16 CHAPTER Public-Key Infrastructures

Since public keys in asymmetric cryptosystems need not be kept secret, key management in those systems is simpler than in symmetric schemes. Private keys, however, must be kept secret. Also, public keys must be protected from falsification and abuse. Therefore, appropriate *public-key infrastructures* (PKI) must be set up. They are responsible for key distribution and management. In this chapter, we describe how such public-key infrastructures work.

16.1 Personal Security Environments

16.1.1 Importance

If Bob wants to generate signatures or decrypt documents using a public-key system, then he needs a private key. Bob must keep this key secret because everybody who knows the key can sign messages in Bob's name or decrypt secret documents that were sent to Bob. Therefore, Bob needs a *personal security environment (PSE)* in which his private keys are securely stored. Since the private keys should not leave the PSE, it also does the signing or decrypting.

Frequently, the PSE also generates the private keys. If the private keys are generated elsewhere, then at least the generating institution knows Bob's secret keys, which may corrupt the security of the system. On the other hand, secure key generation may require resources not present in the PSE. For example, for RSA keys random primes of a fixed bit length are required. In particular, the key generating environment must generate large, cryptographically secure, random numbers. If the random number generator of the PSE is weak, then the public-key system is insecure. It may therefore make sense to have the RSA keys generated by a trusted institution.

16.1.2 Implementation

The more sensitive the documents that are signed or encrypted, the more secure the PSE must be. A simple PSE is a file in Bob's home directory that can be accessed only after entering a secret password. This password may, for example, be used to decrypt the information. The security of a software PSE relies on the security of the underlying operating system. One may argue that operating systems must be very secure anyway and that they are therefore able to protect the PSE. Operating systems, for example, prevent unauthorized users from becoming administrators. On the other hand, it is well known that with sufficient effort the security of most operating systems can be successfully attacked. Therefore, a software PSE is not adequate for applications that require high security.

It is more secure to put the PSE on a smart card. Bob can carry his smart card in his wallet. If the card is in the smart card reader, it only permits very limited access. Manipulating its hardware or software is very difficult (although successful attacks have been reported). Unfortunately, computations on smart cards are still very slow. Therefore, it is impossible to decrypt large documents on a smart card, so public keys encrypt session keys which, in turn, are used to encrypt the documents. The encrypted session key is appended to the encrypted document. The smart card only decrypts the session key. The decryption of the document is then done on a fast PC or workstation.

16.1.3 Representation problem

Even if Alice uses a smart card for signing, there is still a severe security problem. If Alice wants to sign a document, she starts a program on her PC, which sends the document or its hash value to the smart card, where it is signed. With some effort, the attacker, Oscar, can manipulate the signing program on Alice's PC such that it sends a document to the smart card that is different from the one that Alice intended to sign. Because the smart card has no display, Alice is unable to detect this fraud. It is therefore possible that Alice could sign documents that she never wanted to sign. This problem is called the *representation problem* for signatures. The more important documents are for which digital signatures are accepted, the more dramatic the representation problem becomes. The problem is solved if Alice sees what she signs. For this purpose, Alice's PSE needs a display. One possibility is to use a cellular phone as a PSE. But its display is very small. Hence, the documents that can be signed securely on it are rather short. It depends on the solution of the representation problem whether digital signatures can be used to replace handwritten signatures.

16.2 Certification Authorities

If Alice uses a public-key system, it is not sufficient for her to keep her own private keys secret. If she uses the public key of Bob, she must be sure that it is really Bob's key. If the attacker, Oscar, is able to substitute his own public key for Bob's public key, then Oscar can decrypt secret messages to Bob and he can sign documents in Bob's name.

One solution of this problem is to establish trusted authorities. Each user is associated with such a *certification authority* (CA). The user trusts his CA. With its signature, the CA certifies the correctness and validity of the public keys of its users. The users know their CA's public key. Therefore, they can verify the signatures of their CA. We now explain in more detail what a CA does.

16.2.1 Registration

If Bob becomes a new user of the public-key system, then he is registered by his CA. He tells the CA his name and other relevant personal data. The CA verifies Bob's information. Bob can, for example, go to the CA in person and present some identification. The CA issues a user name for Bob that is different from the user name of all other users in the system. Bob will use this name, for example, if he signs documents. If Bob wants to keep his name secret, then he may use a pseudonym. Then, only the CA knows Bob's real name.

16.2.2 Key generation

Bob's public and private keys are generated either in his PSE or by his CA. It is recommended that Bob not know his private keys, because then he cannot inform others about those keys. The private keys are stored in Bob's PSE. The public keys are stored in a directory of the CA. Clearly, the keys must be protected while they are communicated between Bob and his CA.

For each purpose (for example, signing, encryption, and identification), a separate key pair is required. Otherwise, the system may become insecure. This is illustrated in the next example.

Example 16.2.1
If Alice uses the same key pair for signatures and challenge response authentication, then an attack can be mounted as follows. Oscar pretends that he wants to check Alice's identity. As a challenge, he sends the hash value $h(m)$ of a document m. Alice signs this hash value, assuming that it is a random challenge. But in fact Alice has signed a document, which was chosen by Oscar, without noticing.

16.2.3 Certification

The CA generates a *certificate*, which establishes a verifiable connection between Bob and his public keys. This certificate is a string, which is signed by the CA and contains at least the following information:

1. the user name or the pseudonym of Bob,
2. Bob's public keys,
3. the names of the algorithms in which the public keys are used,
4. the serial number of the certificate,
5. the beginning and end of the validity of the certificate,
6. the name of the CA,
7. restrictions that apply to the use of the certificate.

The certificate is stored, together with the user name, in a directory. Only the CA is allowed to write in this directory, but all users of the CA can read the information in the directory.

16.2.4 Archive

Depending on their use, keys in public-key systems must be stored even after they expire. Public signature keys must be stored as long as signatures generated with those keys must be verified. The CA stores certificates for public signature keys. Private decryption keys must be stored as long as documents were encrypted using those keys must be readable. Those keys are stored in the PSEs of the users. Authentication keys, private signature keys, and public encryption keys need not be put in archives. They must be stored only as long as they are used for authentication, generating signatures, or encrypting documents.

16.2.5 Initialization of the PSE

After Bob has been registered and his keys have been generated and certified, the CA transmits private keys to his CA, if they have been generated by the CA. The CA may also write its own public key and Bob's certificate to the PSE.

16.2.6 Directory service

The CA maintains a *directory* of all certificates together with the name of the owner of each certificate. If Alice wants to know Bob's public keys, she asks her CA whether Bob is one of its users. If Bob is registered with Alice's CA, then Alice obtains Bob's certificate from her CA's directory. Using the public key of her CA, Alice verifies that the certificate was in fact generated by her CA. She obtains the certified public keys of Bob. If Bob is not a user of Alice's CA, then Alice can obtain his public keys from another CA. This is explained below.

Alice may keep in her CA certificates that she frequently uses. However, she must check regularly whether those certificates are still valid.

If a CA has many users, access to its directory may become very slow. It is then possible to keep several copies of the directory and to associate each user with exactly one copy.

Example 16.2.2
An international company wants to introduce a PKI for its 50,000 employees in five countries. The company only wants to maintain one CA. In order to make access to its directory more efficient, the CA distributes five copies of its directory to the five countries. Those copies are updated twice a day.

16.2.7 Key update

All keys in a public-key system have a certain period of validity. Before a key expires, it must be replaced by a new, valid key. This new key is exchanged between the CA and the users in such a way that it does not become insecure even if the old, invalid key becomes known.

The following key update method is insecure. Shortly before Bob's key pair becomes insecure, Bob's CA generates a new key pair. It encrypts the new private key using Bob's old public key and sends it to Bob. Bob decrypts that key using his old private key and replaces the old private key with the new one. If the attacker, Oscar, finds the old private key of Bob, then he can decrypt the message of the CA to

Bob that contains the new private key. Thus, he can find Bob's new private key if he knows the old one. The security of the new private key depends on the security of the old one. This makes no sense. Instead, variants of the Diffie-Hellman key-exchange protocol can be used that avoid the man-in-the-middle attack.

16.2.8 Revocation of certificates

Under certain conditions, a certificate must be invalidated although it is not yet expired.

Example 16.2.3
On a boat trip, Bob has lost his smart card. It contains Bob's private signature key, which he can no longer use for signatures since this private key is nowhere but on the smart card. Therefore, Bob's certificate is no longer valid since it contains the corresponding verification key. The CA must invalidate this certificate.

The CA collects the invalid certificates in the *certificate revocation list* (CRL). It is part of the directory of the CA. An entry in the CRL contains the serial number of the certificate, the date when the certificate was invalidated, and possibly further information, such as the reason for the invalidation. This entry is signed by the CA.

16.2.9 Access to expired keys

Expired keys are kept in the CA's archive and can be provided by the CA upon request.

Example 16.2.4
The CA changes the signature keys of its users each month. Bob orders a new car from Alice and signs this order. But three months later, Bob denies that he ordered the car. Alice wants to prove that the order was actually signed by Bob. She requests Bob's old public verification key from the CA. This key is kept in the archive since it is out of date.

16.3 Certificate Chains

If Bob and Alice do not belong to the same CA, then Alice cannot obtain the public key of Bob from the directory of her own CA but can obtain Bob's public key indirectly.

Example 16.3.1
Alice is registered with a CA in Germany. Bob is registered with a CA in the U.S. Hence, Alice knows the public key of her German CA but not the public key of Bob's CA. Now Alice obtains a certificate for the public key of Bob's CA from her own CA. She also obtains Bob's certificate either directly from Bob or from his CA. Using the public key of Bob's CA, which, in turn, is certified by her own CA, Alice can verify that she obtained a valid certificate for Bob.

As described in Example 16.3.1, Alice can use a *certificate chain* to obtain Bob's authentic public key, even if Bob and Alice belong to different CAs. Formally, such a chain can be described as follows. For a certification authority CA and a name U, denote by CA$\{U\}$ the certificate that certifies the public key of U. Here, U can either be the name of a user or the name of another certification authority. A certificate chain that for Alice certifies the public key of Bob is a sequence

$$\mathrm{CA}_1\{\mathrm{CA}_2\}, \mathrm{CA}_2\{\mathrm{CA}_3\}, \ldots, \mathrm{CA}_{k-1}\{\mathrm{CA}_k\}, \mathrm{CA}_k\{\mathrm{Bob}\}.$$

In this sequence, CA_1 is the CA where Alice is registered. Alice uses the public key of CA_1 to verify the public key of CA_2, she uses the public key of CA_2 to obtain the authentic public key of CA_3, and so on, until she finally uses the public key of CA_k to verify the certificate of Bob.

This method only works if trust is transitive (i.e., if U_1 trusts U_2 and U_2 trusts U_3, then U_1 trusts U_3).

Solutions of the Exercises

Exercise 1.12.1
Let $z = \lfloor \alpha \rfloor = \max\{x \in \mathbb{Z} : x \leq \alpha\}$. Then $\alpha - z \geq 0$. Moreover, $\alpha - z < 1$, since $\alpha - z \geq 1$ implies $\alpha - (z+1) \geq 0$, which contradicts the maximality of α. Therefore, $0 \leq \alpha - z < 1$ or $\alpha - 1 < z \leq \alpha$. But there is only one integer in this interval, so, z is uniquely determined.

Exercise 1.12.3
The divisors of 195 are ± 1, ± 3, ± 5, ± 13, ± 15, ± 39, ± 65, ± 195.

Exercise 1.12.5
$1243 \bmod 45 = 28$, $-1243 \bmod 45 = 17$.

Exercise 1.12.7
Suppose that m divides the difference $b - a$. Let $a = q_a m + r_a$ with $0 \leq r_a < m$ and let $b = q_b m + r_b$ with $0 \leq r_b < m$. Then $r_a = a \bmod m$ and $r_b = b \bmod m$. Moreover,

$$b - a = (q_b - q_a)m + (r_b - r_a). \tag{16.1}$$