Chapter 9

Key Management

There are actually two distinct aspects to the use of public-key encryption in this regard:

- The distribution of public keys.
- The use of public-key encryption to distribute secret keys.

9.1 Distribution of Public Keys

Several techniques have been proposed for the distribution of public keys. Virtually all of these proposals can be grouped into the following general schemes:

A. Public announcement

B. Publicly available directory

C. Public-key authority

D. Public-key certificates

9.1.1 Public Announcement of Public Keys

On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large (figure 9.1). Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key.

9.1.2 Publicly Available Directory

A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organisation (figure 9.2). Such a scheme would include the following elements:

A
B
KU_a
KU_a
KU_a
KU_a
KU_b
KU_b
KU_b
KU_b

Figure 9.1: Uncontrolled Public Key.

- The authority maintains a directory with a {name, public key} entry for each participant.
- Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
- A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
- Periodically, the authority publishes the entire directory or updates to the directory. For example, a hard-copy version much like a telephone book could be published, or updates could be listed in a widely circulated newspaper.
- Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

This scheme is clearly more secure than individual public announcements, but still has vulnerabilities. If an opponent succeeds in obtaining or computing the private key of the directory authority, the opponent could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Another way to achieve the same end is for the opponent to tamper with the records kept by the authority.

9.1.3 Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. As before, the scenario

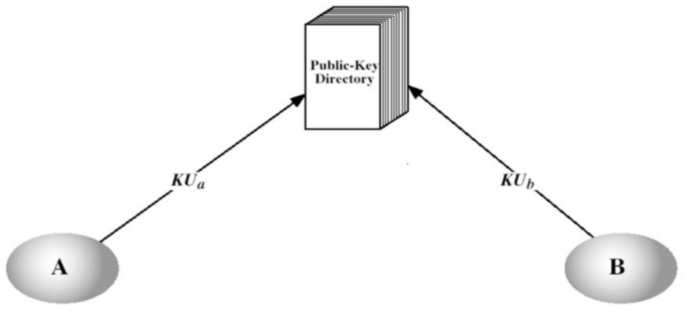

Figure 9.2: Public-Key Publication.

assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. However this isn't perfect as the public-key authority could be somewhat of a bottleneck in the system. The reason for this is that a user must appeal to the authority for a public key for every other user that it wishes to contact. Also the directory of names and public keys maintained by the authority is vulnerable to tampering.

9.1.4 Public-Key Certificates

An alternative approach to the above is the use of **certificates** that can be used by participants to exchange keys without contacting a public-key authority. Each certificate, containing a public key and other information, is created by a certificate authority and is given to the participant with the matching private key. A participant conveys its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. Four requirements can be placed on this particular scheme:

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.

An example of this scheme can be seen using the following transaction:

$$C_A = E_{\mathrm{KR}_{auth}}[T, \mathrm{ID}_A, KU_A]$$

where C_A is A's certificate, KR$_{auth}$ is the private key of the certificate authority, ID$_A$ is A's identification and KU$_A$ is A's public key.

A can then pass C_A to any participant who reads and verifies it as follows:

$$D_{KU_{auth}}[C_A] = D_{KU_{auth}}[E_{KU_{auth}}[T, ID_A, KU_A]] = (T, ID_A, KU_A)$$

An example of a certification service is the X.509 scheme and is described now. The Secure Socket Layer (SSL) protocol used to protect web based transactions (to be seen later in the course) uses this service.

9.1.4.1 X.509 Authentication service

The ITU-T recommendation X.509 Directory Authentication Service is part of the X.500 series of recommendations that define a directory service. The directory is, in effect, a server or distributed set of servers that maintains a database of information about users.

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of Public Certificates. Each certificate contains the public key of a user and is signed with the private key of a Trusted Certificate Authority (CA). X.509 is based on the use of public key cryptography and digital signatures (e.g. RSA and different hash functions such as MD5 etc.).

X.509 was initially issued in 1988. The standard was subsequently revised to address some of the security concerns; a revised recommendation was issued in 1993. A third version was issued in 1995 and revised in 2000.

The user certificates which are at the heart of X.509 are assumed to be created by some trusted Certificate Authority (CA) and placed in the directory by the CA or by the user. The Directory Server itself is not responsible for the creation of the public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

User certificates generated by CA have following characteristics:

- Any user with access to the public key of the CA can recover the user public key that was certified.
- No party other than the CA can modify the directory

An outline of the X.509 is shown in figure 9.3 and includes the following elements:

- Version: Differentiates among successive versions of the certificate format;teh default is version 1.If the Issuer Unique Identifier or Subject Unique Identifier are present, the value must be version 2. If one or more extensions are present, the version must be version3.

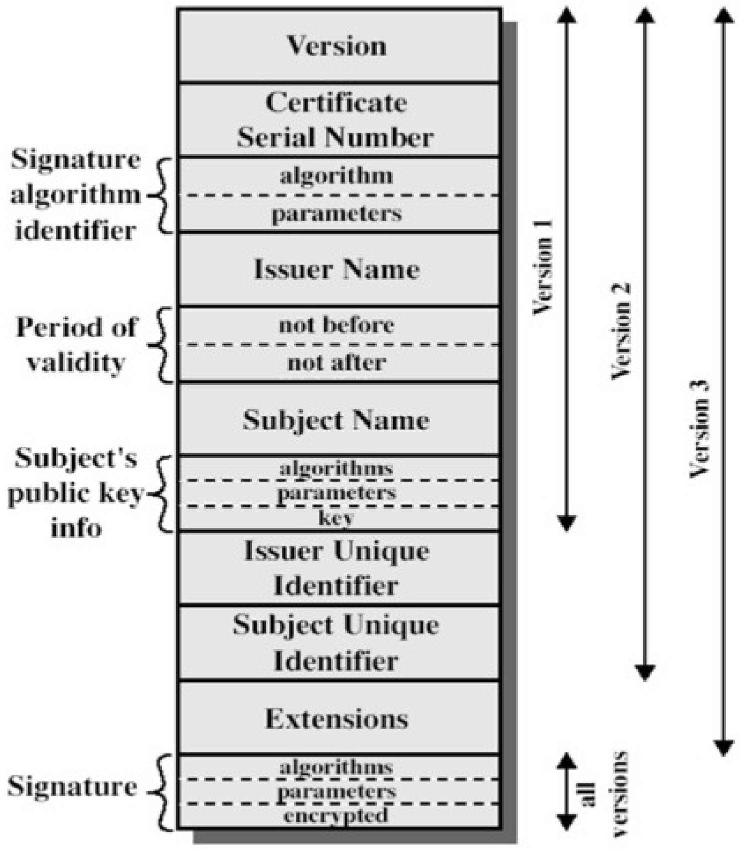

Figure 9.3: The X.509 certificate.

- Serial Number: An integer value, unique within the issuing CA, that is unambiguously associated with this certificate;
- Signature: The algorithm used to sign the certificate, together with any associated parameters; this has little utility due to a repeat of this information in the Signature field.
- Issuer name: X.500 name of the CA that created and signed this certificate;
- Period of validity: Two dates, the first and last on which the certificate is valid;
- Subject name: The name of the user to whom the certificate refers;
- Subject's public key information: The public key of the subject plus an identifier of the algorithm for which the key is to be used together with any associated parameters;
- Issuer unique identifier: An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities;

- Subject's unique identifier: An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities;
- Extensions: A set of one or more extension fields; these were added in version 3.
- Signature: Covers all of the other fields of the certificate. It contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifier.

The standard uses the following notation to define a certificate:

$$CA << A >>= CA\{V, SN, AI, CA, T_A, A, Ap\}$$

where $Y << X >>$ is the certificate of user X issued by certification authority Y and $Y\{I\}$ is the signing of I by Y. It consists of I with an encrypted hash code appended.

As was mentioned there have been three versions of the X.509 to date. The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed. Five requirements not satisfied by version two are:

1. The Subject field is inadequate to convey the identity of a key owner to a public-key user. X.509 names may be relatively short and lacking in obvious identification details that may be needed by the user.
2. The subject field is also inadequate for many applications, which typically recognise entities by an Internet e-mail address, a URL, or some other Internet related identification.
3. There is a need to indicate security policy information. This enables a security application or function, such as IPSec, to relate an X.509 certificate to a given policy.
4. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.
5. It is important to be able to identify separately different keys used by the same owner at different times. This feature supports key life cycle management, in particular the ability to update key pairs for users and CAs on a regular basis or under exceptional circumstances.

9.2 Public-Key Distribution of Secret Keys

Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering , or both is possible. However, few users will wish to make exclusive use of public-key encryption for communications because of the relatively slow data rates that can be achieved. Accordingly, public-key encryption is more reasonably viewed as a vehicle for the distribution of secret keys to be used for conventional encryption.

9.2.1 Simple Secret-Key Distribution

An extremely simple scheme put forward by Ralph Merkle is illustrated in figure 9.4. If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair $\{KU_A, KR_A\}$ and transmits a message to B consisting of KU_A, and an identifier of A, ID$_A$.
2. B generates a secret key, K_s, and transmits it to A, encrypted with A's public key.
3. A computes $D_{KR_A}[E_{KU_A}[K_s]]$ to recover the secret key. Since only A can decrypt the message, only A and B will know the identity of K_s.
4. A discards KU_A and KR_A and B discards KU_A.

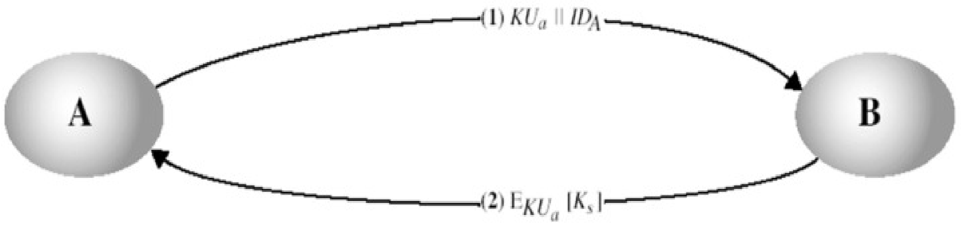

Figure 9.4: Simple Use of Public-Key Encryption to Establish a Session Key.

A and B can now securely communicate encryption and the session keys K_s. At the completion of the exchange, both A and B discard K_s. Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication. Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping. The protocol is vulnerable to an active attack. If an opponent, E, has control of the intervening communications channel, then he can compromise the communications in the following way without being detected:

1. A generates a public/private key pair $\{KU_A, KR_A\}$ and transmits a message intended for B consisting of KU_A and identifier of A, ID$_A$.
2. E intercepts the message, creates its own public/private key pair $\{KU_E, KR_E\}$ and transmits $KU_E||$ID$_A$ to B.
3. B generates a secret key, K_s, and transmits $E_{KU_E}[K_s]$.
4. E intercepts the message, and learns K_s by computing $D_{KR_E}[E_{KU_E}[K_s]]$.
5. E transmits $E_{KU_A}[K_s]$ to A.

The result is that both A and B know K_s and are unaware that K_s has also been revealed to E. A and B can now exchange messages using K_s. E no longer actively interferes with the communications channel but simply eavesdrops. Knowing, K_s, E can decrypt all messages, and both A and B are unaware of the problem. Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

9.3 Secret Key Distribution with Confidentiality and Authentication

Figure 9.5 provides protection against both active and passive attacks. For this example it is assumed that A and B have exchanged public keys by one of the schemes described earlier in this section. Then the following steps occur:

1. A uses B's public key to encrypt a message to B containing an identifier of A (ID$_A$) and a nonce (N_1), which is used to uniquely identify this transaction.
2. B sends a message to A encrypted with KU_A and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Since only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.
3. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_s and sends $M = E_{KU_b}[E_{KR_A}[K_s]]$ to B. Encryption of this message with B's public key ensures that only B can read it, encryption with A's private key ensures that only A could have sent it.
5. B computes $D_{KU_A}[D_{KR_b}[M]]$ to recover the secret key.

This scheme ensures both confidentiality and authentication in the exchange of a secret key.

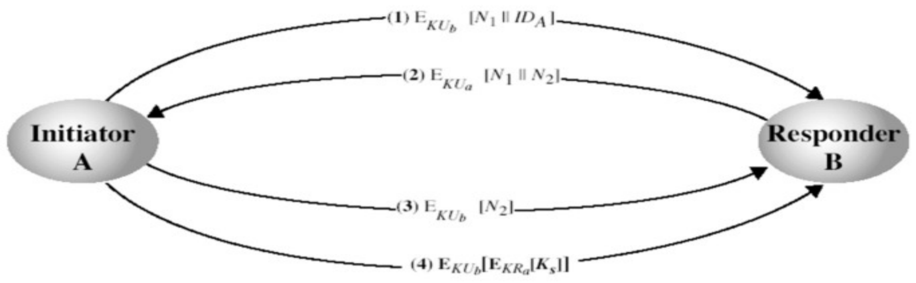
(1) $E_{KU_b}[N_1 \| ID_A]$
(2) $E_{KU_a}[N_1 \| N_2]$
Initiator A
Responder B
(3) $E_{KU_b}[N_2]$
(4) $E_{KU_b}[E_{KR_a}[K_s]]$

Figure 9.5: Public-Key Distribution of Secret Keys.

9.4 A Hybrid Scheme

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes. This scheme retains the use of a key distribution centre (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key. A public key scheme is used to distribute the master keys. The following rationale is provided for using this three level approach.

- **Performance:** There are many applications, especially transaction-oriented applications in which the session keys change frequently. Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption. With a three-level hierarchy, public-key encryption is used only occasionally to update the master key between a user and the KDC.

- **Backward Compatibility:** The hybrid scheme is easily overlaid on an existing KDC scheme, with minimal disruption of software changes.

The addition of a public-key layer provides a secure, efficient means of distributing master keys. This is an advantage in a configuration in which a single KDC serves a widely distributed set of users.