup Wizard shows a summary and then stores the entered information on the Trust Authority server.

Run the customization program – The last step in setting up Trust Authority is to run the `CfgStart` command that does the final configuration. This command is run automatically after finishing the Setup Wizard. There might be warning or error messages produced during the execution of `CfgStart` and logged in the log file. Make sure that it says “successful” at the end of the process. You can view the log files. On AIX, log files are stored in /usr/lpp/iau/logs, and on Windows NT, they are in <install_dir>\logs, where <install_dir>, by default, is C:\Program Files\IBM\Trust Authority.

This completes the installation and basic configuration of Trust Authority. The configuration process includes the necessary configuration for the HTTP Server (including SSL) and the LDAP directory (unless an existing LDAP directory is being used, in which case, there are some additional, manual steps to be done). The configuration process also creates a self-signed Trust Authority CA root certificate that will be required in Web browsers to communicate securely with Trust Authority.

After completion of the installation, it is recommended that you run a quick test (see the next section) before the Trust Authority server is stopped and restarted.

6.4.2 Installation test

After the installation and basic configuration are completed, a simple test can be performed to see the proper function of Trust Authority. The test is based on the fact that Trust Authority, by default, auto-approves certain certificate requests. This way, no administrator intervention is necessary to create and issue a certificate. Since Trust Authority automatically approves Web browser certificates with a validity period of one year, the Trust Authority installation, at this point, is ready to approve and issue such certificates. (This auto-approval will be discussed later in this chapter.)

To test this, point your browser at:

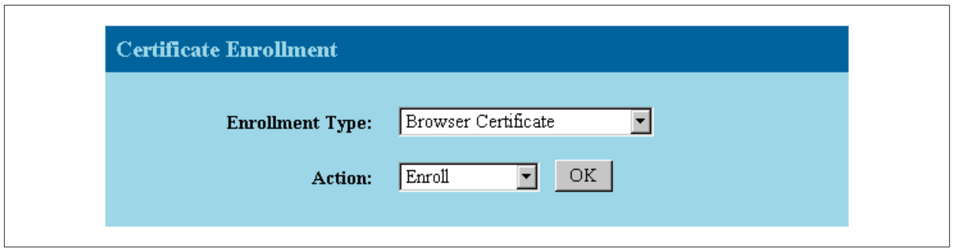
```
http://<TA server name>/<security domain name>/index.jsp
```

For example, if the TA server name is *alpha.beta.gamma.com*, and the security domain name (as defined through the Setup Wizard before) is *MyDomain*, the URL would be:

```
http://alpha.beta.gamma.com/MyDomain/index.jsp
```

The upcoming page has a link for downloading the Trust Authority’s self-signed root certificate (CA certificate). This should be done as the first step because Trust Authority is a new CA that the browser needs to be made aware of. Downloading the root certificate, however, is not mandatory but recommended in order to avoid unnecessary “new site” pop-up warnings from the browser on subsequent connections to the Trust Authority server.

At the bottom of the same Web page, there is a Certificate Enrollment dialog box as shown in Figure 21. Accept the defaults to enroll for a **Browser Certificate** and click **OK**.



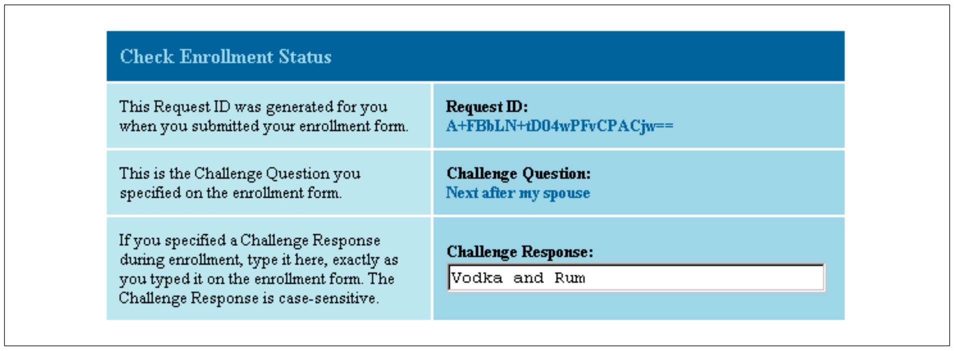
Enrollment Type: Browser Certificate

Action: Enroll

Figure 21. Certificate enrollment dialog (partial page)

The next page (notice that this is now a secure HTTPS connection to the Trust Authority server) presents the default enrollment form for a certificate. Do not change the certificate type because other certificate types may not be auto-approved. It should be a 1-year Web client authentication certificate. Fill in the information as requested and click **Submit Enrollment Request** at the bottom. The browser will now create a key pair (you may be informed by a pop-up message), of which the public key is being included with the certificate request, and the private key remains local in a password-protected file.

The next page that comes up in response to the certificate request is important because it contains the request ID that was generated for your request. If you entered a challenge question and response on the previous enrollment dialog, you will be required to enter the challenge response here. The bottom part of that page is shown in Figure 22.



This Request ID was generated for you when you submitted your enrollment form.	**Request ID:** A+FBbLN+tD04wPFvCPACjw==
This is the Challenge Question you specified on the enrollment form.	**Challenge Question:** Next after my spouse
If you specified a Challenge Response during enrollment, type it here, exactly as you typed it on the enrollment form. The Challenge Response is case-sensitive.	**Challenge Response:** Vodka and Rum

Figure 22. Check enrollment status form (partial page)

After clicking **Check Enrollment Status** at the bottom of the page, the certificate will be automatically downloaded to the browser, and the text on the upcoming page will indicate this. If this is not the case, you may have been too fast; allow Trust Authority a little time to create, sign, and issue the certificate.

The correct receipt of the certificate can be verified through the configuration settings of the browser. With Netscape Communicator, for example, click the **Security** icon and then select **Yours** in the certificates list. If everything worked fine, the new certificate appears in the list. Notice when viewing the details of the new certificate that the Trust Authority is displayed as the CA that issued this certificate.

If the test was not successful, there might be several reasons. First, ensure that the test was done correctly and there was no obvious indication of an error. Then, check the log files in Trust Authority to see whether they indicate a point of possible misconfiguration. If all problem determination fails, consider reinstalling Trust Authority from scratch.

If the test was successful, it is recommended to stop Trust Authority, restart the system, and rerun the same test. This ensures that the configuration, as well as the start and stop procedure, works correctly. Two commands, `Start_TA.sh` and `Stop_TA.sh`, are provided on AIX for starting and stopping Trust Authority (you must be logged in as *cfguser* when running these commands). On Windows NT, the commands are `Start_TA.bat` and `Stop_TA.bat`, respectively, and they can be run from a command line or through the Start menu. Refer to the installation documentation for additional considerations when starting and stopping Trust Authority.

6.4.3 Installing the RA Desktop

The RA Desktop is supported on Windows 95/98/NT and runs as an applet in a Web browser. The installation of the RA Desktop is straightforward by using the InstallShield mechanism. You can also refer to the *Installing the Trust Authority RA Desktop* document available online on the Trust Authority library Web site.

The RA Desktop client module must be installed on the client system where the RA Desktop is to be run. The installation module *RADInst.exe* is shipped with the TA package, and the client system requirement is discussed in 6.3.2, “Client environment” on page 121.

The InstallShield installation process requires two pieces of information:

- The directory path for the installation

- The URL for the RA server

The default installation path for the installation of the RA Desktop is C:\Program Files\IBM\Trust Authority\RA Desktop. The URL for the RA server must be constructed as follows (assuming the default port 1443 is used):

```
https://<RA server>:1443/<registration domain>
```

For example, if the RA server is located on *alpha.beta.gamma.com*, and the security domain name is *MyDomain*, the URL is (default port assumed):

```
https://alpha.beta.gamma.com:1443/MyDomain
```

The RA server URL can be changed later by running the InstallShield again by selecting **Start** -> **Programs** -> **IBM SecureWay Trust Authority** -> **RA Desktop Configuration**.

You may have noticed that the RA Desktop uses port 1443 on the RA server machine. This is a port configured into the HTTP Server that requires client certificate (SSL) authentication. In other words, before the RA Desktop can be run by an authorized RA administrator, this RA administrator must have a valid certificate that is being used for authentication during the SSL session setup. Moreover, the RA administrator (or certificate owner) must have been authorized to run the RA Desktop in Trust Authority. These two requirements are met by following the steps.

In order to get a valid certificate for the RA administrator, follow the steps described in 6.4.2, “Installation test” on page 126, to request a 1-year browser certificate, that is, the person who is going to be an RA administrator must point his or her browser at the URL below and follow the instructions:

```
http://<TA server name>/<security domain name>/index.jsp
```

This must be done from the Web browser that will be the RA administrator’s browser because the certificate will be tied to that browser and cannot be used from any other browser. It is important to request a 1-year browser certificate because the auto-approve process comes in handy. Other certificate types would require administrator approval, which is not possible at this time. If you have already run the test as previously described and want to use the certificate that was generated, there is no need to request another certificate at this time.

Then, this RA administrator (the user who requested the certificate in the previous step) must be authorized to run the RA Desktop. To do this, the *request ID* of the certificate request as returned from Trust Authority must be retained and used in the next step. The request ID looks like a random character string with two equal signs (==) at the end.

Next, go to a command prompt and change to the directory where the `add_rauser` command is located. Enter the following command:

```
add_rauser <path>/domain.cfg <security domain name> <request ID>
```

For example, if your security domain name is *MyDomain*, the command on AIX could look like:

```
add_rauser /usr/lpp/iau/pkrf/etc/domain.cfg MyDomain eDgf4dqdbkKd8Hhs==
```

And on Windows NT:

```
add_rauser c:\Program Files\IBM\Trust Authority\pkrf\etc\domain.cfg
MyDomain eDgf4dqdbkKd8Hhs==
```

(The above represents one single line.) This command adds the user whose certificate credential UUID was added to the list of RA administrators.

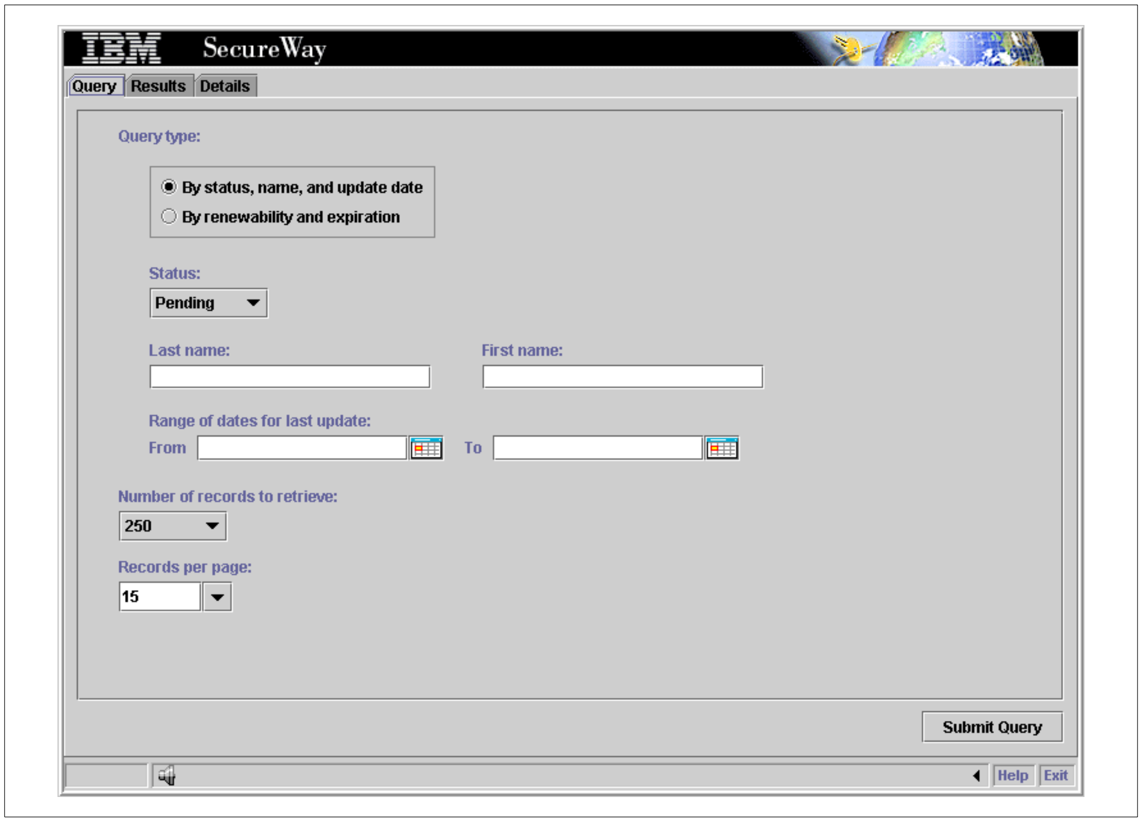
If you do not have the request ID from the certificate request for some reason, the `credential_uuid` can be used with the `add_rauser` command instead. To get the `credential_uuid`, you need to locate a certain record in the Trust Authority database. As the user with access privileges to DB2 (cfguser by default), open a DB2 command session and enter the following commands:

```
db2> connect to pkrfdb
db2> select last_name, credential_uuid from requests
```

Depending on the number of requests in the database, you may have to limit the number of output records with additional *where* clauses (see the *Trust Authority System Administration Guide*). Locate the correct request by reference to name and time, and note the credential_uuid associated with that request. Then, run the `add_rauser` command as described above with the credential_uuid instead of the request ID.

After this configuration step, the RA administrator can start the RA Desktop from the very same browser from which the certificate was requested (and then received). To start the RA Desktop, select **Start** -> **Programs** -> **IBM SecureWay Trust Authority** -> **RA Desktop**.

Because client certificate authentication is now required, the browser will ask for a certificate and a password to open the local certificate database (if not configured otherwise). Allow some time for the applet to load, and the RA Desktop, shown in Figure 24 on page 132, appears.



Query type:

- [✓ By status, name, and update date]
- By renewability and expiration

Status: Pending

Last name:

First name:

Range of dates for last update:

From

To

Number of records to retrieve: 250

Records per page: 15

Figure 23. RA Desktop (initial screen)

Through the RA Desktop, the RA administrator approves certificate requests, revocations, and performs other actions on certificates. Please see the *IBM SecureWay Trust Authority Registration Authority Desktop Guide*, available online, for more details.

6.4.4 Installing the TA Client

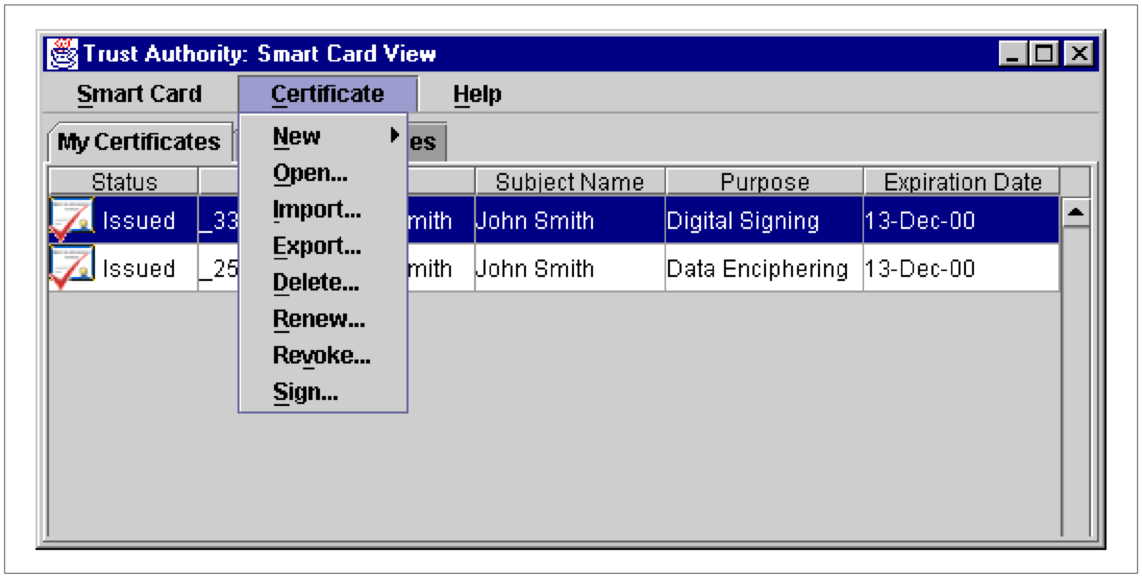
Installing the TA Client is supported by an InstallShield process. For hardware and software prerequisites, please refer to 6.3.2, “Client environment” on page 121. The TA Client installation is also described in the online document *Installing the Trust Authority Client Application*.

The TA Client module, *TACInst.exe*, is a self-extracting program, available on the product media. It uses the Windows InstallShield for installation and configuration. Upon installation, InstallShield requires the following information:

- The directory path for the installation
- A password for a Smartcard

The default installation path for the installation of the TA Client is C:\Program Files\IBM\Trust Authority. If you change this path, make sure that you follow the special procedure outlined in the installation documentation. The password for a Smartcard is actually for a virtual Smartcard being created by the TA Client or for a physical Smartcard (the version of the TA Client at the time of writing supported the IBM Gemplus Smartcard only).

After installation, the TA Client can be started via **Start** -> **Programs** -> **IBM SecureWay Trust Authority** -> **Trust Authority Client**. Upon successful installation, the TA Client starts with a prompt for the Smartcard password and then presets itself with the interface shown in Figure 24. Note that the example shown in Figure 24 already lists two certificates that have been issued and imported into the TA Client for this particular user.



Status			Subject Name	Purpose	Expiration Date
Issued	_33	mith	John Smith	Digital Signing	13-Dec-00
Issued	_25	mith	John Smith	Data Enciphering	13-Dec-00

Figure 24. TA Client user interface

The TA Client allows a user to manage his or her certificates and to request and receive certificates from the Trust Authority CA. In Figure 24, the Certificate pull-down menu is shown to give you an idea of the functions a user can perform with the TA Client. The TA Client's functions are self-explanatory to a large extent, but you can also read the *Trust Authority User's Guide* (available online) for more information.

6.5 The user’s view of Trust Authority

The most common users of a Trust Authority system are likely to be end users, or End-Entities (EEs). They need the CAs signature to sign their public keys, thus creating their certificates. They may need different types of certificates for the different purposes. End-Entities also rely on the TA central directory for their and others certificates.

The following two sections outline how an End-Entity (EE) normally interacts with the TA system. TA provides two types of user interfaces: enrollment Web pages and the TA Client application. Only the enrollment Web page can be used to enroll a certificate. Both the enrollment Web page and the TA Client application can then be used to manage certificates. The TA Client application provides more management functions and the enrollment Web page can only manage browser certificates. This redbook does not intend to describe all the details of using the TA system and how different Web browsers can handle certificates. However, some important steps and features of TA are pointed out in the following sections so that readers can have a basic concept about the functionality of the system. For more detailed information concerning the usage of TA, please refer to the documentation that is available online on the IBM Web site listed in the introduction to this chapter.

6.5.1 Using the enrollment Web page

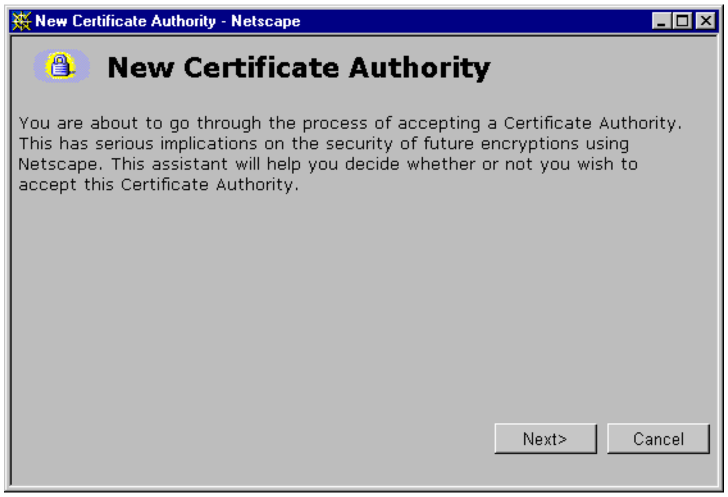
To use the Trust Authority system, a user must make himself or herself known to the system. This is achieved by first applying for a certificate. With Trust Authority, certificates have to be enrolled by using a Web browser. The URL for the enrollment page is:

```
http://<RA server>/<domain name>/index.jsp
```

Where <RA server> is the name or IP address of the RA server, and <domain name> is the name of the registration domain managed by that RA system that the user belongs to. The registration domain name has been configured during the TA configuration. A user needs to know the URL in order to register for a certificate.

The default Web page that TA presents upon connecting to the URL contains a link to install the CA certificate. Subsequent Web pages will use secure HTTPS connections to the RA server, and thus, that CA certificate should be installed to avoid further security pop-up messages from the browser. Installing the CA certificate basically tells the browser to allow secure connections to that server without warning pop-up screens (unless the browser is configured in a different manner).

When clicking the link to install the CA certificate, the browser will pop up a screen asking if the user wants to accept this new CA. Figure 25 shows such a pop-up screen from Netscape Communicator.



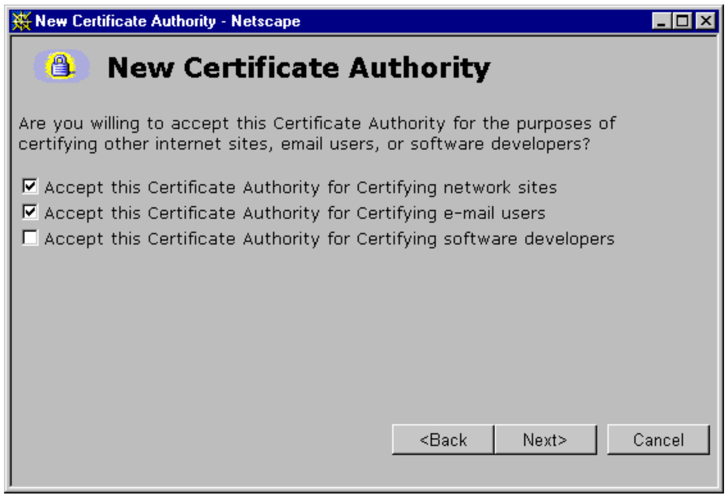
New Certificate Authority

You are about to go through the process of accepting a Certificate Authority. This has serious implications on the security of future encryptions using Netscape. This assistant will help you decide whether or not you wish to accept this Certificate Authority.

Figure 25. Accepting a new CA

The user can keep on clicking the **Next>** button to look at more screens. In one of these screens, the user can choose to look at the CAs certificate and decide if this CA can be trusted and added to the browser's trust list.

Another screen asks the user for the purpose of this CA as shown in Figure 26 on page 135.


New Certificate Authority - Netscape

New Certificate Authority

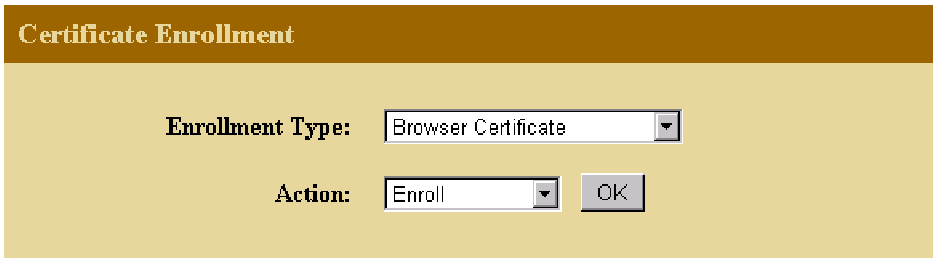
Are you willing to accept this Certificate Authority for the purposes of certifying other internet sites, email users, or software developers?

- [✓] Accept this Certificate Authority for Certifying network sites
- [✓] Accept this Certificate Authority for Certifying e-mail users
- [ ] Accept this Certificate Authority for Certifying software developers

Figure 26. Selection of purposes for new CA

The user should choose the options that are applicable to him or her, most likely the ones shown in Figure 26. After a few more pop-up screens, the user may click the **Finish** button to eventually add this CA certificate to the browser, or click **Cancel** if he or she does not want to trust this CA.

Still on the same Web page, there is a dialog box at the bottom as shown in Figure 27.


Certificate Enrollment

Enrollment Type: Browser Certificate

Action: Enroll

Figure 27. Enroll for a browser certificate in the enrollment Web page

The three available enrollment types (see Figure 27) are:

Browser Certificate – To enroll for a Web browser (SSL) certificate that enables a user to authenticate to a Web server by using this certificate if client authentication is required.

Server or Device Certificate – To enroll for a server or device certificate. By default, a server or device certificate must be approved by an RA administrator.

Certificate Preregistration – To place a preregistration of any type of certificate. As with server or device certificates, an RA administrator will need to approve preregistration requests, unless configured for auto-approve.

Depending on the selected enrollment type as explained above, the second selection list in Figure 27 on page 135 offers two or four possible actions:

Enroll – To enroll for any of the enrollment types . This is the first action a user would normally choose to request (enroll for) a certificate.

Check Status – To check the status of an earlier enrollment. On the subsequent page, the user will have to enter a *request ID* and, optionally, a *challenge response* to identify a request of which the status is being checked. Both, the request ID and the challenge response, must have been obtained from a previous enrollment.

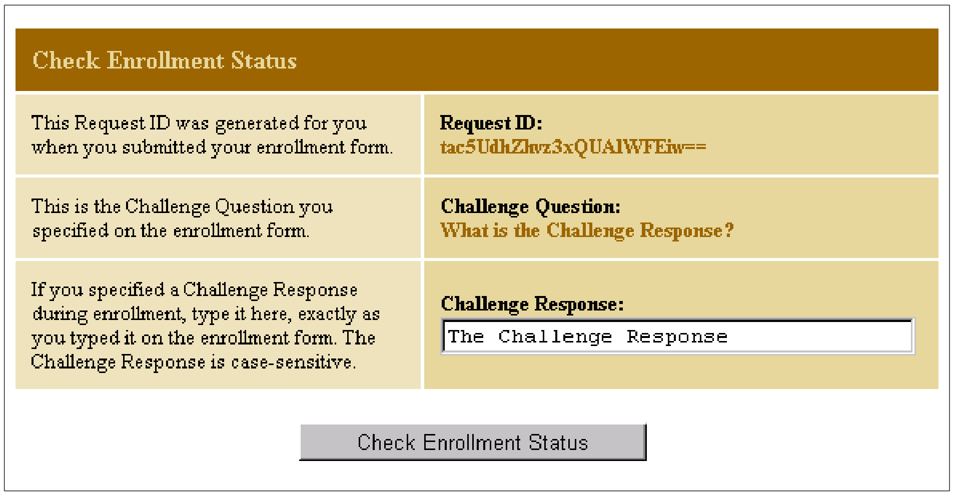
Renew – To renew a browser certificate. When selecting this action, client authentication will be initiated, requesting the user to present a certificate. This will be the certificate to be renewed on the subsequent page.

Revoke – To revoke a browser certificate. As for renewal, a client authentication process identifies the user's certificate that needs to be revoked. The subsequent page presents some information for verification and allows the user to revoke his or her certificate by a click with the mouse.

6.5.2 Requesting a certificate

Let us assume in the following the user is about to enroll for a new browser certificate, thus selecting **Browser Certificate** and **Enroll** in the dialog shown in Figure 27 on page 135. After clicking **OK**, the certificate enrollment form appears. Note that this is now a secure HTTPS connection to the RA server. The enrollment form is a default form provided with Trust Authority that can be customized to meet special requirements. On that form, the user can select from a list of applicable certificate types (browser or e-mail, one or two years expiration period) and fill in the required and optional information. For example, an optional challenge question with a response that only the user knows can be entered in order to increase the security when dealing with this request later on. The request is finally submitted by clicking **Submit Enrollment Request**. Trust Authority responds with a confirmation page that contains some further instructions on how to proceed. There is also a dialog included at the bottom of that confirmation page as shown in Figure 28 that contains the *request ID* generated by TA. The request ID uniquely identifies this request. It is important for the user to note this request ID because it will be necessary in order to check the enrollment status and/or to download the certificate. Fortunately enough, this Web page contains that request ID in its URL, so it can easily be bookmarked. Thus, if the user carries on with this form, it might not be necessary to write down the request ID or bookmark the page, though it is recommended to avoid possible problems.

If a challenge question was specified in the enrollment form, the corresponding answer has to be entered exactly the same way as on the enrollment form. Clicking **Check Enrollment Status** (Figure 28) requests TA to return status information pertinent to this enrollment.


Check Enrollment Status

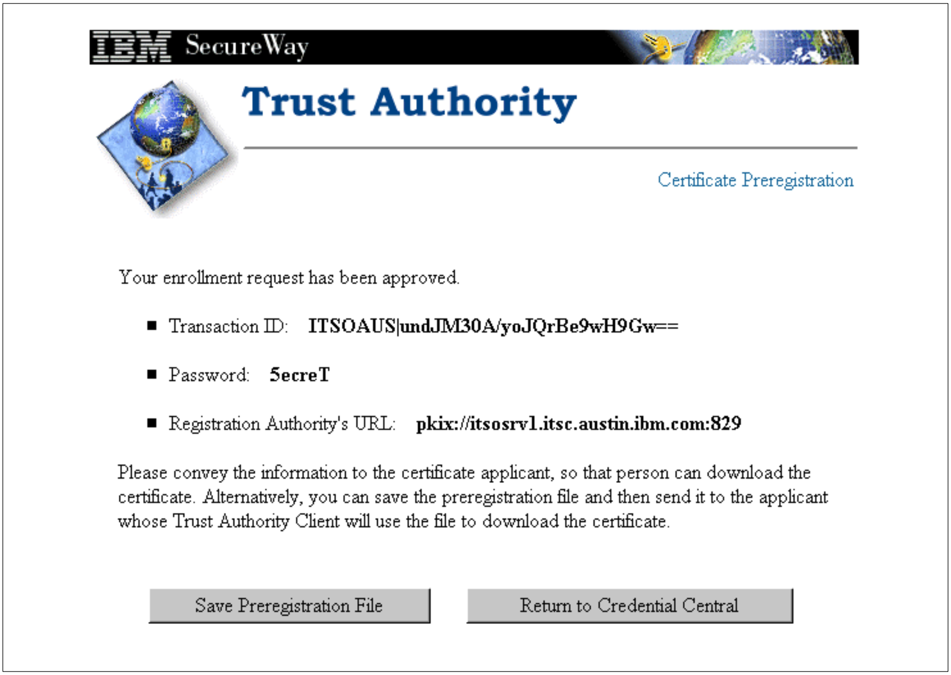
This Request ID was generated for you when you submitted your enrollment form.	**Request ID:** tac5UdhZhvz3xQUAlWFEiw==
This is the Challenge Question you specified on the enrollment form.	**Challenge Question:** What is the Challenge Response?
If you specified a Challenge Response during enrollment, type it here, exactly as you typed it on the enrollment form. The Challenge Response is case-sensitive.	**Challenge Response:** The Challenge Response

Figure 28. Checking enrollment status dialog

Depending on the certificate type requested and the status of the enrollment, the response varies. In the case when an auto-approved certificate was requested and enough time has passed for the CA and RA to process the request, the certificate will automatically be downloaded to the browser and the upcoming Web page will indicate the success of this operation. Note that this process is done automatically and the user does not get involved in any complicated certificate handling procedures. The user may be asked to enter a password for a local certificate database in which the certificate will be stored.

Another possible response to the Check Enrollment Status operation might be that it says that the enrollment request is *pending*, in other words, awaiting approval from an RA administrator. In this case, an RA administrator must approve the certificate request through the RA Desktop application.

If a *certificate preregistration* has been requested on the enrollment form, the enrollment status may indicate that the request has been approved (either automatically or by an RA administrator), and the user is also presented some information that is required to download the certificate through the TA Client as shown in an example in Figure 29. See the following section 6.5.3, “Using the Trust Authority Client application” on page 139, for more information about how to proceed with such a request.



Your enrollment request has been approved.

- Transaction ID: **ITSOAUS|undJM30A/yoJQrBe9wH9Gw==**
- Password: **5ecreT**
- Registration Authority's URL: **pkix://itsosrv1.itsc.austin.ibm.com:829**

Please convey the information to the certificate applicant, so that person can download the certificate. Alternatively, you can save the preregistration file and then send it to the applicant whose Trust Authority Client will use the file to download the certificate.

Figure 29. Approved certificate request

If a *server or device certificate* has been requested and approved, the status response contains the certificate and some options to save the certificate in a file. Section 6.5.4, “Handling server or device certificates” on page 140 explains this in more detail.

Note that if e-mail notification has been selected when the enrollment request was prepared, the user will also be notified by e-mail upon approval of a certificate request.

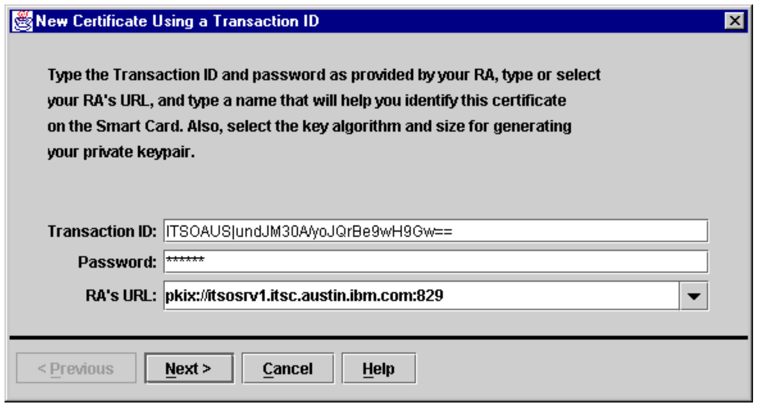
It should also be noted at this time that the enrollment Web page can also be used for certificate renewal or revocation by simply choosing a different action (see Figure 27 on page 135). Subsequent dialog pages are self-explanatory and, therefore, not further detailed here.

6.5.3 Using the Trust Authority Client application

As we have seen above, a browser (SSL) or e-mail (S/MIME) certificate requested by a user through a Web browser is automatically downloaded to the browser when the user checks the status of his or her enrollment. In the case of a certificate preregistration, the user will have to use the TA Client application to get the certificate from TA. An approved *preregistration* can be thought of a permission for a user to get a certificate. The preregistration can be requested by a user or by someone else, for example, a security administrator. If auto-approve is not configured for the type of certificate being preregistered, an RA administrator will have to approve the preregistration, just like a normal certificate request.

Trust Authority responds to an approved certificate preregistration with a page as shown in Figure 29 on page 138. TA returns a transaction ID, a password, and the RA PKIX URL. The information on this page must be used with the TA Client application in order to get the actual certificate from TA. Carrying the information over to the TA Client can be done either by manual typing (or copy and paste operations), or through the use of a preregistration file.

The TA Client application can be thought of a user and management interface for keys and certificates stored on a Smartcard. The TA Client supports virtual Smartcards that function like real, physical Smartcards, but are kept in files on the local disk storage. Through the provided preregistration information, the user can then request and download the certificate from Trust Authority. Figure 30 on page 140 shows the corresponding dialog window of the TA Client.



Type the Transaction ID and password as provided by your RA, type or select your RA's URL, and type a name that will help you identify this certificate on the Smart Card. Also, select the key algorithm and size for generating your private keypair.

Transaction ID: ITSOAUS|undJM30A/yoJQrBe9wH9Gw==

Password: ******

RA's URL: pkix://itsosrv1.itsc.austin.ibm.com:829

Figure 30. Requesting a certificate using the TA Client

Upon completion and submission of the dialog shown in Figure 30, the TA Client creates a public/private key pair in order to assemble a certificate request that is sent to the RA. This process then runs in the background, transparently to the user, and after a short time, the certificate will be downloaded to the Smartcard and listed in the TA Client's main window (see Figure 24 on page 132).

Similar to the enrollment Web page, the TA Client application supports other certificate management tasks, such as certificate renewal and revocation. In addition, the TA Client can delete certificates or pending requests, import/export a certificate between a Smartcard and a file, and view certificate properties.

The use of the TA Client for users who understand the purpose of certificates is self-explanatory to a large extent. Information can also be found in the *Trust Authority User's Guide*, which is available online from the Trust Authority library Web site.

6.5.4 Handling server or device certificates

The handling of *server or device certificates* is slightly different because the originating certificate request is normally created outside Trust Authority. TA is then used to process the request and issue certificates, which must then be forwarded to the respective server or device. The point where the originating request is created may be a Web server, an application server, or a networking device. It is important to understand that this is also the point where the generation of the public/private key pair takes place.

In general, the process for obtaining server or device certificates is as follows:

1. Most likely, the TAs CA root certificate needs to be imported into the server or device.
2. Using utilities provided with the server or device, a public/private key pair needs to be created that will be embedded in a certificate request.
3. The certificate request from step 2 needs to be forwarded to Trust Authority. This can be done through a Web interface provided by TA.
4. An RA administrator must approve the pending certificate request (unless the requested certificate type is configured for auto-approve). Upon approval, TA creates the certificate.
5. The certificate needs to be downloaded from the RA and imported into the server or device.

Let us look at a popular example: a Web server. The *IBM HTTP Server powered by Apache* is such an example that supports server and client authentication using certificates. Note that a *server or device certificate* is only necessary for server authentication, while a *browser certificate* (as described in 6.5.2, “Requesting a certificate” on page 136) is required for client authentication. The former, server authentication, is quite common for secure HTTPS connections, while client authentication using certificates is (still) rare.

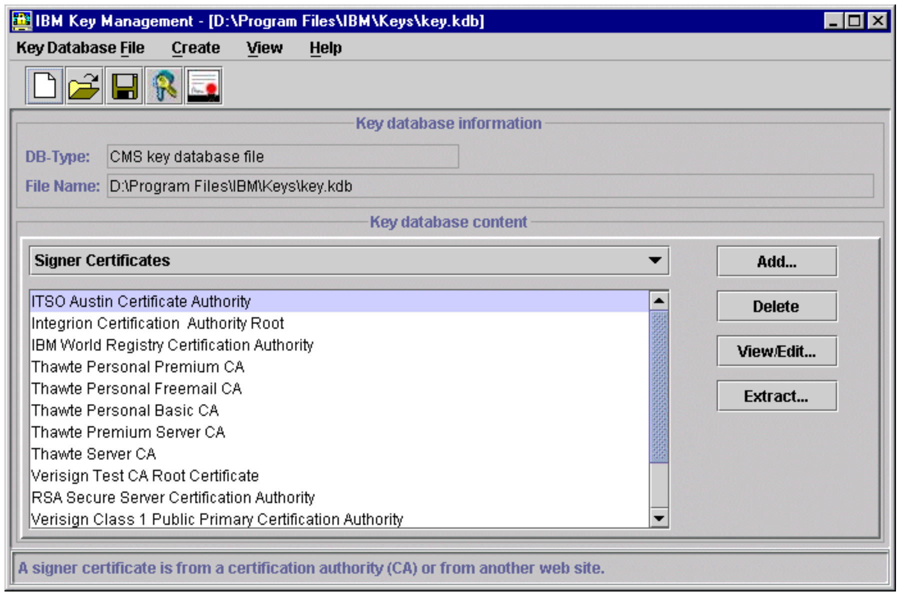
Web servers and clients use SSL for secure HTTPS communication. The IBM HTTP Server (IHS) uses the IBM Global Security Kit (GSKit) for the support of SSL. The IBM GSKit is supported on IBM AIX, Windows, and other platforms. The support for SSL is an IBM enhancement to the popular Apache server, which is an open source product that, at the time of writing, does not support SSL due to export regulations that limit the use of cryptographic code.

GSKit supports most of the functions involved in certificate handling, while IHS needs some customization in order to utilize these features. GSKit supports SSL interfaces for secure communication and certificate management functions by means of a graphical user interface. In fact, all certificate handling in respect to Trust Authority is done with GSKit and it is only a matter of customization for the IHS to correctly use the SSL services provided by GSKit. Note that GSKit is used in many other IBM products as well, thus the following description applies not only to IHS. Also, it should be mentioned that GSKit has a feature to create self-signed certificates that could be used for SSL connections; however, it is certainly not desirable for an overall security solution that each server uses its own, self-signed root certificate. Therefore, GSKit self-signed certificates may be used (and are sometimes handy) in small test environments only.

The following is an overview description of the steps involved in getting a TA server or device certificate for an IHS Web server using IBM GSKit.

First, using the graphical user interface of GSKit, open an existing, or create a new, keyring file. You will notice that GSKit already comes with a number of well-known, trusted Certificate Authorities, represented by their root certificates.

Next, the *root certificate* of your own TA security domain needs to be imported to the list of existing trusted CAs. This is done by selecting a certificate type of **Server or Device Certificate** and the **Enroll** action on the first certificate enrollment page (see Figure 27 on page 135). On the next page, after clicking **OK**, choose the option that says **Save CA Certificate to File** to save the TA root certificate to a file of your choice. Then, with GSKit, make sure you have **Signer Certificates** selected on the main window before clicking the **Add...** button. On the upcoming dialog, choose **Binary DER data** and enter the file name of the certificate to import the root certificate.



DB-Type: CMS key database file
File Name: D:\Program Files\IBM\Keys\key.kdb

Signer Certificates

ITSO Austin Certificate Authority
Integrion Certification Authority Root
IBM World Registry Certification Authority
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
Verisign Test CA Root Certificate
RSA Secure Server Certification Authority
Verisign Class 1 Public Primary Certification Authority

A signer certificate is from a certification authority (CA) or from another web site.

Figure 31. Imported CA certificate

Figure 31 on page 142 shows the result of such an import operation where the new certificate was named ITSO Austin Certificate Authority (highlighted in Figure 31).

Next, a certificate request needs to be created by GSKit using the **New Certificate Request...** function from the **Create** pull-down menu. A dialog window then requests the information to be included in the request. The result is a PKCS #10 certificate request that is written to a local file. It is ASCII text that may look like the following example:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBuDCCASECAQAweDELMAkGA1UEBhMCVVMxDjAMBgNVBBETBTc4NzU4MQ4wDAYD
VQQIEwVUZXhhczEPMA0GA1UEBxMGQXVzdGluMQwwCgYDVQQKEwNJQk0xFDASBgNV
BAsTC0lUU08gQDRzdGluMRQwEgYDVQQDEwtJVFNPIEF1c3RpbjCBnzANBgkqhkiG
9w0BAQEFAAOBjQAwgYkCgYEAo/Ju6C9TrE36WBxWWDZyinJ5nQybNPxPujmc8NY6
X83BYz/wZDc1Qou5JjbDSpnvQu9W1BMUfA/KecrnQq62OAL/oV93f4HOVGcBnR+Q
uyk8AUXHHACZZ5kCsLSjn03CFesdEQRqI+aJgK1YtfHRYZX1xgK4FWD3HUaF+jJ5
Jb0CAwEAAaAAMA06CSqGSIb3DQEBBAUAA4GBAGLlxDBGNtjCTmGpL8zH4P0PNj21
d0Q+uG9Lrxi4ti3ZbHkwgLrJXGQ7Eghbj5NoWay2Nbfz3hBdFs7L2MHhUe7W7YeG
y/NY7d55gEMbCBOh0iyWLcs/yCJNsnKdJ5ZqFGAM+vKlka0w1pSdY0GDi7A0WR+J
BXHVyJsVtzVVy+j0
-----END NEW CERTIFICATE REQUEST-----
```

This certificate request has to be forwarded to Trust Authority. In the lower part of the enrollment Web page where the CA root certificate has been obtained (see above), there is a PKCS #10 entry field, along with other fields that override the corresponding values contained in the request. Use a copy and paste operation to copy the PKCS request into the form. Make sure to copy all lines as shown in the previous example. Fill in any other required and/or optional data as desired and send the request to TA.

Upon approval, either automatic or by an RA administrator, the TA response to a Check Enrollment Status request contains the certificate and allows you to store it in a file in various formats. Choose a format suitable for the application. For example, *Binary Raw Certificate* is a suitable format for GSKit.

On the GSKit graphical user interface, select the **Personal Certificates** view and click the **Receive...** action button to open the file selection dialog. Select **Binary DER data** as the data type and enter the correct file name and path of the certificate file just received. This imports the certificate into GSKit and completes the certificate handling operation. The imported certificate is now listed in the Personal Certificates view of GSKit.

What is left is to customize IHS in such a way that it supports SSL via GSKit and recognizes this certificate. You should make sure that IHS was installed to include SSL support, which might be a separate installable option, depending on the operating system platform. IHS comes with two default configuration files of which one contains sample SSL definitions. While complete SSL configuration explanations for IHS are beyond the scope of this book, the following are the essential directives in the configuration file that you should look after. They define a virtual Web server using the default HTTPS port 443:

```
...
Port 80
Listen 80
Listen 443
...
<VirtualHost :443>
SSLEnable
...
SSLServerCert  ITSO-Austin
...
SSLClientAuth   none
...
DocumentRoot /usr/ns-home/docs
...
</VirtualHost>
...
SSLDisable
...
Keyfile "D:/Program Files/IBM/Keys/key.kdb"
```

Note that the value given for the `SSLServerCert` directive must match the certificate name (label) in GSKit and the `Keyfile` directive specifies the keyring file created by GSKit. Note also that client authentication is not configured in the example. The `SSLClientAuth` directive supports the values `none`, `optional`, or `required`, depending on how client certificate authentication is to be utilized. Please see the IHS online documentation for more information about how to configure other options for SSL.

This concludes the example for the IBM HTTP Server. Please note that the latest versions of the IBM HTTP Server include a Web-based configuration tool that supports an administrator in doing these configurations. The basic principles and the process for the handling the newer versions of the IBM HTTP Server, or of different servers or devices, are generally the same, although minor variations may exist.

6.6 The administrator's view of Trust Authority

In the previous sections, we described the enrollment of certificates and the acquisition and management of certificates. However, the steps for verifying and approving the certificate requests have not been outlined. In fact, these are the tasks of the RA server. For Trust Authority, the RA Desktop application is designed for such purposes. The RA Desktop is a graphical user interface (GUI) secure applet for handling enrollment requests and managing the certificate records.

When using the RA Desktop, an *RA administrator* must be properly authenticated by presenting the appropriate RA administrator digital certificate. In other words, the Web server is configured for client certificate authentication when accessing the RA Desktop application. The RA Desktop runs on Windows 95/98/NT and can conveniently be started by selecting **Start** -> **Programs** -> **IBM SecureWay Trust Authority** -> **RA Desktop**, or by pointing a browser to the appropriate URL. An RA administrator must be authorized to run the RA Desktop as explained in 6.4.3, "Installing the RA Desktop" on page 128.

Upon positive authentication, the Web browser will download and initialize the RA Desktop applet as shown in Figure 24 on page 132.

The RA Desktop applet offers specific management options. In the **Query** tab page, the RA administrator can prepare a query by choosing or entering values for the fields provided. For example, the RA administrator can form a query to display a list of pending or approved certificates, optionally limited for a specific name and/or time period. Other options allow the administrator to limit the number of records returned by the query. The query page does not need much explanation since its fields are self-explanatory, and thus, its use is intuitive.

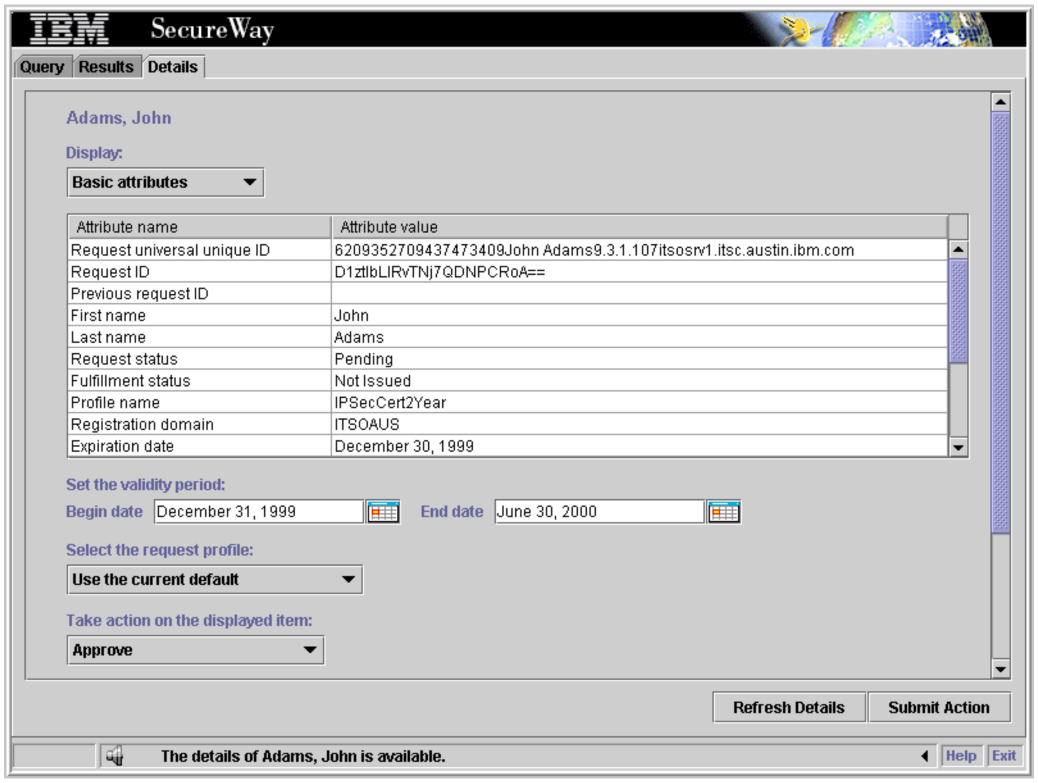
After submitting a valid query, the **Results** tab page is displayed with a list of matching records. This page provides several usability options for sorting (by clicking a respective column heading), scrolling, and column resizing. Details about a single certificate record can be viewed by double-clicking a record or by selecting that record and either clicking the **Details** tab or on the **Show Details** button.

In addition to viewing the records, the RA administrator can, more importantly, take certain actions on single or multiple records. Some common and important actions include:

- Approve pending requests – This is likely one of the most often used functions where an RA administrator approves a pending certificate request.
- Keep requests in pending status – An administrator can decide to keep a pending request in the pending status, for example, if additional verifications need to be performed.
- Reject requests – If a request was made in error, or the administrator cannot approve it for any other reasons, he or she can reject a request. The user will be notified about the rejection when checking the status of the certificate request.
- Specify another request profile for a certificate request – An administrator can specify another request profile. For example, the administrator can change a user's request for a 2-year certificate to a 1-year certificate.
- Revoke certificates – This revokes a certificate.
- Change renewability of certificates – Certificates can be flagged to be either renewable or not. This status can be changed.
- Change the validity period of certificate requests – An administrator has the option to alter the requested validity period of a certificate.
- Change an attribute value – A few processing attributes can be changed, such as the comment to user attribute.
- Add a comment to requests or certificates – Comments can be added by an administrator to the actions performed.

The procedure for the above actions is, in principle, the same for all: one (or more) certificate record must be selected in either the Results or the Details view of the RA Desktop. Then, an appropriate action must be selected from the selection lists in the lower part of the page, and that action will be initiated by clicking **Submit Action**. Optionally, comments can be added to such actions by typing into the comments field.

Figure 32 on page 147 shows an example screen shot taken as the RA administrator approves a certificate request. Note that in the upper part of the window, the details of the certificate request are displayed, while the **Approve** action has been selected in the actions list underneath. Also, in this example, the administrator has changed the validity period of the requested certificate to December 31, 1999 through June 30, 2000.



Adams, John

Display: Basic attributes

Attribute name	Attribute value
Request universal unique ID	6209352709437473409John Adams9.3.1.107itsosrv1.itsc.austin.ibm.com
Request ID	D1ztlbLIRvTNj7QDNPCRoA==
Previous request ID	
First name	John
Last name	Adams
Request status	Pending
Fulfillment status	Not Issued
Profile name	IPSecCert2Year
Registration domain	ITSOAUS
Expiration date	December 30, 1999

Set the validity period:
Begin date December 31, 1999 End date June 30, 2000

Select the request profile: Use the current default

Take action on the displayed item: Approve

The details of Adams, John is available.

Figure 32. Approval of a certificate request

The use of the RA Desktop is intuitive once the basic capabilities and functions as listed above are understood. Further information can be found in the *Registration Authority Desktop Guide*, available online at the Trust Authority library Web site.

6.7 Customizing and managing Trust Authority

The previous sections explained the use of Trust Authority from both a user's and an RA administrator's view in a given, standard installation. Customizing and managing a Trust Authority system involves more than just managing certificates.

This section explains the tasks involved in customizing and managing the TA system. However, only a high-level description is provided since it is not the intention of this book to replace the *Trust Authority Configuration Guide* and the *Trust Authority System Administration Guide* that are available online at the Trust Authority library Web site. Please refer to these publications for more detailed information about the tasks described here.

Customizing the TA system includes the following tasks:

- Defining alternative TCP/IP communication ports for the components
- Changing the default certificate templates
- Changing the default certificate enrollment forms
- Adapting some default operational behavior of TA

Managing the TA system includes the following tasks:

- Starting and stopping the components
- Changing component passwords
- Verifying the proper function of the components
- Working with the audit subsystem

These tasks are further explained in the following sections.

Note

Changing any operational parameters of Trust Authority should be done with extreme care. It is highly recommended that configuration changes be performed and tested in a separate test environment prior to applying them to a production environment. Some changes, such as changes to the certificate formats, are beyond a pure technical nature and may affect the overall security if not understood properly.

6.7.1 Defining alternative TCP/IP ports

Most TCP/IP communication port numbers of the TA components can be changed (see the *Trust Authority System Administration Guide*). This should not normally be done but may be convenient, for example, in a test environment to better separate multiple instances of an installation.

Figure 33 on page 149 shows the default ports assigned during the installation. Note that most ports used in TA adhere to corresponding standards and should, therefore, not be changed without thorough understanding.

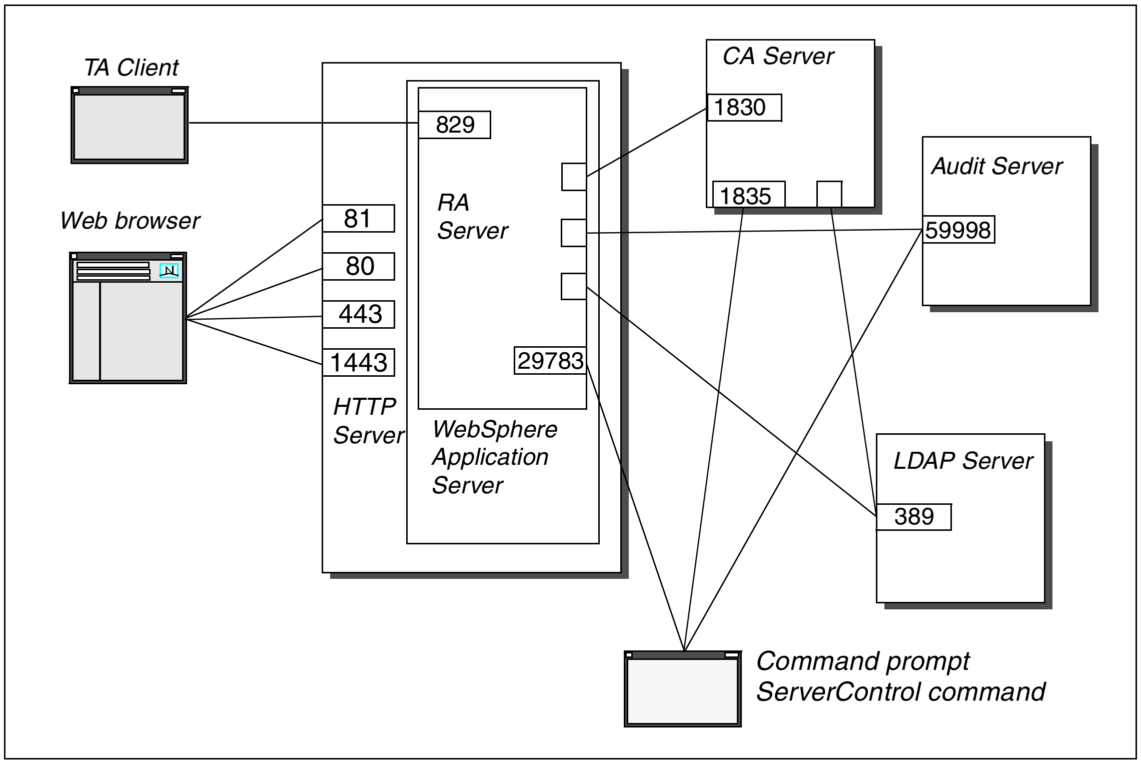


Figure 33. Trust Authority TCP/IP port assignments

Note that port 81 is used only temporarily during configuration and will not be used thereafter in normal operation. It should also be noted that Figure 33 does not list all ports used by Trust Authority.

Ports should not be changed unless really required. The best way to use different ports is to plan ahead and define these ports during the customization of TA when using the Setup Wizard (see 6.4.1, “Installation and basic configuration” on page 123). If changes have to be made after initial customization, configuration files need to be edited and Trust Authority needs to be restarted. The following files are of interest in regards to port assignments:

- <install_path>/etc/TrustAuthority/jonahca.ini

  The jonahca.ini file contains configuration parameters for the CA. Among others, it defines the CAs listener port (default 1830) in the Transport section. It also contains an entry that holds the RAs listener port (default

829) in the URLs section. The following shows the relevant sections from the jonahca.ini file with the port numbers printed in bold face:

```
...
[Transport]
TCPPort=1830
...
[URLs]
/C%EQ%US/ST%EQ%Texas ... RA=pkix://ta.itso.austin.ibm.com:829
...
```

- <install_path>/pkrf/Domains/<domain>/etc/jonahra.ini

  The jonahra.ini file contains configuration parameters for the RA. In the Transport section, the RAs listener port (default is 829) is defined, and in the URLs section it contains the CAs port (default is 1830). The following shows the relevant sections from the jonahra.ini file with the port numbers printed in bold face:

```
...
[Transport]
TCPPort=829
...
[URLs]
/C%EQ%US/ST%EQ%Texas ... CA=pkix://ta.itso.austin.ibm.com:1830
...
```

- <install_path>/pkrf/Domains/<domain>/etc/raconfig.cfg

  The raconfig.cfg file includes definitions that specify the RAs operational behavior. Among others, the RAs HTTP ports are defined (defaults are 80, 443, and 1443). Note that ports 80 and 443 are not explicitly defined since they rely on the HTTP Server's definitions for HTTP and HTTPS connections. The following lists the relevant lines from a sample raconfig.cfg file, where ta.itso.austin.ibm.com is the TA server name and MyDomain is the RA security domain name:

```
...
PUBLIC_SERVER_URL_PATH=http://ta.itso.austin.ibm.com/MyDomain/
UE_SERVER_AUTH_URL_PATH=https://ta.itso.austin.ibm.com/MyDomain/
UE_CLIENT_AUTH_URL_PATH=https://ta.itso.austin.ibm.com:1443/MyDomain/
...
PKIX_RA_URL=pkix://ta.itso.austin.ibm.com:829
...
RAD_CLIENT_AUTH_URL_PATH=https://ta.itso.austin.ibm.com:1443/MyDomain/
...
```

- <install_path>/pkrf/Domains/<domain>/etc/httpd-RD_0.conf

  The httpd-RD_0.conf file is the configuration file for the IBM HTTP Server. It contains the definitions of the listener ports on which the RA applets are listening. The defaults are 80, 443, and 1443. The following are the relevant directives from a sample HTTP Server configuration file, where 1.2.3.4 is the server's IP address and the relevant port numbers are printed in bold face:

  ```
  ...
  Port                          80
  Listen                        1.2.3.4:80
  Listen                        1.2.3.4:443
  Listen                        1.2.3.4:1443
  ...
  <VirtualHost 1.2.3.4:80>
  ServerName                    ta.itso.austin.ibm.com
  ...
  </VirtualHost>
  ...
  <VirtualHost 1.2.3.4:443>
  ServerName                    ta.itso.austin.ibm.com
  ...
  </VirtualHost>
  ...
  <VirtualHost 1.2.3.4:1443>
  ServerName                    ta.itso.austin.ibm.com
  ...
  </VirtualHost>
  ```

  Please note that the above port numbers also appear in comments and log file names that you may want to change as well.

When changing any of the port assignments, make sure that Trust Authority is stopped and restarted to use the new ports. Also, notice that a single port, such as the CAs listener port, is normally defined at more than one location and it needs to be changed in all locations.

Before you change any of these ports, please refer to the *Trust Authority System Administration Guide* (available on the Trust Authority library Web site). This guide not only contains a detailed listing of all the configuration parameters in the above configuration files, but also lists the locations where the ports need to be changed. In addition, this guide also explains whether or not individual configuration parameters can or should not be changed in the above configuration files.

6.7.2 Changing certificate templates

The certificate types that are supported by a TA installation are defined in a file named *certificate_profiles.cfg*, which is located in the registration domain's etc directory. For example, if the registration domain is MyDomain, the file on AIX would be:

/usr/lpp/iau/pkrf/Domains/MyDomain/etc/certificate_profiles.cfg

Besides some general definitions and variable assignments (not further explained here), this file contains a section for each certificate type that is headed with the generic certificate type name in brackets ([]).

As an example, let's define a new certificate type for e-mail protection (S/MIME) that is valid for five years. A few new lines, shown in Figure 34, could then be added to the configuration file.

```
...
; --- S / M I M E  C e r t  5  Y e a r ---
[S/MIMECert5Year]
display_name=E-mail Protection (5-Year)
required_fields=first_name,last_name,email_address

; --- Define Extensions
extensions_required=keyUsage,extendedKeyUsage,subjectAltName,certificatePolicies
keyUsage=digitalSignature&keyEncipherment
extendedKeyUsage=emailProtection
subjectAltName=rfc822Name=%%MAIL%%

; subjectName
subjectName=/C=%%C%%/O=%%O%%/OU=%%OU%%/ST=%%ST%%/L=%%L%%/MAIL=%%MAIL%%/CN=%%CN%%

; --- Auto: approve or pend
decision_default=approve

; --- Define validityNotAfter values
validityNotAfter_range=0,1826
validityNotAfter_default=1826
...
```

Figure 34. New definitions for a 5-year S/MIME certificate

The definitions for a 1-year S/MIME certificate (S/MIMECert1Year) can serve as a template to be copied, with the following changes:

[S/MIMECert5Year] – This is the new generic name, also serving as a section heading in the configuration file for this certificate. The same name will be used in other configuration files. Note the brackets that serve as a section heading.

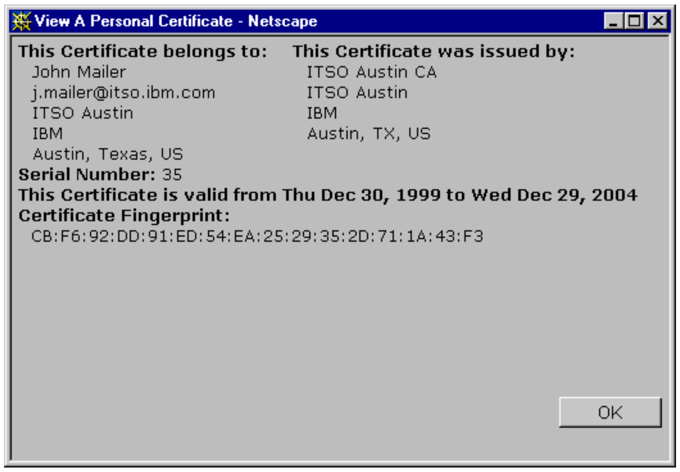
display_name – This will be the name listed in the enrollment form when the user enrolls for such a certificate. Note that additional configuration is required as discussed below.

subjectName – The subject name has been changed in this example for illustration purposes. By default, a TA-created S/MIME certificate has the e-mail address concatenated to the subject's (owner's) *common name* (CN) with a + character in between. In our example, the e-mail address will be displayed separate from the subject's common name.

decision_default – The auto-approve setting has been changed in this example to approve. Any requests for this certificate will be auto-approved without manual approval by an RA administrator.

validityNotAfter_range, validityNotAfter_default – The validity period has been changed to 1826 to represent five years in number of days (5 x 365.25).

Using the definitions as described above, a certificate for a fictitious user John Mailer would be generated as shown in Figure 35 (displayed with Netscape Communicator).



This Certificate belongs to:
John Mailer
j.mailer@itso.ibm.com
ITSO Austin
IBM
Austin, Texas, US

This Certificate was issued by:
ITSO Austin CA
ITSO Austin
IBM
Austin, TX, US

Serial Number: 35
This Certificate is valid from Thu Dec 30, 1999 to Wed Dec 29, 2004
Certificate Fingerprint:
CB:F6:92:DD:91:ED:54:EA:25:29:35:2D:71:1A:43:F3

Figure 35. Sample 5-year S/MIME certificate

Note that the subject's e-mail address appears as a separate line (caused by the altered subjectName definition) and that the validity period is actually five years.

As indicated above, additional configuration files need to be edited to make the new certificate type available. First, the RA configuration must be made aware of the new certificate type in order to list it as an available type on its enrollment Web page. This must be done in the raconfig.cfg file, located in the same directory as the certificate_profiles.cfg file as previously explained. The raconfig.cfg file defines, among other things, each available certificate type per request type. Assuming Netscape Communicator is being used, we must locate the line that starts with:

```
CertProfiles_NSBrowserSSLCert=
```

This line defines the certificate types that can be requested from a Netscape Web browser. All we need to do is to add our new certificate type, represented by its generic name S/MIMECert5Year, to that list of types. If the new certificate is to be made available for Microsoft browsers and preregistration, the generic name must be added to the respective line, too.

For the sake of completeness, the new certificate type may also be added to the list of available request profiles available in the RA Desktop. This can be done by editing the raprofiles.cfg configuration file, located in the same directory as the other two files. The new certificate type should be added to the line that starts with:

```
RequestProfiles=
```

This example has created a new certificate type. Similarly, existing types can be (carefully) altered or entirely removed. A restart of TA after the changes to the above configuration files is not necessary because these files are read each time when the corresponding service requires them.

6.7.3 Changing certificate enrollment forms and processes

Changes to the enrollment forms, other than just adding or removing certificate types as explained in the previous section or changing standard text, requires additional skills in editing HTML and/or Java Server Pages. Most likely, such a change also includes changes to the processing of data, as for example, for user input validation.

The *Trust Authority Customization Guide* (available online on the Trust Authority library Web site) contains information about the names of the individual files and how such changes can be performed. It also explains how customized programs can be added to the *RA program exit*, also called the *policy exit*, for additional processing or communication with other applications. A sample program is supplied with Trust Authority that handles the auto-approval of certificates.

The RA program exit offers a simple but powerful way to customize the registration process. Upon certain actions, the RA calls this external program and acts according to its output. The actions when the external program is being called are:

- Browser SSL certificate enrollment requests
- Browser SSL certificate renewal requests
- PKCS #10 and PKCS #7 certificate enrollment requests
- End-user initiated revocation requests
- Preregistration requests
- Trust Authority Client application-initiated renewal and revocation requests
- After a certificate is issued to a client
- During RA Desktop-initiated actions (approve, reject, pend, revoke)

The sample auto_approve program shipped with TA automatically approves all certificate requests that have decision_default=approve specified in the certificate definition profile (file: certificate_profiles.cfg) as explained in 6.7.2, “Changing certificate templates” on page 152.

The RA program exit is defined in the raconfig.cfg configuration file with the afservice.PolicyExit configuration parameter. The program name specified in this line can be changed if another program is to handle the exits. Parameters are passed to that program via standard in (stdin), and the result is expected on standard out (stdout). All parameters relevant to a certain action are passed to the external program as a single line that uses URL encoding rules (see the *Trust Authority Customization Guide* for details). This parameter line can be rather long (approximately 4000 characters); therefore, it is recommended that its contents and structure be studied carefully.

The following example is meant to give you an idea of what can be done with the RA program exit. It is a very simple UNIX shell script program that changes the auto-approval behavior of the RA slightly. The program auto-approves all certificate requests of any certificate types for any user who specifies his or her organization to be IBM and the organizational unit to be ITSO Austin, no matter what the default definition for auto-approval is for that particular certificate type. Certificate requests for all other users are handled normally, that is, they are passed on to the auto_approve program.

In order to make it work, the auto_approve program name in the raconfig.cfg configuration file must be substituted by the name of the program as listed below. The line to change is the one that defines the afservice.PolicyExit configuration parameter.

```
#!/bin/ksh
##########################################################################
# Sample script for forced approval of certificate requests from users
# from organization 'IBM' and organizational unit 'ITSO Austin'.
# All other users are handled through the TA-supplied 'auto_approve'.
# To be used for testing purposes only.
##########################################################################

Def_O="IBM"             # Organization to be granted default approval
Def_OU="ITSO Austin"    # Org. Unit to be granted default approval

# Read indefinitely from stdin
while read stdin; do
  # Break input into the basic variables
  line="$(echo "$stdin" | sed 's/&/"\\n/g'  | sed 's/=/="/g')"
  eval $(echo "$line")
  res=""
  if [ "$RaPolicyExitArg" = "Enroll" ]; then
    # Break request_vars into single variable assignment pairs
    line="$(echo "$request_vars" | sed 's/%26/"\\n/g' | sed 's/%3d/="/g')"
    line="$(echo "$line" | sed 's/%2b/ /g')"
    eval $(echo "$line")
    # Check if request comes from specific organization and unit
    if [ "$O" = "$Def_O" -a "$OU" = "$Def_OU" ]; then
      # Mark this request as "approved"
      res="reqStatus=Approved"
    fi
  fi
  if [ "$res" = "" ]; then
    # Not a known O and OU, so let 'auto_approve' handle this case
    res="$(echo "$stdin" | $(dirname $0)/auto_approve)"
  fi
  # Return result to stdout
  echo "$res"
# End of loop
done
```

This example is just a very simple program that demonstrates the general capabilities provided with the RA program exit. The RA program exit can be used for a number of purposes. For example, it can be used to check the user's credentials and/or eligibility for the requested certificate type in cooperation with other services and applications.

6.7.4 Other customization

Trust Authority can be customized to some extent on the component level by editing the respective configuration files. A specialized editor, the *IniEditor*, is provided that allows an administrator to easily edit configuration parameters in a configuration file, though any text editor can be used as well.

The reference section of the *Trust Authority System Administration Guide* contains a list and brief description of all the parameters in the configuration files and whether it is safe to change them. The same guide also explains how to start and use the IniEditor program.

The configuration files are:

CA server configuration file:	jonahca.ini
RA server configuration file:	jonahra.ini
Audit Server configuration file:	AuditServer.ini
Audit client configuration file:	AuditClient.ini

Although the *Trust Authority System Administration Guide* lists some parameters within the configuration files as “safe to change after configuration,” extreme care should be taken when these parameters are changed. Such changes include TCP/IP port numbers (as discussed in 6.7.1, “Defining alternative TCP/IP ports” on page 148), the frequency of CRL updates and CRL life time, certain URLs, polling intervals, the distinguished name of the CA, and a number of audit and trace options.

Before any modifications to the configuration files are made, make sure that you understand the purpose of the individual parameters. As always, it is strongly recommended to verify any modifications in a test environment before any changes are applied to a production environment.

6.7.5 Managing Trust Authority

Managing and running Trust Authority can involve a broad spectrum of activities, such as:

- Starting and stopping the Trust Authority services
- Changing passwords
- Managing each of the components, such as the operating system, the HTTP Server, the WebSphere Application Server, DB2, the LDAP server, and the Trust Authority components
- Working with the audit subsystem
- System backup and restore

- Monitoring the system(s)

In normal operation, Trust Authority does not require special attention and runs self-sufficiently after the installation, configuration, and customization work is done. In a production environment, however, it is advisable to have some procedures and skills ready to support any special tasks, such as problem determination when problems are suspected or apparent.

It is beyond the scope of this book to detail all the possible tasks involved in both routine and unplanned operation of Trust Authority. The Trust Authority manuals that are available online (see Appendix C, “Related publications” on page 221) describe such operations in greater detail. The following is a brief description of the tasks mentioned above provided for your convenience to give you some ideas of what is involved in the management tasks.

6.7.5.1 Starting and stopping Trust Authority

To start Trust Authority on Windows NT, select **Start** -> **Programs** -> **IBM SecureWay Trust Authority** -> **Start Trust Authority**. To stop TA, click **Start** -> **Programs** -> **IBM SecureWay Trust Authority** -> **Stop Trust Authority**. On AIX, two commands are available for that purpose: `Start_TA.sh` and `Stop_TA.sh`, respectively. In order to run these commands, you must be logged on as the user under which Trust Authority runs. By default, this is the user cfguser.

Both actions, start and stop, will prompt you for the Control Program password, which is used to enable the system to auto-start or shut down all TA components.

6.7.5.2 Changing passwords

To change the various passwords for Trust Authority’s subsystems and components, a tool is provided that offers a menu as shown in Figure 36 on page 159.

Note

You should not change any default passwords until after installation and configuration is complete. In addition, if the password of the system user account used during installation (cfguser) is changed on AIX, make sure that you follow the special instructions in the README document.

```
------------ Change Trust Authority Passwords -------------
Enter the option number for the component you want to change.
You will be prompted to enter the current password and then
the new password.
-----------------------------------------------------------
Change password for
1) Quit
2) Control Program
3) Directory Administrator
4) Audit Administrator
5) 4758 CA Profile
Enter Option :
```

Figure 36. TA utility for changing various passwords

The tool's name is changePWD.sh on AIX and changePWD.bat on Windows NT.

In the menu (Figure 36), the Control program password is the password required for starting and stopping Trust Authority (see previous section). The Directory Administrator password controls access to the elements that TA stores in the LDAP directory. The Audit Administrator password provides access to the audit logs and the audit administration tools. The 4758 CA Profile password controls access to the 4758 PCI Cryptographic Coprocessor CA profile (if a coprocessor is installed).

6.7.5.3 Managing platform services and components

Trust Authority is built on top of several of services and components, herein considered *platform*. As outlined in previous sections, the platform includes:

- The hardware and the operating system with a working networking, Java, and C++ runtime (AIX) environment
- IBM DB2
- An LDAP server
- The WebSphere Application Server
- The IBM HTTP Server
- An optional IBM 4758 PCI Cryptographic Coprocessor

Although each of these components can be fairly complex if they are used and managed separately, the Trust Authority installation and configuration process does all the necessary work such that they appear transparently to the systems operator and/or user. Special skills for these components are only required if special performance tuning or complex problem analysis becomes necessary. Except for the hardware and operating systems, the *Trust Authority System Administration Guide* contains some specific information about how you can check the status of, and review log information created by, the components listed above. This allows you to do some basic managing and problem determination of the components.

6.7.5.4 Working with the audit subsystem

The audit subsystem stores audit records of all relevant transactions within the Trust Authority system (see Figure 19 on page 113 and 6.2.3, “The Audit Server (AS)” on page 117). Audit records are stored in a DB2 database, and thus, retrieval and formatting of audit records can be done with standard DB2 tools, such as Structured Query Language (SQL) queries and reports.

The Trust Authority audit database uses a schema that is based on the recommendations that are described in the *Public Key Cryptography for the Financial Services Industry* standard, X9.57. The full audit database tables are described in the reference section of the *Trust Authority System Administration Guide*. For easier use, two views are provided with Trust Authority that allow an administrator to more easily access the audit data than when dealing with the raw audit tables. The view names are:

- viewar
- viewar_t

The difference between the viewar and viewar_t views is that the latter has all text columns truncated to 40 characters for easier handling if full text is not required.

The columns contained in these two views are listed below in Table 1.

Table 1. Audit table view fields

Column name	Description	Data type
serial_num	The serial number of the audit record	integer
sourcetime	The timestamp when the audit event was generated	timestamp
createtime	The timestamp when the audit record was created	timestamp
event	The name of the event	varchar
source	The audit client that generated the audit event	varchar
component	The component type of the audit client that generated the audit event	varchar
auth_entity	The entity that authorized the audit event	varchar
auth_role	The role of the entity that authorized the audit event	varchar

Column name	Description	Data type
affected_entity	The identity of the entity affected by audit event	varchar
affected_entity_type	The type of the affected entity	varchar
storage_media	The storage media associated with the audit event	varchar
extra_info	Additional information associated with the event	varchar

To take a look at the individual audit records, an administrator can execute any DB2 commands against the raw tables or the views as previously explained. Sufficient privileges for accessing the DB2 database must be attained before executing such commands. In a default installation, the user cfguser has these privileges. For example, the following commands display all records in the viewar_t view (output from the commands is not shown):

```
> db2
db2 => connect to adtdb
db2 => select * from viewar_t
```

In most cases, the result from these commands would be too much, and thus, the SQL statement can limit the records shown. For example, the following SQL statement would only show records that relate to the entity John Smith:

```
db2 => select * from viewar_t where affected_entity = 'John Smith'
```

The audit subsystem can be customized by editing its configuration files (see 6.7.4, “Other customization” on page 157).

6.7.5.5 System backup and restore

For any backup and restore procedures that need to be in place for Trust Authority, as well as any other computer system environment, you must consider that Trust Authority is heavily dependent on DB2. Any backup and restore procedure must be able to handle database backup and restore. Note that online backup of a relational database requires special considerations that can only be circumvented by stopping the application (and DB2) while performing backup operations.

Since LDAP also runs DB2 underneath, the same considerations apply to the LDAP server, be it in the same machine or in a remote system.

The *Trust Authority System Administration Guide* (available online on the Trust Authority library Web site) contains a section that outlines these considerations.

6.7.5.6 Monitoring the system(s)

In a production environment, adequate means should be in place to monitor critical system parameters and alert system administrators of any abnormal situation. Besides common systems monitoring, such monitoring in a Trust Authority environment should include:

- The Trust Authority application processes
- The WebSphere Application Server and the HTTP Server and their log files
- DB2 and its utilization of allocated resources
- The LDAP server process and its associated DB2 subsystem

6.8 Supported standards and technology

To wisely choose a PKI solution, it is important to check the standards and technology supported by, or implemented into, the product. If a product supports many open standards, this means that this product can interoperate with other standards-compliant products (from the same vendor or from other vendors). An organization will benefit in many ways: easy migration and scaling, less programming effort, and reduced programming time due to the well-defined architecture and APIs. This section briefly introduces the standards supported by Trust Authority. Some IBM technologies, such as IBM KeyWorks and Jonah, will also be introduced, and in fact, these are the actual implementation of the standards. By understanding the underlying standards and technology, users will have a better idea of how the TA components interact with each other and how the product can be customized and extended for the users' specific needs.

Please refer to Chapter 5., "PKI-related standards" on page 101 for more specific information about the standards. Please also refer to *Trust Authority Up and Running* for the list of standards supported by Trust Authority.

6.8.1 Standards

To allow vendor interoperability with other standards-compliant products, IBM SecureWay Trust Authority has implemented many open and industry-accepted standards in its components. For the server components, Certificate Authority, Registration Authority, and SecureWay Directory, standards for certificates, CRLs, key lengths, encryption or decryption, signing, hash algorithms, communications between components, and Directory Access Protocols are supported and utilized. It is important to know that certificates created by Trust Authority can be readily used by most common cryptographic purposes, such as SSL, IPSec, VPN, and S/MIME, as it supports most of the fields and extensions defined in the X.509v3 (PKIX) certificate standard.

Other than the server components, standards are also supported in the cryptographic components: the IBM 4758 PCI Cryptographic Coprocessor hardware and the IBM CCA Cryptographic Coprocessor Support Program. Standards of encryption, decryption, signing, and hash algorithms are implemented to support the cryptography and to generate key pairs securely. As pointed out in Section 4.4.1, “Server machines and cryptographic hardware” on page 86, the 4758 cryptographic card has met FIPS 140 level 4 requirements for resistance to physical attacks.

Table 2 lists the standards supported by Trust Authority for public key cryptography.

Table 2. Standards supported by IBM SecureWay Trust Authority

Component	**Standard**
Certificate Authority	• X.509v3 certificates • CRLv2 • Key lengths up to 1024 bits for encryption and key exchange keys • Key lengths up to 2048 bits for CA signing keys • RSA algorithms for encryption and signing • MD5 and SHA-1 hash algorithms • PKIX CMP via TCP/IP for communications with RA
Registration Authority	• SSLv2 and v3, with client authentication • PKCS #10 browser and server certificate format, with a Base64-encoded PKCS #7 response • PKIX CMP certificate format, with a PKIX CMP response • IPSec certificate format • S/MIME certificate format • Browser certificates for Microsoft Internet Explorer v4.x and v5.x, Netscape Navigator and Netscape Communicator v3.x and v4.x • Server certificates for Netscape Enterprise Server, Microsoft Internet Information Server • Smart card certificates (PKCS #11 interface) for Trust Authority Client application, Netscape Navigator and Netscape Communicator v3.x and v4.x • LDAP standard for communications with the Directory • PKIX CMP via TCP/IP for communications with CA • PKIX CMP via TCP/IP for communications with Client application
SecureWay Directory	LDAP Version 3.0, with RFC 1779 syntax

Component	Standard
IBM 4758 PCI Cryptographic Coprocessor	• FIPS 140 level 4 requirements for resistance to physical attacks • Support for industry-accepted cryptography standards: •DES for encryption/decryption • RSA for signing/signature verification • PKCS #1 block type 00 • PKCS #1 block type 01 • PKCS #1 block type 02 • MD5 and SHA-1 hash algorithms • X9.9 and X9.23 ANSI • ISO 9796
IBM CCA Cryptographic Coprocessor Support Program	Provides services for the 4758 coprocessor, including the secure generation of RSA key pairs with modulus lengths as long as 2048 bits, and: • SET • DES for encryption and decryption • RSA for signing and signature verification • MD5 and SHA-1 hash algorithms

As pointed out in Section 5.2, “Internet RFCs” on page 102, the PKIX group has published many Internet-based PKI standards, for example, the X.509 Version 3 certificate format and the certificate management protocol standards. To understand more about certificates and the related administration, it is important to refer to the standards.

Standards are not confined to certificates. The Common Data Security Architecture (CDSA), introduced in Section 5.5, “Other standardization efforts” on page 107, was designed by Intel Corp. and adopted by The Open Group as an industry-accepted specification to make computer platforms more secure for all applications. CDSA solutions are open, interoperable, cross-platform supported, and extensible. A single and functionally rich API, called the Common Security Services Manager (CSSM) API, and a set of services are defined in CDSA. This API is important because it invokes the services of the service providers in a way that is independent of the API of the service provider itself. Some examples of service providers are encryption, certificate management, key recovery, trust policy, and data storage. Therefore, it makes the specific details of a particular service provider transparent to the application. A set of services is provided for each of the multiple service providers to plug into it. This maintains a link between the applications and the plug-in modules so that the service providers can communicate with the CDSA interface.

So, what is CDSA’s relation with Trust Authority? CDSA has been implemented in the IBM KeyWorks Toolkit, which is considered one of the most mature and sophisticated implementations of CDSA. The IBM KeyWorks Toolkit, in turn, is deployed in Trust Authority as an underlying technology. A simplified KeyWorks-based application architecture is shown in Figure 37. The next section details KeyWorks a bit more.

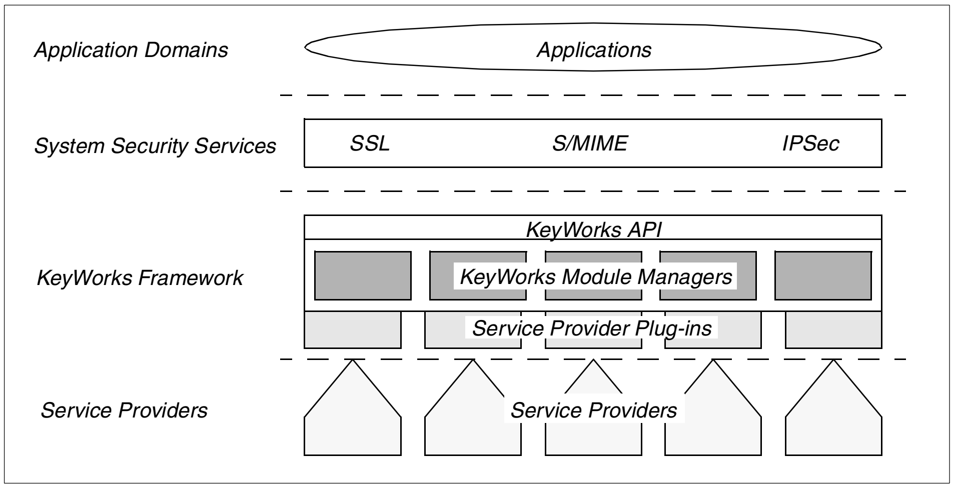


Figure 37. Simplified KeyWorks-based application architecture

6.8.2 Technology

When Trust Authority is installed, IBM KeyWorks Toolkit is installed automatically. KeyWorks is the first product to implement CDSA Version 2.0. With IBM KeyWorks, applications can encrypt and decrypt information, verify a digital signature, retrieve a certificate from a directory, and decide if a certificate can be trusted. It also provides support for hardware-based encryption or other hardware tokens that support the PKCS #11 interface. These cryptographic operations can be performed by using a standards-based API.

Another underlying technology used by Trust Authority is called *Jonah*. Jonah is a PKIX reference implementation supplied by IBM based on the IETF PKIX specifications and the CDSA software framework. Jonah has implemented the CDSA specifications that are needed to form a basic CA. Therefore, Jonah is a CDSA architecture that primarily supports PKIX applications, such as Trust Authority.

Chapter 7. Deployment scenarios for Trust Authority

After having explored the theoretical framework of a PKI and how the IBM Trust Authority product can serve as the heart of such a PKI, this chapter takes a more practical approach. A sample, but realistic, scenario is introduced and then practical hints and tips are given about how IBM Trust Authority can be exploited in this sample scenario.

The scenario is representative of initiatives for deploying and integrating PKI concepts and technologies in many industries. While fictitious, it parallels work that is underway in the banking industry. Some of the details are incomplete, since the specifications are typically restricted to participating financial institutions.

7.1 Situation

In our sample scenario, executives of a bank have agreed in principle to participate in an industry initiative that is targeted at developing a suite of electronic services for corporate customers.

The Chief Information Technology Officer for our fictitious bank has been asked to study the specifications, determine the direct and indirect impact on bank operations and bank information processing, make recommendations as to content and delivery timing of the new services portfolio, plus build the production-level technology infrastructure and complete a proof-of-concept for one or two applications.

7.2 Background information

Several international banks have formed a global organization with the objective to facilitate business-to-business banking and trust transactions. The representing organization, which we will call *ITRUST*, has studied the issues and tested technology components over the past year, has developed a set of specifications, including formats, protocols, and components, and has documented a set of deployment standards and compliance criteria that participants must follow.

From a technology perspective, the specifications describe an architecture for information processing that implements and supports electronic credentials known as *digital certificates*. These digital certificates are part of an electronic trust model known as Public Key Infrastructure, or PKI.

In the ITRUST PKI, digital certificates are issued by participating banks to representatives, or more accurately, computing systems that represent corporate customers. The electronic credential is used within banking and trust transactions as the basis for exchanging information, authorizing transfers, or executing contracts on behalf of the corporate customer. In addition to the format of these electronic credentials, the specification lists technology components necessary to integrate electronic identity checking into business applications, as well as components that provide up-to-the-moment validity checking of credentials.

The integration and use of ITRUST credentials is more complex than standard uses for PKI. Banking services and trust transactions between corporate accounts can be either intra-bank or inter-bank. Consequently, electronic credentials issued by a banking institution to one of its corporate accounts may need to be validated by a second bank during a multi-party transaction. The use of a single credential for transactions at multiple banks results in a trust model with two, three, or four participants. This type of trust model has implications in terms of software interoperability, network infrastructure, business agreements, and service levels among participating banks and their customers.

7.3 Technical details of ITRUST

The ITRUST controlling organization has defined an electronic trust model that maps the peer relationship among banks and the contractual relationship between each bank and its corporate customer(s) to a hierarchy. The ITRUST controlling organization represents the root authority. A bank that participates in the ITRUST initiative is delegated the authority to issue digital certificates to its corporate customers. In PKI terminology, a participating bank represents a level 1 Certificate Authority (CA) within a hierarchy. ITRUST level 1 Certificate Authorities must comply with the ITRUST business agreements and must adhere to the technical and operational specifications of the ITRUST architecture.

The technical and operational specifications of the fictitious ITRUST architecture include:

- Formats, protocols, and components that make it possible to apply for, accept, and use the keys and digital certificates of a level 1 CA, per to ITRUST specifications
- Compliance with ITRUST requirements for protection of CA signing keys
- Formats that make it possible for a level 1 CA to issue and manage identity certificates to corporate customers, per the ITRUST specifications

- Formats, protocols, and components that make it possible to serve PKI credential validation requests from other banks, per the ITRUST specifications and service level agreements
- Correct and reliable integration of ITRUST concepts into banking and trust transactions, per the ITRUST specifications

7.4 Solution design issues

As part of a review process, the input of several in-house experts and managers from business operations, bank auditing, I/T operations, I/T security, I/T systems development, and maintenance departments, plus representatives from corporate legal, needs to be solicited in order to determine the value and the impact of this PKI on the existing enterprise I/T architecture and operations. The following sections describe possible perspectives.

7.4.1 Business operations view

From a business operations perspective, a participating bank has the opportunity to realize risk reduction, expense reduction, and service improvements with the infrastructure outlined in the ITRUST specifications. Additionally, this infrastructure will position a bank's information technology infrastructure for future value-added services. Here are some examples of the near term value of PKI to a bank's services and I/T operations.

Improve controls and accountability over who has access to sensitive information, who can execute transactions, and how sensitive information is exchanged:

- Reduce losses, and the risk of loss, due to fraud
- Reduce the workflow and complexity of administering computer systems and computer programs based on addition and removal of users
- Reduce the information technology costs associated with maintaining private networks

With these potential benefits, there are several concerns that must be addressed in the ITRUST deployment within a bank:

- It is believed that the new technologies and related business processes will result in changes to the “look-and-feel” of electronic banking services as well as customer support processes. These issues should be examined closely during any proof-of-concept projects.

- It is also recommended that all ITRUST-related services that the bank develops in the future should be designed with consistency and that the “trust factor” be communicated to clients and visible to users.
- The ITRUST specifications will require that the corporate customers have compatible software and systems to generate encryption key pairs, package certificate requests, receive and store keys and certificates, and perform the client-side functions of electronic banking and trust transactions.
- Some transactions that are technically feasible under the ITRUST architecture may not be recognized as legally viable in some countries, economic unions, or municipalities.

Two examples of applications that could be deployed within the ITRUST framework would be as follows: (1) an *Account Review* (bill presentment) Application that uses ITRUST credentials in place of user names and passwords, and (2) a *Payment Authorization Application* that uses digitally signed transmittals. These applications are discussed in more detail in 7.6, “Sample applications” on page 177.

7.4.2 Bank auditor view

Bank I/T and business processes will be given additional scrutiny by regulating and underwriting bodies based on issuance and use of electronic credentials. As with other aspects of bank audit, the distribution and management of electronic credentials and the use of electronic credentials within transaction processing and service delivery will be examined for correct and reliable operation based on prevailing guidelines.

In the United States, for example, there are a number of rules and opinions that provide initial direction to electronic banking operations, as well as criteria that represents “due diligence” in the design, operation, and assurance of Public Key Infrastructure systems and services.

- The US Treasury, Office of the Comptroller of the Currency, offers guidance to bank CEOs and CIOs of all national banks and also bank examiners on the subject of PKI Certification Authority Systems, which “... describes the roles of banks in emerging systems, and identifies the risks of such systems using the OCC supervision-by-risk framework.”
- The US Treasury, Office of the Comptroller of the Currency, has published a Comptroller's Handbook for Internet banking to be used by bankers and bank examiners “... on identifying and controlling the risks associated with Internet banking activities.”

- The Federal Deposit Insurance Corporation (FDIC) has provided rules for operation of PKI and Certificate Authority Systems within member banks.

7.4.3 I/T operations view

Implementing ITRUST and realizing the business operations improvements has implications to I/T operations. Four significant impacts to bank I/T systems and operations warrant further study:

- New technologies and processes will be required to generate, distribute, manage, and validate the electronic credentials appropriate to customer segments. It might not be clear what additional skills, computing facilities, or audit controls will be required.
- Access control and authorization processing at system, application, transaction, and possibly data object levels will need to be re-engineered. This re-engineering will force consistent implementations across every platform used to deliver the affected bank services and will result in a transfer of responsibilities from system-level administrators to others.
- Network connections will need to be reconfigured to provide connectivity to other enterprises and corporate customers.
- Changes in configuration and operation of networks, systems, applications and support processes will require a security review.

7.4.4 I/T systems development and maintenance view

From an end-to-end perspective, the ITRUST architecture accounts for the possibility of several types of transactions and data flows:

- A direct transaction between a bank and an account-holding business
- An indirect transaction between two businesses using the same bank
- An indirect transaction between businesses using different banks

Some of the systems and services in the end-to-end flow must be addressable on an extranet, and most likely on the Internet.

There are two design alternatives for applications: *passive integration* and *active integration*. Passive integration refers to adapting existing transactions in a way that is, hopefully, transparent to the current components and data flow. Active integration refers to engineering, or re-engineering, an application to use new protocols, services, and components. There are at least two levels of active integration: *PKI enabling* and *transaction enabling*.

PKI-enabling components provide options for performing access control, authorization checking, message integrity, and confidentiality.

Transaction-enabling components provide for non-repudiation and other value-add services.

Performance of I/T applications will be affected when they are required to implement distributed system calls in order to perform authorization checks and make access control decisions. These multi-system calls may include system and network delays, since the electronic credential of the transaction initiator may be owned by another bank, and the validation check may traverse open, connectionless transport networks.

Compliance requirements from ITRUST, National banks, and internal auditors put strict development and maintenance constraints on the system development life cycle for electronic banking services and support systems that are either physically, or network, accessible outside of the bank's facilities.

7.4.5 Corporate legal view

Based upon review of the legislation (existing and proposed), regulation, case law, American Bar Association papers, and other technical opinions, the bank's legal department takes the following position with respect to the proposed electronic services:

- The legal standing of electronic transactions that substitute an electronic signature for a physical signature is somewhat behind the pace of technology advances in the area.
- For the protection of the bank, new contract amendments should be drafted and executed with corporate customers to cover electronic credentials, digital signatures, and the intent and limitations of the proposed electronic services.
- In preparation for the potential for future liability litigation related to electronic services, digital signature, or breach of privacy, the solution development, deployment, and operations teams should maintain records and documentation supporting the correct design, and operation of bank electronic services should be compiled and archived.
- Because of the changing legal environment, periodic reviews of the services and changes to legislation, case law, and regulations related to banking and information privacy should be conducted.

7.4.6 I/T security view

Just as structured programming techniques and multi-tier client/server architectures seek to isolate application functions into logical layers that can be distributed and shared, PKI has the effect of distributing identity services, upon which authentication and access control decisions are made within equipment, operating systems, and applications.

Integrating PKI into a bank's I/T environment and deploying new Internet-based banking services have several security implications:

- Robust design and implementation of security measures in the major component architectures within the enterprise I/T architecture, including: data architectures, application architectures, platform architectures, and network architectures
- Diligent monitoring and analysis of the entire I/T operation for any fault or activity that would compromise the trustworthiness of the facilities, the network, the systems, the data, the PKI, or the ability of the organization to pass compliance tests from either ITRUST, government regulators, or internal examiners
- Enforceable policies and business processes that ensure compliance with the issuance, use, and revocation of electronic credentials

7.5 Solution design approach

PKI solutions take many forms, including technology demonstrations, proof-of-concept, pilot, and scalable production. The requirements for each of these forms include PKI technical components; however, the security aspects for pilot and production deployments must address the business risks.

While there are many differences from PKI to PKI, a consistent and comprehensive approach to design can be achieved. Each design can be described in terms of five elements, which will interleave the technology, business process, and security considerations into the solution. These five elements are:

- Electronic trust model
- Fulfillment framework
- Deployment framework
- Assurance framework
- Integration framework

7.5.1 Electronic trust model

Electronic trust models map the traditional roles and responsibilities to the concepts defined by PKI. The overall structure of a PKI is represented as *non-hierarchical*, *hierarchical*, or *cross-certified*. The format and contents of the digital certificates, along with statements of acceptable policies and practices, are part of the electronic trust model.

For ITRUST, the electronic trust model is defined by the controlling organization, with the banks being members of a hierarchy. This scheme allows for credentials issued by one ITRUST bank to be recognized by another ITRUST bank. The ITRUST PKI deployment scenario in this document does not describe how the ITRUST root Certificate Authority (CA) was created. It is only concerned with the deployment of a level 1 Certificate Authority at a participating bank.

In general, the types of consulting, design, and integration expertise required to properly define and document an electronic trust model for a level 1 CA includes the following: I/T security, PKI theory and practice, legal contracts and liability, and business and technology policy creation.

7.5.2 Fulfillment framework

Building an electronic trust infrastructure requires design of end-to-end processes that result in the distribution of PKI credentials to the subscribing community or communities. The processes include automated processes and manual processes, as well as the interface between human and machine. While these processes can be derived from their traditional equivalent, there are some specific differences:

- The processes do not need to be carried out in the physical presence of an authorizing person, such as a bank manager.
- The processes can be invoked at any time, at any hour of the day, and from any location.
- The processes may be fully automated, or may require the intervention of an authorizing person who approves remotely via an administration program.
- The workload for enrollment and approval is subject to many variables, including size of population served, enrollment scheduling or spot demand, mass production, and method of storing PKI credentials.
- There may be several variations of enrollment and approval to suit the “trust” requirements of a bank, a corporate customer, or a subscribing community.

A critical objective of the ITRUST specification is to ensure that the enrollment and approval frameworks will all result in compatible PKI credentials; however, each participating institution will need to evaluate and select options for enrollment and approval processes to fit the communities that it serves.

In general, the types of consulting, design, and integration expertise required to properly define and deploy a PKI fulfillment framework include business process, I/T security skills, PKI technology skills, and application design.

7.5.3 Deployment framework

A deployment framework represents the processes, components, and mechanisms used by parties to a business transaction in order to present credentials, to have a credential validated to the satisfaction of both parties, and to be in compliance with business policies.

For a Public Key Infrastructure, the mechanisms include key generation, encryption, decryption, signing, validation, symmetric key exchange, secure transport, and secure payload. There are a variety of protocols and services that can be used by components that implement these mechanisms, depending on the application flow and the technologies available.

As with fulfillment frameworks, deployment frameworks have manual processes and automated processes. The manual components can be divided into the mechanical processes for human-machine interface and other procedures associated with compliance. The automated processes consist of information exchange and the algorithms that are executed within the components and mechanisms.

In order to perform PKI-based authentication, some or all of the clients, servers, and applications within the network must have PKI components that support a compatible set of mechanisms and services. The components must be integrated and tested for interoperability.

PKI-based authentication is commonly found in four applications:

- Secured communication between Web browsers and Web servers, also known as Secure Sockets Layer, or SSL
- Signed or confidential exchange of electronic mail and documents, also known as Secure/Multipurpose Internet Mail Extension, or S/MIME
- Authenticated or confidential communication for some or all of the Internet Protocol (IP) exchanges between nodes, also called IP Security or IPSec
- Generalized encryption and signing using asymmetric key algorithms, such as RSA and others

For PKI, trusted computing environments are built on the trust that the PKI credentials are valid. The validity of a PKI credential is based partly on the integrity of the static content in the certificate, and partly on the continued compliance of the parties identified in a given certificate with the policies and agreements that are designed to ensure trustworthiness.

The ITRUST architecture includes specifications for PKI-enabled services and mechanisms, along with sample implementations for passive and active integration with business computing systems.

In general, the types of consulting, design, and integration expertise required to properly define and deploy a PKI deployment framework include I/T security skills, PKI technology skills, and system integration.

7.5.4 Assurance framework

An assurance framework represents the systems management, security, and risk management processes, components and mechanisms that ensure the continued viability of the computing environment. The assurance framework has elements directly related to the PKI design plus elements related to the broader discipline of Information Security Management, which are targeted at eliminating or minimizing vulnerability from threats in the following categories: unauthorized access, denial-of-service, repudiation, accidents, poor design, malice, and fraud.

The first element of the assurance framework relates to PKI design. In general, a PKI can be considered trusted if it has been designed properly, if it has been deployed properly, and if the compliance agreements are enforceable. This requires due diligence in all phases of planning, design, implementation, and operation. In order for a trusted PKI to be deployed properly, it must be monitored and the participants must comply with procedures and agreements.

The second element of the assurance framework relates to the broader discipline of Information Security Management, which is intended to actively ensure:

- The integrity of the Certificate Authority systems
- The integrity of the directory service
- The currentness of the certificate revocation lists
- The integrity of private keys in the business computing systems
- The integrity of private keys belonging to the subscribers
- The currency of the information used by the credential validator systems
- The availability of all critical components to all critical relying parties

The validation services defined by ITRUST are considered part of the assurance framework in that accuracy and timeliness of dynamic validation services is the critical component in minimizing the window of vulnerability for the business application and minimizing the risk of providing warranty services.

In general, the types of consulting, design and integration expertise required to properly define and deploy a PKI assurance framework include business process, business and technology risk assessment, I/T security skills, PKI technology skills, and application design.

7.5.5 Integration framework

How the solution is integrated depends on the components and technologies within the existing enterprise I/T environment, the components prescribed in the PKI deployment framework, and the requirements placed on the solution by I/T operations, I/T security, and business operations.

The types of expertise required to properly define and deploy a PKI deployment framework include PKI technology skills and system integration, systems management , and client/server application development.

7.6 Sample applications

In the scenario described earlier, bank business operations has recommended that two applications be developed using the ITRUST PKI. The first is a conversion of an existing Account Review Application that provides remote viewing of account information, including transactions, balances, and so on. The second is a new electronic payment authorization transaction that would replace an existing paper process.

The business objectives for the deployment of these applications are:

- To introduce corporate customers to the use of digital certificates
- To improve electronic services
- To develop an electronic payment authorization transaction that will satisfy legal scrutiny

The solution needs to account for the way in which PKI is integrated into the application via protocols, services, and mechanisms, as well as how events that occur electronically, and how the electronic identities upon which the events are based, can be maintained over time.

7.6.1 Design method

The architecture task consists of converting the business-level requirements into a set of coordinated processes and technologies that achieve the business objectives within the risk management parameters. Solution design for PKI typically requires the combined efforts of several I/T disciplines, including client/server, networking, database, security, and systems integration. Each business process in the solution may have unique considerations with respect to electronic trust model and fulfillment, deployment, and integration issues, but all solutions will utilize a common infrastructure and be managed by common assurance processes.

The business-level security requirements will be described in terms of three infrastructure requirements: authorization, accountability, and asset protection, plus three operational requirements: administration, availability, and assurance. The technology-level security requirements will be described in terms of the threats to be addressed, such as unauthorized access or denial-of-service. Only the threat responses directly addressed by cryptography will be considered within this analysis.

The business-level functional requirements will be described in terms of end-to-end processes. The technology-level functional requirements will be mapped to one or more of the four standard alternatives for deploying a Public Key Infrastructure within I/T applications. The application templates are:

- HTTP with Secure Sockets Layer (SSL) or HTTPS
- Secure Multipurpose Internet Mail Extension (S/MIME)
- IP Layer Security (IPSec)
- Custom flows using platform services and application-level programming interfaces with custom data structures

Pre-architected solutions, such as Lotus Notes, represent a fifth option; they will not be considered within this analysis.

Selection of application technology will be based on the following criteria:

- Suitability to the business process
- Viability in supporting the risk management requirements
- Acceptability to the target user communities

7.6.2 The Account Review Application

The bank offers a business-to-business credit card that sends semi-monthly statements by standard postal service to the accounting departments of cardholding companies. There are three shortcomings with the current design:

- Only one copy of the billing statement is sent to the customer, or directly to the customer's accountant. This results in manual copy-and-distribute workflow at the customer.

- Postal delivery averages three days, which can represent up to 30 percent of the (10 working day) billing cycle.
- Customers want account status with daily transaction information.

Business operations sees an opportunity to make the static billing statement information available on demand electronically and offer a potential value-added service that provides daily transaction posting information, as well as customizable transaction summaries.

7.6.2.1 Business process flow

The business process flow for this application is very straightforward as shown in Figure 38. The bank (through an authorized clerk, customer support representative, or automated process) provides information on demand to corporate customers (through an authorized individual or automated process) as an alternative to statements delivered via postal service.

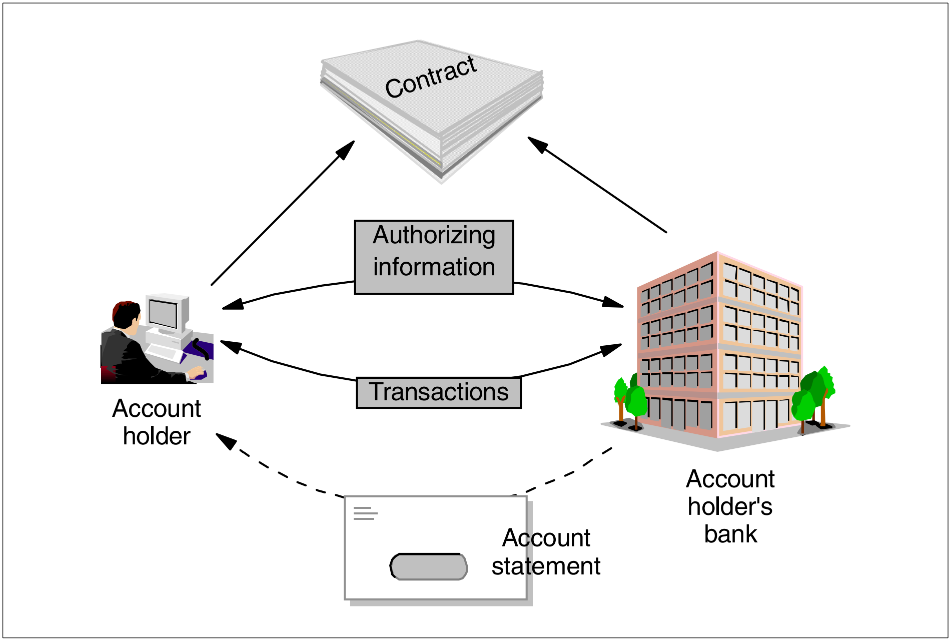


Figure 38. Business process flow

7.6.2.2 Business-level security requirements

As with any electronic transaction, the extent to which security services are implemented in account review transactions depends on the requirements for business controls and risk management. Table 3 summarizes typical business-level security requirements for the Account Review Application.

Table 3. Business-level security requirements

Category	Requirements
Authorization (verifying identities and controlling access)	•Only authorized individuals should have access to the account review transactions and the query results. •Authorization is determined by demonstrating ownership of the private key associated with a valid digital certificate. •Each individual identified by the digital certificate will be designated by the authorized representatives of the account holder through some secure means. •Each designated individual will complete the process to enroll for a digital certificate to be used in conjunction with the Account Review Application.
Accountability (tracking actions and events unique to individuals or entities)	•Given an action or event, the bank can determine who did it and what was done. •The bank is responsible for the correct and reliable operation of its computing systems. •The account holder is responsible for the correct and reliable operation of its computing systems.
Asset protection (controlling data confidentiality, privacy, and integrity)	•The confidentiality of any information assets associated with the Account Review Application that are in the control of bank will be secured when stored and will be encrypted when transmitted on open networks. •The integrity of any information assets associated with the Account Review Application will be maintained when transmitted and when stored on bank computing systems. •The true and permanent copy of the account review information will be maintained by the bank. •All copies retained outside of the bank's computing systems, whether stored electronically or in hardcopy, are the responsibility of the account holder. •Both the bank and the account holder are responsible for the protection of encryption keys, passwords, software, and systems associated with access to and use of the Account Review Application. •The bank will not maintain copies of encryption and signing keys used by the Account Review Application, since there is no long-term encryption or signing requirement.
Administration (defining and managing rules)	•The bank owns the responsibility to issue, manage, and revoke digital certificates associated with the Account Review Application. •The account holder owns the responsibility to designate individuals for enrollment or un-enrollment.

Category	Requirements
Assurance (validating that controls are working as designed, efficiently and effectively)	•The bank will conduct periodic tests and reviews of all systems and processes. •The account holder will conduct periodic tests and reviews of all systems and processes. •The bank will periodically request confirmation of compliance from account holders.
Availability (providing information and services where and when needed)	•The Account Review Application will be available on a best effort basis, with telephone support as a backup. •Account holders will be able to revoke credentials at any time the Account Review Application is available.

7.6.2.3 Technical security requirements

Since the trust model design is inherited from ITRUST, the remaining technical security design activity in this scenario is limited to the threats to the business process from unauthorized access and repudiation. In real life, a separate risk assessment and vulnerability analysis should be performed to identify additional technical and assurance requirements for the solution.

7.6.2.4 Functional architecture

Based on these observations, the standard application templates for PKI have been evaluated in Table 4. The ratings only apply to this scenario, and are not intended as a general statement for all such applications.

Table 4. Functional architecture

Selection Criteria	HTTPS client auth	S/MIME	IPSec	Custom
Business process suitability	possibly	no	unlikely	possibly
Viability for security	possibly	n/a	no	possibly
User acceptance	possibly	n/a	n/a	unlikely

Account Review Applications are often implemented with a “three tier” client/server model using a query/response protocol flow. These flows represent real time transactions, requiring application level authentication. S/MIME is a store and forward protocol that does not lend itself well to this application. IPSec provides system-to-system-level authentication, but no application-level authentication.

For the purposes of illustration in this scenario, SSL client authentication in conjunction with the HTTP application-level protocol is considered preferable to custom application flows because user acceptance is more likely when custom client software is not required.

Note

In real life, this selection is contingent on whether the certificate format and content prescribed by the ITRUST specifications are compatible with the available SSL client and server implementations.

The solution concept is depicted in Figure 39.

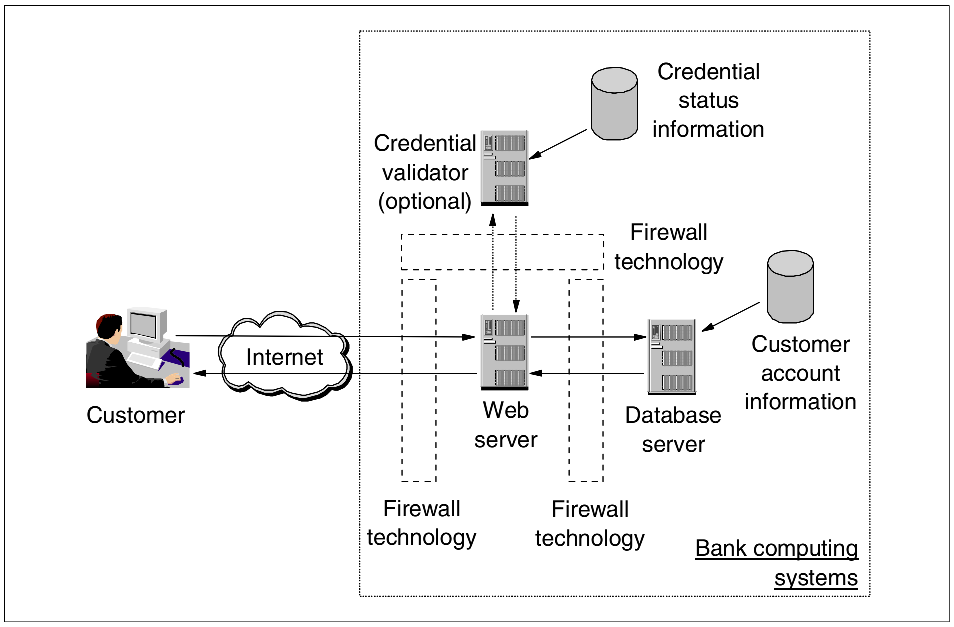


Figure 39. Technical solution concept

For the purposes of this solution, the process is initiated when a customer uses a Web browser to access the appropriate page of the bank’s Web site. A server side application authenticates the customer by validating the customer's credentials. An access control decision is made based on the results of the credential check, and a pseudo-session is established that allows the user to query and retrieve information corresponding to his/her access control permissions. The Web server application process communicates with the bank's internal database services to retrieve and format the appropriate account information. The actual mechanisms for creating the pseudo-session, relaying credentials to the data server, and maintaining status during the idle period between user queries is beyond the scope of this discussion.

The SSL protocol provides the mechanisms, protocols, and services for exchange and basic validation of digital certificates between Web browsers and Web servers. Hypertext Transfer Protocol (HTTP) provides the mechanisms, protocols, and services for sending, receiving, displaying, and processing text, graphics, and multimedia between Web browsers and Web servers. HTTP transactions consist of client-side browser requests, transmitted to Web server-based gateway applications, and possibly relayed to a backend application. A data formatting operation may be performed within the gateway application.

Remember from previous chapters that SSL can be used with Web-based applications for client authentication. The client (user) is then required to present a digital certificate (stored on his or her workstation or on a Smartcard) when an SSL session is being established. The server-side application can then retrieve user credentials from that certificate to perform further authentication and validation. Validation of certificates within SSL, however, is limited to checking of static information within the certificate. Bank business processes likely require an up-to-the-moment credential status. This can be provided through SSL extensions in the server process to access remote directory or certificate status servers.

In this flow, the customer must have an Internet-enabled workstation supporting both the TCP/IP communication protocols found in every current operating system and HTTP/SSL application protocols found in Web browsers from Netscape, Microsoft, and others.

7.6.3 The Payment Authorization Application

There are many examples of supplier/buyer relationships between corporate clients of a bank. Some of the procurements associated with these relationships can be one time, and some are recurring. Examples of recurring procurements result from services contracts, or installment purchases.

Recurring payments can be handled via invoice/payment, as shown in Figure 40 on page 184, or through electronic transfers. Recurring electronic transfer may be established via a paper process where the authorizing agent of the consumer provides originating account information (including bank, account number, and amount) to the authorizing agent of the supplier. The information for the receiving account is appended, and the payment authorization is presented to the supplier's bank for processing. After validation, a recurring electronic payment is initiated.

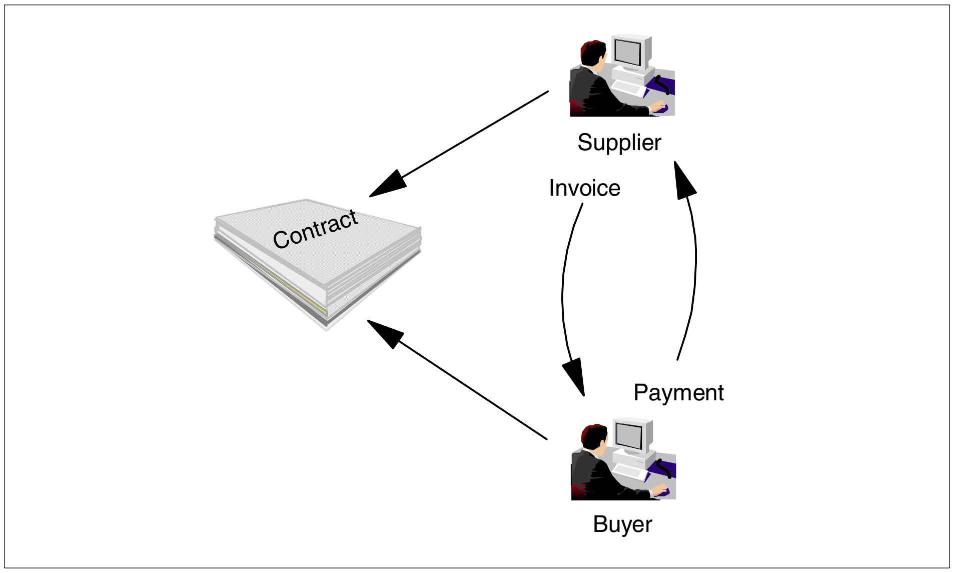


Figure 40. Invoice/payment process

There are shortcomings of the current paper process (Figure 40) for establishing recurring payment, including:

- The workflow for initiating payment is time consuming, taking several days to process.
- The manual processes and paper documents contribute to bank record keeping expense.
- The workflow for terminating payment is time consuming, taking several days to process; this can result in extra expense to the bank and its customers.

7.6.3.1 Business process flow

An electronic Payment Authorization Application needs to accomplish all of the process flows shown in Figure 41 on page 185. This application consists of a sequence of data entry and data update transactions where the data objects are relayed and appended with the digital signature of each party to the workflow.

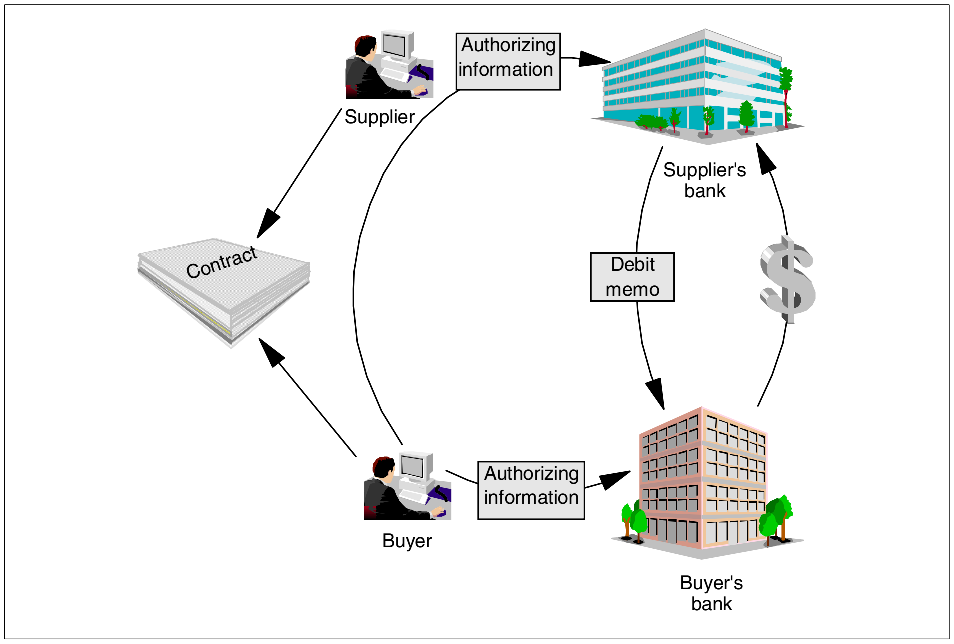


Figure 41. Business process flow

For example, if the buyer wants to register a periodic electronic payment to the supplier, the data objects might include the electronic signature of the buyer, the electronic signature of the supplier, their respective accounts numbers, and the details of the transfer. The information will need to flow to the supplier's bank and possibly the buyer's bank in order for the recurring payment to be enabled.

At their discretion, the supplier's bank, and possibly the buyer's bank, would validate the credentials and the signatures, or perform secondary transactions to validate the credential in relation to account information and the acceptability of the credential holders as parties to the transaction.

7.6.3.2 Security requirements

As with the previous Account Review Application or any other electronic transaction, the extent to which security services are implemented in a Payment Authorization Application depends on the requirements for business controls and risk management. Table 5 on page 186 summarizes business-level security requirements for the Payment Authorization

Application that are in addition to the requirements specified previously in the Account Review Application.

Table 5. Security requirements

Category	**Requirements**
Authorization	•Only authorized individuals can initiate electronic payment authorizations. •At the account holder's option, authorization of an individual could be restricted by dollar amounts, for example, <$10K, <$100K, or <$1M.
Accountability	•The bank is responsible for the correct and reliable operation of its computing systems. •The account holder is responsible for the correct and reliable operation of its computing systems.
Asset protection	•The confidentiality of any information assets associated with the Payment Authorization Application that is in the control of bank will be secured when stored and will be encrypted when transmitted on open networks. •The bank will maintain copies of encryption and signing keys used by the Payment Authorization Application to provide long-term encryption or signing of the true and permanent document.
Administration	•The bank owns the responsibility to issue, manage, and revoke digital certificates associated with the Payment Authorization Application. •All electronic credentials and encryption/signing keys used in conjunction with the Payment Authorization Application will be hand-delivered from the bank to the account holder. •The account holder owns the responsibility to designate individuals for enrollment, or un-enrollment. •At the bank's option, the account holder may be given direct control over the approval or revocation of digital credentials.
Assurance	•The bank will demonstrate on demand the receipt, validation, and confirmation of electronic payment authorizations.
Availability	•The Payment Authorization Application will be available on a best effort basis, with courier or postal service mail as a backup. •Account holders will be able to revoke credentials at any time during normal business hours.

7.6.3.3 Functional architecture alternatives

Based on these parameters, the standard application templates for PKI have been evaluated in Table 6. As before, the ratings only apply to this scenario and are not intended as a general statement for all such applications.

Table 6. Alternatives

Selection Criteria	HTTPS client auth	S/MIME	IPSec	Custom
Business process suitability	possibly	possibly	no	possibly
Viability for security	possibly	possibly	n/a	possibly
User acceptance	possibly	possibly	n/a	unlikely

Payment Authorization Applications represent a multi-party workflow where application-level authentication is tied to the originator of the data object rather than to the last system or user from which the data object was relayed. This type of authentication can be achieved with digitally-signed data. SSL, S/MIME, and custom applications can provide the mechanisms for digitally-signed data, with some variations in the implementation. IPSec does not provide application-level authentication.

Electronic workflow can be achieved through (a) client/server technology that is distributed among the participants, or through (b) centralized software acting as a clearinghouse or workflow coordinator. Within the ITRUST scenario, a payment clearinghouse for corporate banking customers would need to be operated by one or more participating banks, or an independent agency that is sanctioned by ITRUST. For the purposes of this ITRUST scenario, the distributed processing environment will illustrate more of the issues and options of PKI deployment.

In addition to the basic requirement for digital signature, there are other technical requirements in the solution, including the following: (1) the requirement for all parties to be able to understand and process the digitally-signed data object, and (2) the requirement for all parties to support protocols and services required to exchange the data object. Other requirements for the solution may include credential validation for multiple parties, authorization check of credential versus restricted policy, plus, secondary transactions for data gathering, synchronization, and record keeping requirements.

Many variations of the workflow are possible. The form may reside at the supplier's bank. The buyer may submit the completed form to the supplier's bank, and the supplier's bank may require the supplier to acknowledge the submittal by electronic signature. These variations may affect the application template(s) chosen, the server requirements at each location, the client software used by each party, and other aspects of the end-to-end process.

A pictorial view of one solution alternative for the Payment Authorization Application is provided in Figure 42.

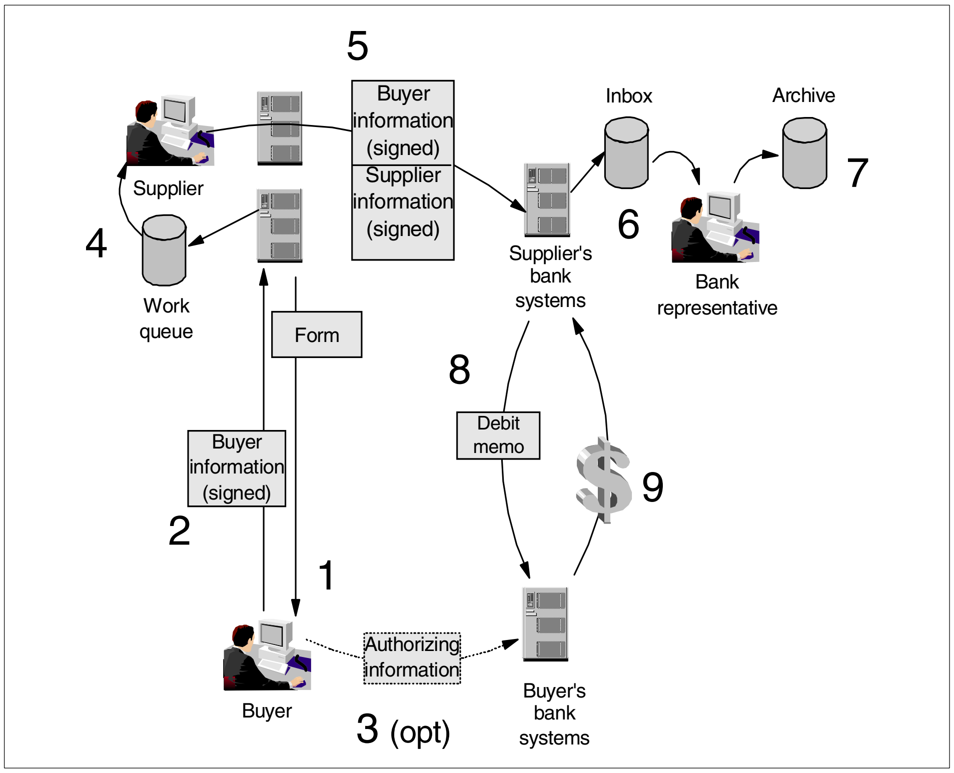


Figure 42. Solution alternative

In Figure 42, the flow is as follows:

1. The buyer accesses and completes a standardized electronic form that will be interpreted by all parties.
2. The buyer digitally signs the completed data structure with an agreed upon algorithm, using an authorizing digital certificate.
3. Optionally, the buyer may be required to send some authorization information to his or her bank.
4. The buyer sends the signed information to the supplier who, in turn, appends additional information and his or her signature.

5. The supplier relays the combined document to the bank for review and processing

The remainder of the workflow, numbers 6, 7, 8, and 9, represent linkage to existing bank-to-bank processes.

An alternate workflow is depicted in Figure 43. This design provides the same information to the supplier's bank with fewer the technology requirements at the supplier site, fewer electronic operations, and simplified control over the format of the form. Because of these advantages, it will be the preferred approach.

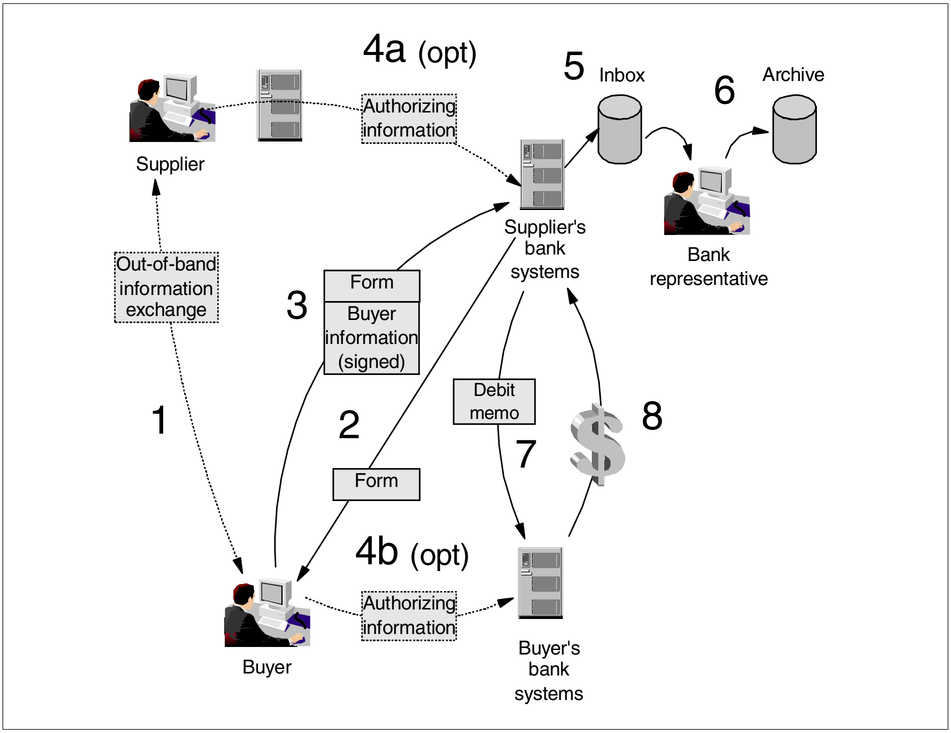


Figure 43. Alternate workflow

The flow represented in Figure 43 is as follows:

1. Represent part of the purchase process in which pertinent information is exchanged in person.
2. Form retrieval could be achieved by HTTP, FTP, or electronic mail.

3. Completed and signed forms could be transmitted via HTTPS or S/MIME. If necessary, one or both parties could notify their relationship bank via independent communication.
4. Optionally, authorization information may need to be sent to the buyer’s bank.
5. The data object represents the combined authorization of the buyer and supplier. This data object must use signing and encryption algorithms, plus a data structure format that can be processed by the bank.

For the purpose of illustration in this scenario, the Payment Authorization Application will use online HTTP form retrieval, along with an S/MIME form submittal between the buyer and the supplier's bank.

Note

There are multiple versions of S/MIME, each supporting different security mechanisms. In real life, the solution will require interoperability tests prior to final component selection to ensure compatibility with electronic mail systems used by all parties.

7.7 Deployment scenario with IBM SecureWay Trust Authority

This section illustrates how IBM Trust Authority can be applied to the PKI requirements in the ITRUST scenario and highlight some important deployment considerations that affect design, integration, and operation for any PKI.

7.7.1 IBM Trust Authority within the ITRUST scenario

As described in the previous chapter, IBM Trust Authority provides an integrated technology base for deploying and managing a Public Key Infrastructure.

Table 7 on page 191 recaps the security requirements discussed earlier in this chapter that the features of IBM Trust Authority address. Note that some of the requirements from Table 3 on page 180 and Table 5 on page 186 are filled by the PKI protocols, services, and mechanisms, by policies, or by advanced security countermeasures, rather than by IBM Trust Authority itself.

Please review section 7.8, “Special deployment considerations” on page 201 for more details about these items.

Table 7. Security requirements and Trust Authority features

Category	Requirements	Trust Authority feature
Authorization	Only authorized individuals should have access to the account review transactions and the query results.	Trust Authority manages the status of certificates via Certificate Revocation Lists on the LDAP by Trust Authority. Designated applications can query this list dynamically.
	Authorization is determined by demonstrating ownership of the private key associated with a valid digital certificate.	Trust Authority creates a digital certificate by signing a public key as a result of registration approval.
	Each designated individual will complete the process to enroll for a digital certificate to be used in conjunction with the Account Review Application.	Trust Authority provides a secure registration process.
	At the account holder's option, authorization of an individual could be restricted by dollar amounts: <$10K, <$100K, or <$1M.	Trust Authority provides the ability to create custom certificate profiles that can include policy extensions to be interpreted by the applications.
Accountability	Given an action or event the bank can determine who did it and what was done.	Trust Authority provides an audit log for all registration and administration actions.
Asset protection	Both the bank and the account holder are responsible for the protection of encryption keys.	The Trust Authority registration process helps to ensure that user keys are kept private when they are created. Trust Authority protects its own keys.
Administration	The bank owns the responsibility to issue, manage, and revoke digital certificates associated with the Account Review Application.	Trust Authority provides a customizable registration program that can produce multiple types of certificates based on parameters.
	Authorization is determined by demonstrating ownership of the private key associated with a valid digital certificate.	Trust Authority provides programmable policy exits to validate an applicant prior to signing the applicant's public key.
Assurance	The bank will conduct periodic tests and reviews of all systems and processes.	Trust Authority provides Audit data that can be reviewed for compliance.
	The bank will periodically request confirmation of compliance from account holders.	Trust Authority can require users to revalidate periodically.
Availability	Account holders will be able to revoke credentials at any time the Account Review Application is available.	Trust Authority provides remote registration authority functions.

7.7.2 Planning, installing, and configuring IBM Trust Authority

Based on the functional architecture descriptions for the Account Review and the Payment Authorization Applications, the following assumptions can be made about the target configuration for Trust Authority:

- The bank will provide an electronic enrollment process for its corporate customers that can be accessed from the Internet.
- Prior to enrollment, a representative from every corporate customer will provide identifying information to the bank for each individual authorized to enroll for a digital certificate, indicating the individual's role and privileges with respect to the Account Review and Payment Authorization Applications.
- Trust Authority components will be distributed among two systems:
  - A Certificate Authority server positioned on a protected network
  - A Registration server with an integrated LDAP directory accessible to user communities, but protected from Internet threats

For simplicity, in this example, the platform for each component will be the Microsoft Windows NT operating system. In real life, platform, operating system, and solution configuration may be different, depending on compliance requirements for participating in ITRUST, and the results of a detailed security threat analysis. Some of these considerations are discussed in 7.8, “Special deployment considerations” on page 201.

Figure 44 on page 193 illustrates the PKI fulfillment framework described earlier in this chapter.

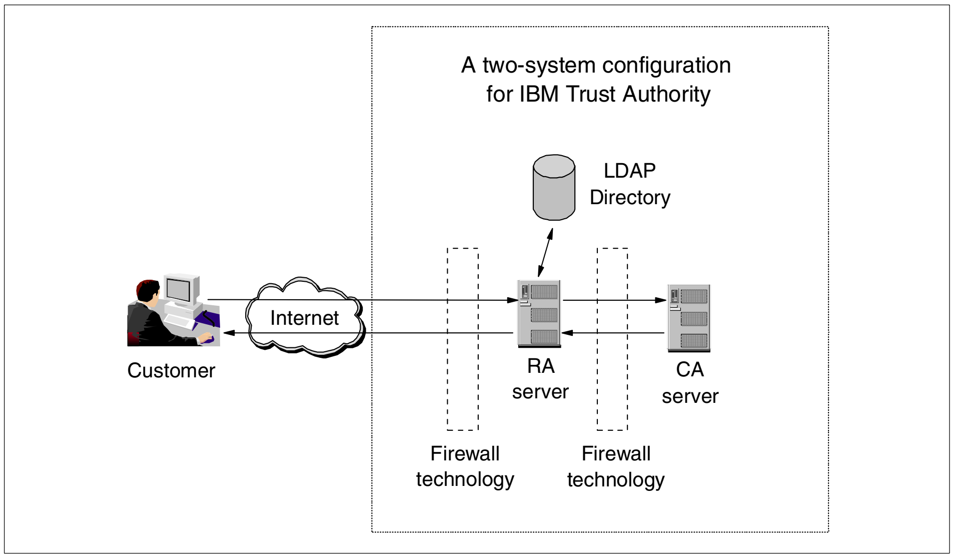


Figure 44. PKI fulfillment framework

For the purpose of this exercise, the bank's CA server will be located on a system with a single network attachment to a private IP network. The bank will use the Web-based registration process provided by Trust Authority and customized to the bank's specifications for images, form content ,and business process workflow. The bank's Registration Authority server will be located on a system with two network attachments, one attached to a private IP network and one attached to a protected network that is addressable from the Internet (see Figure 44).

The *protected network* is intended to address security threats, such as denial of service and unauthorized access, which are common in the Internet environment. The configuration of the protected network, also called a Demilitarized Zone (DMZ), and the hardening of the operating system platforms are beyond the scope of this example.

The LDAP directory will be co-located with the RA server to facilitate the possibility of dynamic certificate validation requests from the bank's Web servers, which might be positioned on the DMZ. As an alternative, a master LDAP directory could have been kept within the private network, and a replicated directory could have been positioned in the DMZ.

Both the RA server and the CA server must have Domain Name Service (DNS) entries with *Address*, or *A* records and reverse resolution, or *PTR* records. The use of hostnames and Domain Name Services is required for validation within the Trust Authority automated configuration process. Domain Name Services and other TCP/IP infrastructure are not shown on in Figure 44 on page 193.

In addition to the standard Trust Authority components, the computer system which will run the RA server will also contain a business-oriented Web server operating on port 80. The Trust Authority-related Web server processes will be configured on alternate ports to avoid port conflicts. These ports will be accessible via links on the default system’s home page.

7.7.2.1 Installing IBM Trust Authority

The installation steps for the Certificate Authority (CA) server system follows the basic steps outlined in 6.4, “Installing Trust Authority” on page 122:

1. Install Windows NT
2. Install NT Fix Pack 5
3. Install DB2
4. Install JDK 1.1.6
5. Install SecureWay Directory client
6. Install Trust Authority, selecting:
   - CA and Audit Server
   - Trust Authority client

Note

The Post Install Configuration process for the CA server must not be run until after the Post Install Configuration process has been completed for the RA server. Refer to section 7.7.2.2, “Configuring IBM Trust Authority” on page 195.

The installation steps for the Registration Authority (RA) server, including LDAP directory, are:

1. Install Windows NT
2. Install NT Fix Pack 5
3. Install DB2
4. Install JDK 1.1.6
5. Install IBM HTTP Server
6. Install WebSphere Application Server 2.0.3.1
7. Install SecureWay Directory

8. Install Trust Authority, selecting:
   - RA server
   - RA Desktop
   - Directory server components
   - Trust Authority client
9. Install a business-oriented Web server of choice, such as the IBM HTTP Server

The system should be rebooted prior to beginning the Trust Authority configuration process.

7.7.2.2 Configuring IBM Trust Authority

After the basic installation steps are complete, the components of Trust Authority are ready to be configured and integrated through a largely automated process that is driven by the configuration parameters entered through the Setup Wizard.

Tracking the installation process and diagnosing errors would be simplified if scrolling is enabled on MSDOS windows. This can be achieved by accessing the **MSDOS Console** icon in the NT Control Panel and increasing the height value of the screen buffer to 200 or more on the proper field within the Layout tab.

When the Trust Authority components are distributed among multiple interconnected computer systems, the configuration program must be run in a predefined sequence, as follows:

1. Run the Post Install Configuration program found in the Trust Authority program folder under the Start Menu of the machine on which the RA server was installed. At the completion of this step, two MSDOS windows will appear minimized on the Task bar:
   - The IBM Trust Authority Servlet Engine
   - The IBM Trust Authority HTTP Server
2. The RA server is now prepared to accept HTTPS connections on port 81 for download and execution of the Setup Wizard applet.
3. Prepare the input file, changing parameters or accepting defaults.
4. Complete the Setup Wizard, which will prepare and run the configuration process (*CfgStart*) on the RA server system.
5. Run the Post Install Configuration program found in the Trust Authority program folder under the Start menu of the machine on which the CA

server was installed. When prompted, supply the hostname of the RA server.

6. Execute the configuration program CfgStart on the CA server from an MS DOS window. The output window will display the progress. The final message will recommend that you initiate the configuration process on the RA server.
7. Continue to execute the CfgStart -i program on alternating systems until final completion messages are obtained. The CfgStart program will maintain a record of completed configuration steps as a checkpoint for successive executions.
8. Stop Trust Authority through the shortcut found in the Trust Authority program folder under the Start menu of each machine.
9. Restart each machine and start Trust Authority and prepare for tests.

Prior to beginning the customization process for Trust Authority, as described in the section that follows, the URL for the default Trust Authority registration process and the Trust Authority Registration Authority Desktop could be added to an easy to remember Web page to the home page for the business Web server.

7.7.3 Customizing Trust Authority

Once the configured Trust Authority systems are running, the designated RA administrator should request a browser certificate from the RA server using the appropriate URL and port number. Follow the instructions given in 6.4.3, “Installing the RA Desktop” on page 128, using the `add_rauser` command to allow the RA administrator to access the RA services.

Copy the RA Desktop (RADInst.exe) and TA Client (TACInst.exe) installation files into a directory on the business Web server for easy download. After the RA Desktop is installed and operational, the basic setup is in place and able to issue and process digital certificates for its user customer community. More customization, however, is required to achieve the desired look-and-feel and to integrate the certificates authority services into the larger ITRUST community.

7.7.3.1 Customizing the Web server pages

Providing a link to Trust Authority certificate services on the business-oriented Web server home page can simplify the customizing required to present a consistent look and feel to the end users. It also reduces the requirement to create aliasing via Domain Name Services.

In the ITRUST scenario, a business Web server was configured to provide a basic home page that identifies the site and offers links to three types of services:

- *Public Services*, such as stock quotes and contact information.
- *New Account Services*, which points to the Trust Authority server on port 8000 (remember that Trust Authority has been configured on alternate ports because the default port 80 is in use by the business Web server).
- *Account Holder Services*, which points to HTML and application pages supporting the Account Review and Payment Authorization applications. These links should specify HTTPS as the protocol, and the directories should be configured to require SSL with client authentication.

The linkage between the business Web pages and the Trust Authority Web pages is shown in Figure 45. Notice the alternate port assignments for the Trust Authority Web server on the top right.

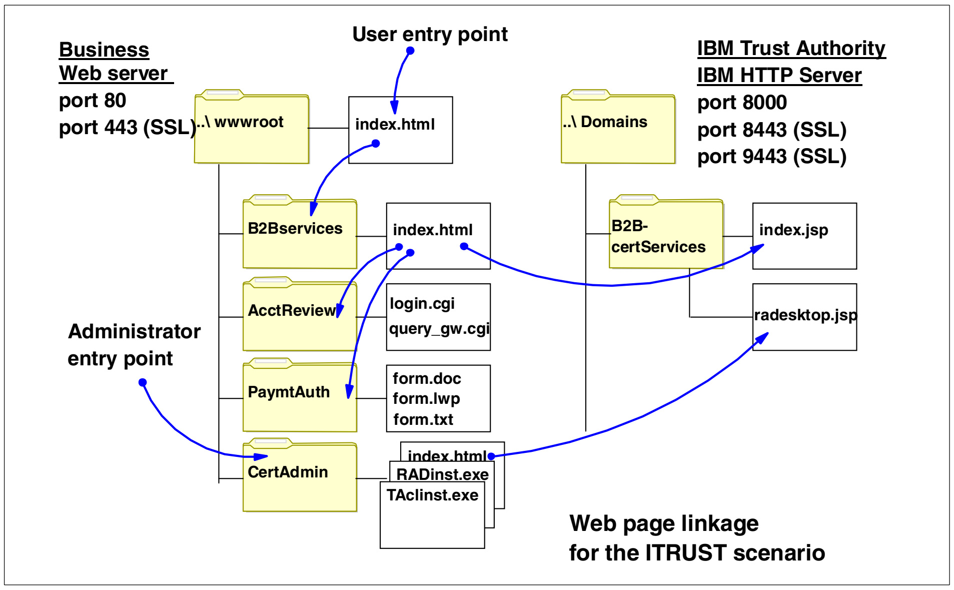


Figure 45. Web page linkages

7.7.3.2 Customizing the approval process

The default certificate registration approval processes for IBM Trust Authority are automatic approval for 1-year browser SSL certificates, manual approval for e-mail certificates, and manual approval for server certificates. This should be satisfactory to support all of the requirements for the ITRUST scenario.

7.7.3.3 Customizing the certificate contents

Aside from providing standard values for the certificate information for the bank as issuer identifier, the user as subject identifier, and date and time values for certification validation periods, two key areas of certificate customization are key usage and certificate policy extensions.

The *key usage* extension documents the ways in which a specific certificate can be used. By providing default certificate profiles for browser certificates, Web server certificates, and electronic mail certificates, IBM Trust Authority has provided the key usage options required by the Account Review Application within the ITRUST scenario as it has been designed.

In the Payment Authorization Application, the *policy extension* could provide a mechanism for communicating information about the guidelines that the buyer's company, or the buyer's bank, applies to the certificate holder's ability to initiate an electronic transfer.

In the ITRUST scenario, in order for a payment authorization policy extension to be useful to every bank, the structure and assigned values for the field should be part of the ITRUST specification. There is a further discussion of policy extensions in 7.8, “Special deployment considerations” on page 201.

7.7.3.4 Certifying the Certificate Authority key and credentials

When the configuration process for the Trust Authority installation is complete, the bank has an operational Certificate Authority service capable of issuing certificates to its users; however, since the signing certificate has been “self-signed,” the user certificates will not be recognized at other banks.

There are two methods of linking the certificates issued by one Certificate Authority service (ITRUST bank) to those issued by another Certificate Authority service (ITRUST bank). Those methods are cross-certification and certificate hierarchy.

In our example scenario, we assume the ITRUST consortium has selected certificate hierarchy as the linking mechanism.

At the time of this writing, IBM Trust Authority V3.1 utilizes the PKIX CMP online protocols to perform cross-certification or certificate hierarchy linkage. The implication in the ITRUST scenario is that the bank's CA server must have network connectivity to the ITRUST root CA server.

The processes for linking a Trust Authority installation with another Certificate Authority via cross-certification or into a certificate hierarchy are similar. To complete either the cross-certification or hierarchy processing, network connectivity must be available. For hierarchy, the self-signed CA signing Certificate of your CA should be removed from the directory. It will be replaced with the hierarchy certificate at the end of the process.

The steps are documented in the *Trust Authority Administration Guide*. The general steps are as follows:

1. Access the registration service for the higher level CA, or the CA with which to be cross-certified, from any Web browser.
2. Complete a preregistration form, making sure to accurately enter the distinguished name for your CA on the form, including CN, OU, and C.
3. Save the preregistration confirmation for and the preregistration file on disk.
4. Transfer the files to the CA server.
5. Enter the `CaCertRq` command from a command line prompt, supplying the information provided from the preregistration process (see the following explanation).

The parameters supplied to the CaCertRq program determine which operation is performed. As with other aspects of PKI, interoperability issues must be evaluated.

For a hierarchy, the command is of the following type:

```
CaCertRq -h -m .. -r preregreq-filename -P 1835 prereg-password
```

When the process has been completed, the updated CA signing certificate can be downloaded from the registration page from your CA. Note the updated fields and issue date on the certificate.

7.7.4 Application-related considerations

Application design is a product of the business process, the creativity of the designer, and the tools that are available. The PKI protocols, services, and mechanisms used within application design represents the PKI deployment framework as described earlier.

The Account Review and Payment Authorization Application described in this scenario are admittedly simplistic; however, they do achieve the desired function without burdening the user beyond having applications and services that are commonly available, that is, a Web browser and an S/MIME-capable electronic mail client.

The integration framework for the PKI represents the components selected and how those components are connected and used within the solution. A great challenge for application designers lies in the open standards and components found in the Internet environment. Quite often, the technical functions and the interoperability issues associated with browsers, Web servers, and electronic mail clients are not easily discovered or understood by an inexperienced integration team.

Here are some options for the Account Review Application:

- Static Web pages that are an image of printed pages could be stored in a protected area of a Web application server with access control limited to designated certificate holders.
- Transaction and/or database query gateways could translate and/or relay Web browser requests to backend systems.
- On IBM OS/390, the certificate holder's identity could be mapped to a RACF profile under which transaction server (for example, CICS), or database query transactions, are run.

In the ITRUST example scenario as described in previous section, dynamic credential checking within the Account Review Application either by the Web server or application process would be appropriate. Here are some options for the Payment Authorization Application:

- An electronic copy of a document that provides form input can be stored on the business Web server, retrieved on demand, and returned as an attachment to a signed and encrypted electronic mail message
- An HTML form with imbedded JavaScript commands can be accessed from the business Web server. The submit button on the form could result in a signed and encrypted response that is sent via electronic mail or via a POST operation to a Web server application program on the bank's business Web server.

Information and examples for using JavaScript to sign forms can be found, for example, at the following URL:

`http://developer.netscape.com/tech/security/formsign/stex.html`

Programmatic checking of credentials within the Payment Authorization example scenario program would be difficult, because of the protocols and services within S/MIME flows. As an alternative, it would be feasible for the buyer's bank to provide a query-response credential status check on its business-oriented Web server.

Please refer to 7.8, “Special deployment considerations” on page 201, for more information on dynamic credential checking and other design issues.

7.8 Special deployment considerations

There are many options and issues associated with deploying PKI certificates and integrating PKI components into business applications. These options and issues listed below represent some of the critical variables within PKI design that require careful consideration:

- CA key management (key length, key generation, key protection)
- Policy extensions within certificates
- Dynamic credential validation
- Long-term storage of keys and protected documents
- Expanded use of directory services
- Component integration and interoperability
- Server scalability for CA operations
- Internet threats and countermeasures
- Policies and business agreements

When the successful deployment of a Public Key Infrastructure is dependent on the resolution of one or more of these considerations, the design and integration team should have current expertise and use formal I/T solution design methods.

7.8.1 CA key management

Key management involves key length, key generation, and key protection.

Key length

There are three major considerations associated with viable cryptography:

- Protection of secrets
- Strength of algorithm
- Strength of keys

If a weakness in any one of these areas is considered a valid threat, the trustworthiness of the electronic trust model is in jeopardy. Protection of secrets is largely an issue addressed by policy. Selection of key strength and algorithm type is a product of perceived threat, available technology, and processing load on the computing systems.

In the dynamic environment of the Internet, managing reasonable certificate validity periods is one technique of maintaining a balance between the length of keys, the complexity of algorithms, and the processing requirements on computing systems.

Key generation for CAs
For Trust Authority, the keys that the Certificate Authority uses for signing operations are generated during the automated configuration process, based on the parameters supplied to the Setup Wizard.

Key generation for servers and clients
The X.509 recommendation recognizes three different valid options for key generation: 1) local, 2) by the CA, and 3) by a remote trusted third party. Most client and server software provide for local key generation, key storage, and cryptographic operations within the product. Some devices, particularly routers and other special purpose devices, may not have the processing capacity to generate high-assurance keys. In these situations, administrative controls play an important role in key generation and distribution.

The advantage of local key generation, that is, key generation on the computer system that will perform cryptographic operations using the public/private key pair, is that the custody of the private key is the sole responsibility of the user(s) and/or administrators of the computer system. The consequences of key compromise or misuse rests on those who manage the computer system.

Protection of keys for CAs
The private signing key for a Certificate Authority is an extremely important item. The protection of this key is critical to the viability of the electronic trust model for the entire PKI. These keys can be stored in hardware, and the computer systems in which the keys are used can be protected from physical access or remote electronic access.

As an alternative to a dedicated and protected CA server, and when only sporadic use of the CA is required, a laptop computer, say an IBM ThinkPad, could be considered as the platform of choice. This provides some interesting security implications and potential advantages for an operational PKI:

- The ThinkPad containing the Trust Authority CA server could be removed from the network and physically secured when not in use, preventing the possibility of theft of equipment and keys, plus eliminating the problem of electronically tampering with unattended systems.
- The ThinkPad platform has an option for encrypting the hard drive.
- The ThinkPad platform has an option to prohibit system boot if the controlling Smartcard is not inserted.

- Recovery from a hardware malfunction could be achieved by inserting the hard drive into a spare system.

Protection of keys for servers and clients
The private (signing and/or encryption) key(s) for servers and clients are important items as well. Depending on the individual or computerized process that the key represents, the protection of these keys may be critical to trusting an identity, to the keeping information private, to conducting financial transactions, or to executing legal contracts.

These keys can be stored in hardware in a variety of devices, such as Smartcards. The computer systems in which the keys are used can be, or should be, protected from physical access or remote electronic access.

7.8.2 Policy extensions within certificates

The X.509 recommendation provides for optional information within certificates known as *policy extensions*. Policy extensions can be a useful means of conveying privileges or limitations about the certificate, or about the person or process the certificate identifies.

The use of policy extensions within certificates should be considered based on the business environment and the overall business process. When policy extensions are used to communicate user attributes, the user's certificate needs to be revoked and reissued whenever the role of the user, or the values of the attribute, change.

The information within policy extensions is dependent on following a formal process for obtaining a globally unique electronic bit string known as an *object identifier*. This identifier is assigned to an owner by an recognized standards body. The object identifier and associated values become part of the certificate contents following a custom certificate profile defined to the IBM Trust Authority installation.

7.8.3 Dynamic credential validation

For electronic transactions to be trusted, the credentials on which those transactions are based must be valid. However, the issuer of the certificate is typically different from the person or process that is relying on the validity of the credential. The developers of the X.509 recommendation, and others, recognized the need for dynamic credential validation and have provided alternatives, including Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP). The availability and confirmed interoperability of products to support CRLs and OCSP should be evaluated during the solution design.

At the time of this writing, IBM Trust Authority supports the publishing of CRLs to the LDAP directory, and the indirect support of OCSP servers through the published CRL. Further integration with OCSP is being examined. IBM Trust Authority provides registration administration functions for revocation.

Unfortunately, neither CRL processing nor OCSP is a standard function within the applications and components commonly associated with PKI: Secure Sockets Layer (SSL), S/MIME, or IPSec. Most Web browsers do not currently have the ability to check to see if the server certificate presented during an SSL handshake has been revoked. S/MIME electronic mail clients do not have the ability to check to see if the intended recipient of an encrypted message has been revoked.

Some Web servers, such as IBM OS/390 Web server, have extended the SSL certificate validation protocol to include CRL processing as a centralized service of the server. The IBM HTTP Server on S/390 can be configured to query an LDAP directory for CRL processing prior to serving a protected page or launching an application process.

In the case where the business Web server platform does not provide a centralized CRL check, trusted Web applications should implement credential validation checks independently via software toolkits. When several applications on a single Web server implement certificate validation, duplication of processing and communication workload results.

7.8.4 Long-term storage of keys and protected documents

Requirements for retention of keys (also called key escrow) varies based on the business process and the legal and risk management requirements of the business process.

In the way that the Account Review Application was defined, the PKI operations performed by SSL components do not require escrow of keys, because the encryption and signing operations are performed on documents as they are transferred, not on documents as they are stored. A lost or expired SSL certificate is usually replaced with a certificate based on a new key pair.

The Payment Authorization Application could be designed to save forms that have been encrypted and signed; however, the only key that would need to be saved is the private key of the receiving user or process at the supplier's bank. Since key escrow is, and archival of encrypted documents are, burdensome, every effort should be taken to apply these processes appropriately in a design.

7.8.5 Expanded use of directory services

As indicated earlier in this book, and in the product documentation, IBM Trust Authority can configure, initialize, and use an LDAP directory to store certificates and CRLs. LDAP directories have additional uses. One use that might be of value to banks is the ability to provide electronic name and address directories (also called directory white or yellow pages) to its customers as a value-added service or the communities served by its customers.

While the SecureWay Directory schema could be configured as a whitepages locator or as a service that distributes electronic mail certificates to facilitate signed and encrypted electronic mail, this is not part of a default IBM Trust Authority installation.

7.8.6 Component integration and interoperability

A key factor in successful PKI deployment is the interoperation of all components, not only for the desired application functions, but also for the functions that support the PKI fulfillment framework. In cases where multiple software and hardware components must be integrated within one platform, care must be used on selecting individual mechanisms for cryptographic support of PKI fulfillment and application flows.

The PKI-related services include:

- Key generation
- Key distribution (certificate request and import)
- Key storage
- Cryptographic operations (encrypt/decrypt and sign/verify)

The PKI-related mechanisms include:

- System services
- Cryptographic system interfaces and controls
- Cryptographic application programming interfaces (APIs)

The PKI-related protocols include:

- Secure payload for certificate request/response
- Secure channel for certificate request/response

Certificate request and delivery protocols are particularly important. Client systems, such as Web browsers, the Trust Authority client, server systems, such as Web servers, application servers, Certificate Authority servers, and so on, may implement one of several certificate request formats, communicate the request through one of several means, and require that the certificate be delivered in one of many formats, by way of a secure channel, or a secure payload over an open channel.

7.8.7 Server scalability for CA operations

There are many factors of the business environment that must be considered when designing the fulfillment framework for a Certificate Authority. These can be expressed in various ways:

- Business cycle, that is, whether or not the time available to enroll the target user communities is constrained. A CA service offered by a health care plan in which all subscribers are to be issued electronic identity cards in the one month period of open registration has a different business cycle than a bank with a relatively constant number of new accounts.
- Volume characteristics, that is, the total number of entities (users and processes) in the target communities.
- Time characteristics, that is, whether the time periods for registration workload can be mapped to a business schedule or time zone.
- Startup demand curve, that is, the rate at which the registration workload arrives during the initial deployment if it is different than what is anticipated after the initial enrollment period is completed.
- Steady state demand curve, that is, the rate at which the registration workload arrives after the initial enrollment period is completed.
- Re-issuance demand curve, that is, the rate at which the re-issuance workload arrives after the initial enrollment period is completed.
- Off-peak deferrable workload, that is, the amount and timing of the registration workload that can be leveled.

Here are two examples for the ITRUST scenario, which would result in different CA server configurations:

- If the business environment requires that all enrollments be completed online via automated approval and the certificates be available within 30 seconds
- If the business environment requires that all enrollments be completed via manual approval and the certificate be available the next business day

7.8.8 Internet threats and countermeasures

I/T security recognizes two categories of threat: insider threat and outsider threat. Outsider threats have been described in various ways. For the purposes of this example, outsider threats attributable to business operations on the Internet include unauthorized access, denial-of-service, repudiation, accidents, poor design, malice, and fraud. These threats are interrelated.

Applications that integrate PKI protocols, services, and mechanisms can reduce threats for unauthorized access and repudiation as long as threats due to accidents, poor design, denial-of-service, and other malicious behaviors do not occur, or are adequately addressed.

The extent to which the threats are addressed in relation to the business risk represents a measure of trustworthiness of the application and, therefore, the trustworthiness of the business process.

7.8.9 Policies and business agreements

There are limits on the trustworthiness of I/T systems. Trustworthiness of business processes that rely on I/T systems can be managed most effectively if there are effective organizational policies and enforceable business agreements.

7.9 Summary and caveats

In this scenario, each bank in ITRUST would be required to design and test processes for maintaining the accurate status of the credentials that it has issued. The trustworthiness of digital certificates issues by an ITRUST bank is dependent on many things, including solid business policies and practices, enforceable business agreements, strict adherence to compliance criteria, and best practices for I/T security.

The trustworthiness of the business process is built on the trustworthiness of the digital certificates, plus the trustworthiness of the business application that the information technology components support.

7.10 IBM Global Services

IBM Global Services has consulting, design, integration, and assessment expertise to assist in every phase of PKI deployment projects. More information can be obtained at:

`http://www.ibm.com/services/e-business/security.html`

Part 4. Appendices

Appendix A. Introduction to LDAP

The term Lightweight Directory Access Protocol (LDAP) has been frequently used throughout this book. IBM SecureWay Trust Authority uses an LDAP directory for storing certificate and CRL information. This appendix gives you a short introduction to LDAP. An LDAP server ships with IBM SecureWay Trust Authority, and it is also available from IBM free (or at no additional cost) for various IBM and non-IBM operating system platforms.

A.1 The history

In 1988, the Comite Consultative International Telephonique et Telegraphique (CCITT), which is now International Telecommunications Union - Telecommunication Standardization Sector (ITU-T), created a standard for directory services, the X.500 standard. It became ISO standard 9594, *Data Communications Network Directory, Recommendations X.500-X.521*, in 1990. This set of standards is still commonly referred to as *X.500*.

X.500 defines a directory that can universally be used for large amounts of data and, today, it is typically used by national telephone organizations for their huge online telephone directories.

To access an X.500 directory, a client uses the *Directory Access Protocol* (DAP) that was defined along with the X.500 standard. Unfortunately, DAP is a rather complex protocol that cannot be easily supported on thin clients, such as desktop computers. X.500 was, therefore, limited to powerful computers and large-scale implementations. The requirement to access centralized directories from slim clients, however, became apparent as the value of centralized directories became obvious for various reasons, such as cost effectiveness for management.

Work being done at the University of Michigan (UMICH) and at Netscape Communications Corp. has led to a simplified version of DAP, which was then called *Lightweight Directory Access Protocol* (LDAP). LDAP supports most of the features of DAP, but lacks some of the complex, seldom-used functions. Its implementation is relatively simple and it can, therefore, be used by desktop applications. The current version of LDAP is Version 3 (LDAPv3) although some products may still be at Version 2.

A.2 Protocol or directory?

As the name implies, LDAP, strictly speaking, is only a protocol that runs over TCP/IP. The LDAP protocol standard not only includes low-level network protocol definitions, but also data representation and handling functionality. A directory that is accessible through LDAP is, therefore, commonly referred to as an LDAP directory. It must be noted and understood, however, that the LDAP standard does by no means define how the data is actually stored in the directory.

Initially, LDAP was designed such that thin clients could access an X.500 directory through a gateway server that did some translation between LDAP and DAP (see Figure 46).

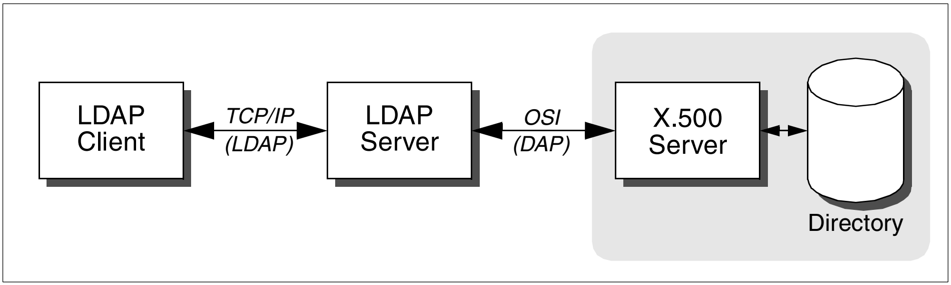


Figure 46. LDAP access to X.500

It did not take long before vendors developed directories that could handle the LDAP protocol natively, rather than performing a translation between LDAP and DAP (see Figure 47). The IBM implementation of an LDAP directory is the *SecureWay Directory*, which is available on AIX, Windows NT, Sun Solaris, OS/400, and OS/390.

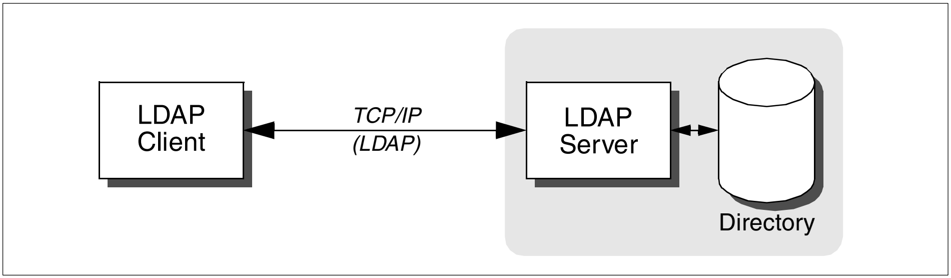


Figure 47. Stand-alone LDAP server

Vendors choose different implementations for the storage of the directory data. While most vendor implementations use flat file databases, the IBM SecureWay Directory uses the high-performance, highly-scalable DB2 relational database as backing store for its LDAP directory.

A.3 What is a directory?

The question of what a directory is has not been answered so far. To look at a simple example, most directories store information about people, much the same way as a printed phone directory (white pages) does. The entries are usually organized in a hierarchical way that makes management and searching easy and intuitive.

LDAP directories are much more powerful and by no means limited to name, phone number, and address entries. In fact, an LDAP directory can store (and subsequently retrieve) almost any kind of data. The type of data that can be stored in an LDAP directory is defined by the directory schema, which can be extended and adapted to meet certain requirements. The task of defining a directory schema and the directory information tree (the hierarchy in which entries are to be stored in the directory) can be compared to the design of a relational database. Thorough analysis of application requirements, corporate standards, and data definitions is necessary to design a directory schema and the directory information tree (DIT). Fortunately, LDAP server products, such as the IBM SecureWay Directory, come with a comprehensive schema that can (and should) be used unless specific requirements dictate alterations and/or additions.

Standards bodies work on standards and proposals for a huge variety of possible data definitions and their hierarchical relationship. IBM supports these standards and proposals by actively participating in the respective organizations and by implementing the results in the IBM SecureWay Directory. The most important standards body for LDAP is the Internet Engineering Task Force (IETF), where representatives of IBM and other key industry leaders are actively supporting these activities.

Micro and macro directories are typically present in large numbers in every organization. For example, most modern operating systems, such as UNIX or Windows 9X/NT, store user account information either locally or on departmental servers. Network operating systems, such as NetWare (Novell), carry some sort of user databases, too. Departments may have their own “private” employee database, while at the same time, corporations have large human resource databases. Note that these examples only included information about people. In addition to this, operating systems store large amounts of data about their system configuration and about other network resources, such as printers and servers.

In fact, most such information is even stored at multiple locations in multiple micro or macro directories, causing a nightmare for administration and maintenance.

A major reason why LDAP has quickly gained so much interest is its potential to provide a single, standards-based directory for all this distributed kind of information. Most directory-related vendor products, including Novell NDS and Microsoft Active Directory, either already support or have announced support for LDAP. Even many wide-spread clients, such as Netscape Navigator or Microsoft Internet Explorer, have been providing support for LDAP since several product releases. Such client support of a Web browser ranges from the simple ability to access public LDAP directories for people lookup to the advanced ability to store personal preferences, including bookmarks, in a centralized LDAP directory to support travelling users.

Many public directory providers, such as Four11 (`http://www.four11.com`), offer public LDAP services that you can connect to using any LDAP-capable client, such as Netscape Communicator.

A.4 The LDAP information model

The LDAP information model is based on a subset of the X.500 information model. Information in an LDAP directory is stored in *entries* that contain *attributes*. Attributes are typed in the form of *type=value* pairs in which the *type* is defined by an object identifier (OID) and the *value* has a defined syntax. Attributes can be *single-valued* (for example, a person can only have one date of birth) or *multi-valued* (a person can have multiple phone numbers).

Each entry in an LDAP directory has a unique *distinguished name* (DN). The directory schema defines rules for DNs and what attributes an entry must and/or may contain. To organize the information stored in directory entries, the schema defines object classes. An object class consists of *mandatory* and *optional* attributes.

Object classes can inherit from other object classes, which provides a method for easy extensibility (for example, new object classes can be defined by just adding new attributes to existing object classes). The concept of object classes, their different declarations, and use is beyond the scope of this short overview. Please refer to specific LDAP reference documentation for more details.

A.5 More LDAP features

The previous sections of this appendix briefly introduce LDAP. This section lists and explains some additional features of current LDAP implementations that might not be obvious from the descriptions in the previous sections.

Scalability
LDAP directories, particularly when they are backed up by a relational database as in the IBM SecureWay Directory, are highly scalable. Large directories with millions of entries are possible with excellent performance. Due to the common standard base, another scalability factor is the easy step-up possibility to more powerful hardware and software. LDAP does not rely on a specific operating system and it is vendor independent.

Availability
LDAP supports replication and splitting of namespaces. Replication allows multiple LDAP servers to store the same directory contents, and thus, clients have alternative servers available should one fail. Splitting allows parts of the whole directory (namespace) to be stored in different servers at different locations. This not only increases availability (no single point of failure), but also offers an easy way for distributed management.

Security
Security features are provided such that unauthorized access to data can be prevented. Secure communication protocols, such as SSL, and authentication mechanisms, along with access control lists (ACLs) for data entries guarantee a maximum level of security. This even allows for storage of sensitive data in an LDAP directory.

Manageability
Early versions of LDAP directory products came with command line tools, and configuration had to be done by editing configuration and schema files. Current versions, such as the IBM SecureWay Directory Version 3.1, facilitate a graphical user interface for both system administration and directory data administration. Dynamically extensible schema allows you to extend the directory schema without interrupting the service.

Standardization
The LDAP protocol and many related client/server capabilities, application program interfaces (APIs), and data definitions are defined by either official standards or corresponding Request for Comments (RFCs) are on the standards track. *Lightweight Directory Access Protocol (v3)*, RFC 2251, for example, defines the basic LDAP protocol. Other features are, although widely accepted and implemented, defined in Internet Drafts. Much of this

work is done by the Internet Engineering Task Force (IETF) and the Distributed Management Task Force (DMTF).

A.6 Where to get more information

If you need more information about LDAP, consider these two IBM redbooks:

- *Understanding LDAP*, SG24-4986
- *LDAP Implementation Cookbook*, SG24-5110

The IBM directory Web site can be accessed at:

- `http://www.ibm.com/software/enetwork/directory`

Other pertinent information can be found at:

- `http://www.ietf.org`
- `http://www.dmtf.org`
- `http://www.umich.edu/~dirsvcs/ldap`

Appendix B. Special notices

This publication is intended to help architects, planners, and integrators to plan and implement a Public Key Infrastructure. The information in this publication is not intended as the specification of any programming interfaces that are provided by any of the IBM products mentioned. See the PUBLICATIONS section of the IBM Programming Announcement for the respective product for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have

AIX ®	AIX/6000
AS/400 ®	CICS ®
DB2 ®	eNetwork
IBM Registry	IBM ®
Netfinity ®	OS/2 ®
OS/390 ®	OS/400 ®
RACF ®	RS/6000 ®
S/390 ®	SecureWay ®
System/390 ®	ThinkPad ®
VisualAge ®	WebSphere
World Registry	

Appendix C. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 IBM Redbooks

For information on ordering these publications see "How to get IBM Redbooks" on page 225.

- *LDAP Implementation Cookbook*, SG24-5110
- *Understanding LDAP*, SG24-4986
- *Understanding IBM SecureWay FirstSecure*, SG24-5498

C.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

C.3 Other resources

The following documentation, along with additional publications for the Trust Authority product, is available online on the Trust Authority library Web site (http://www.ibm.com/software/security/trust/library):

- *Changing Trust Authority Bootstrap Values*
- *Installing Trust Authority on AIX*

- *Installing Trust Authority on Windows NT*
- *Installing the Trust Authority RA Desktop*
- *Installing the Trust Authority Client Application*
- *Trust Authority Up and Running* (Also available with the product)
- *Trust Authority Configuration Guide*
- *Trust Authority User's Guide*
- *Trust Authority Registration Authority Desktop Guide*
- *Trust Authority System Administration Guide*
- *Using the SecureWay Directory With Trust Authority*
- *Using the 4758 Coprocessor With Trust Authority*

C.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://csrc.nist.gov/encryption/aes/aes_home.htm`
- `http://developer.netscape.com/docs/manuals/security.html`
- `http://theory.lcs.mit.edu/~cis/sdsi.html`
- `http://www.cdsasecurity.com`
- `http://www.dmtf.org`
- `http://www.four11.com`
- `http://www.ibm.com/security/cryptocards`
- `http://www.ibm.com/security/services/index.html`
- `http://www.ibm.com/software/commerce/payment`
- `http://www.ibm.com/software/enetwork/directory`
- `http://www.ibm.com/software/security/trust`
- `http://www.ibm.com/software/security/trust/library`
- `http://www.ietf.org/html.charters/ipsec-charter.html`
- `http://www.ietf.org/html.charters/pkix-charter.html`
- `http://www.ietf.org/html.charters/pppext-charter.html`
- `http://www.ietf.org/html.charters/spki-charter.html`
- `http://www.lotus.com`
- `http://www.opengroup.org/public/tech/security/cdsa/index.htm`

- `http://www.opengroup.org/security/cdsa`
- `http://www.rsa.com/rsalabs/pubs/PKCS`
- `http://www.rsa.com/smime`
- `http://www.setco.org`
- `http://www.setco.org/set_specifications.html`
- `http://www.umich.edu/~dirsvcs/ldap`
- `http://www.verisign.com/repository/CPS`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

ABA. American Bar Association.

AC. Attribute Certificate.

ACL. Access Control List.

AES. Advanced Encryption Standard.

API. Application Program Interface.

ASN. Abstract Syntax Notation.

BER. Basic Encoding Rules.

BERT. Basic Event Representation Token.

CCITT. Comite Consultatif International Telephonique et Telegraphique.

CA. Certificate Authority.

CDSA. Common Data Security Architecture.

CHAP. Challenge Handshake Authentication Protocol.

CICS. Customer Information Control System.

CL. Certificate Library.

CMP. Certificate Management Protocol.

CMS. Cryptographic Message Syntax.

CPS. Certificate Practice Statement.

CR. Certificate Repository.

CRMF. Certificate Request Message Format.

CRL. Certificate Revocation List.

CSP. Cryptographic Service Provider.

CSRC. Computer Security Resource Clearinghouse.

CSSM. Common Security Services Manager.

DAP. Directory Access Protocol (X.500).

DB2. Database 2.

DCE. Distributed Computing Environment.

DCS. Data Certification Server.

DER. Distinguished Encoding Rules.

DES. Data Encryption Standard.

DIT. Directory Information Tree.

DL. Data Library.

DMTF. Distributed Management Task Force.

DMZ. Demilitarized Zone.

DN. Distinguished Name.

DNS. Domain Name Service.

ECDSA. Elliptic Curve Digital Signature Algorithm.

EE. End-Entity.

FIPS. Federal Information Processing Standard.

FTP. File Transfer Protocol.

HTTP. Hypertext Transfer Protocol.

IBM. International Business Machines Corporation.

ICL. Issued Certificate List.

IETF. Internet Engineering Task Force.

IHS. IBM HTTP Server.

IP. Internet Protocol.

IPSec. IP Security.

ISO. International Organization for Standardization.

ISP. Internet Service Provider.

ITSO. International Technical Support Organization.

ITU-T. International Telecommunications Union - Telecommunications.

JDK. Java Development Kit.

KEA. Key Exchange Algorithm.

KRS. Key Recovery Server.

KRSP. Key Recovery Service Provider.

LDAP. Lightweight Directory Access Protocol.

MAC. Message Authentication Code.

NIST. National Institute of Standards and Technology.

OCSP. Online Certificate Status Protocol.

OID. Object Identifier.

OSI. Open System Interconnect.

PAP. Password Authentication Protocol.

PEM. Privacy Enhanced Mail.

PGP. Pretty Good Privacy.

PKC. Public Key Certificate.

PKCS. Public-Key Cryptography Standards.

PKI. Public Key Infrastructure.

PKIX. Public Key Infrastructure for X.509 Certificates.

PPP. Point-to-Point Protocol.

QOS. Quality of Service.

RA. Registration Authority.

RACF. Resource Access Control Facility.

RFC. Request for Comment.

RSA. Rivest, Shamir, and Adleman.

SCVP. Simple Certificate Validation Protocol.

SDSI. Simple Distributed Security Infrastructure.

SET. Secure Electronic Transaction.

S/MIME. Secure Multipurpose Internet Mail Extensions.

SPKI. Simple Public Key Infrastructure.

SSL. Secure Sockets Layer.

TCP/IP. Transmission Control Protocol/Internet Protocol.

TDA. Temporal Data Authority.

TP. Trust Policy.

TSA. Time Stamp Authority.

TTP. Trusted Third Party.

VPN. Virtual Private Network.

UCS. Unicode Character Set.

UPS. Uninterruptable Power Supply.

URI. Universal Resource Identifier.

URL. Uniform Resource Locator.

UTF. UCS Transformation Format.

Index

IBM Redbooks evaluation

Deploying a Public Key Infrastructure
SG24-5512-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to `redbook@us.ibm.com`

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction __________

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

____

____

____

____

What other Redbooks would you like to see published?

____

____

____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

____

____

____

____

SG24-5512-00
Printed in the U.S.A.

Deploying a Public Key Infrastructure

SG24-5512-00