

Chapter 4: Key Management and Distribution

4.1 SYMMETRIC KEY DISTRIBUTION USING SYMMETRIC ENCRYPTION

The strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key. For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B. (The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used. Communication between end systems is encrypted using a temporary key, often referred to as a **session key**. Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded. Each session key is obtained from the key distribution center over the same networking facilities used for end-user communication. Accordingly, session keys are transmitted in encrypted form, using a **master key** that is shared by the key distribution center and an end system or user.

A Key Distribution Scenario

The scenario assumes that each user shares a unique master key with the key distribution center (KDC). Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection. A has a master key, K_a , known only to itself and the KDC; similarly, B shares the master key K_b with the KDC. The following steps occur.

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1 , for this transaction, which we refer to as a nonce. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.
2. The KDC responds with a message encrypted using K_a . Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:
 - The one-time session key, K_s , to be used for the session

- The original request message, including the nonce, to enable A to match this response with the appropriate request

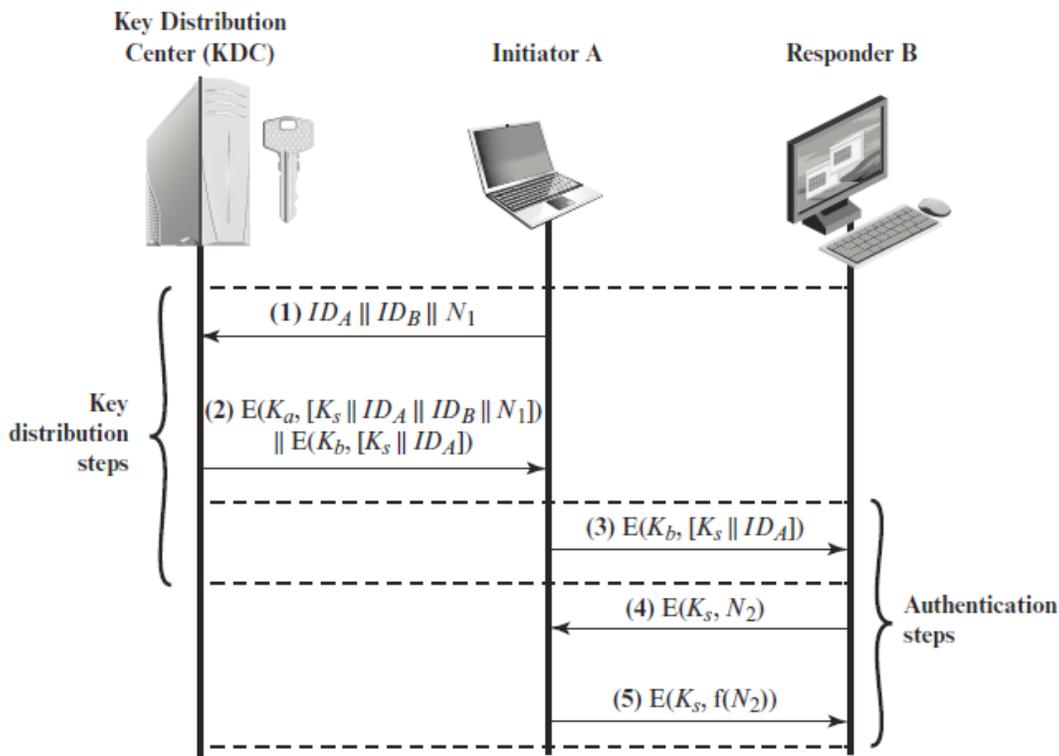


Figure 4.1: Key Distribution Scenario

Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.

In addition, the message includes two items intended for B:

- The one-time session key, K_s , to be used for the session
- An identifier of A (e.g., its network address), ID_A

These last two items are encrypted with K_b (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, $E(K_b, [K_s \parallel ID_A])$. Because this information is encrypted with K_b , it is protected from eavesdropping. B now knows the session key (K_s), knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b).

At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N_2 , to A.

- Also, using K_s , A responds with $f(N_2)$, where f is a function that performs some transformation on N_2 (e.g., adding one).

These steps assure B that the original message it received (step 3) was not a replay. Note that the actual key distribution involves only steps 1 through 3, but that steps 4 and 5, as well as step 3, perform an authentication function.

It is not necessary to limit the key distribution function to a single KDC. Indeed, for very large networks, it may not be practical to do so. As an alternative, a hierarchy of KDCs can be established.

Decentralized Key Control

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized. Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.

A session key may be established with the following sequence of steps (Figure 4.2)

- A issues a request to B for a session key and includes a nonce, N_1 .
- B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value $f(N_1)$, and another nonce, N_2 .
- Using the new session key, A returns $f(N_2)$ to B.

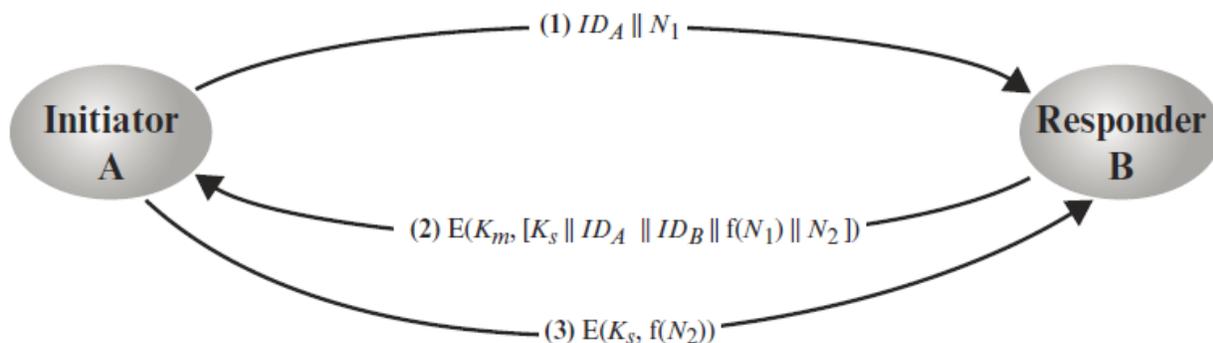


Figure 4.2: Decentralized Key Distribution

4.2 SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION

One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution.

Simple Secret Key Distribution

As illustrated in Figure 4.3. If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of PU_a and an identifier of A, ID_A .
2. B generates a secret key, K_s , and transmits it to A, which is encrypted with A's public key.
3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s .
4. A discards PU_a and PR_a and B discards PU_a .

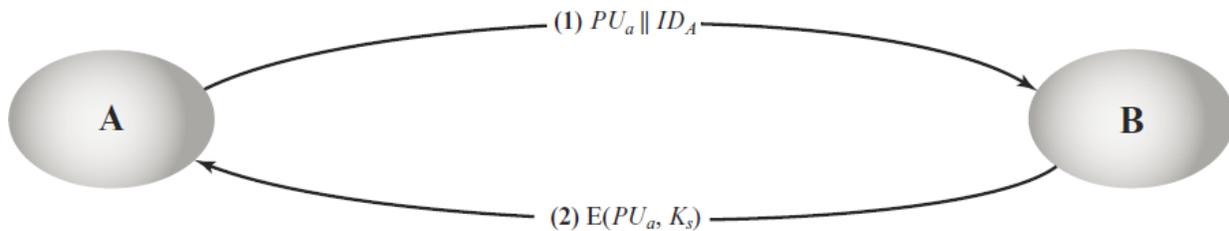


Figure 4.3: Simple Use of Public-Key Encryption to Establish a Session Key

The protocol depicted is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message. Such an attack is known as a man-in-the-middle attack.

In the present case, if an adversary, D, has control of the intervening communication channel, then D can compromise the communication in the following fashion without being detected (Figure 4.4).

1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message intended for B consisting of PU_a and an identifier of A, ID_A .
2. D intercepts the message, creates its own public/private key pair $\{PU_d, PR_d\}$ and transmits $PU_d || ID_A$ to B.
3. B generates a secret key, K_s , and transmits $E(PU_d, K_s)$.
4. D intercepts the message and learns K_s by computing $D(PR_d, E(PU_d, K_s))$.
5. D transmits $E(PU_a, K_s)$ to A.

The result is that both A and B know K_s and are unaware that K_s has also been revealed to D. A and B can now exchange messages using K_s . D no longer actively interferes with the communications channel but simply eavesdrops. Knowing K_s , D 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.

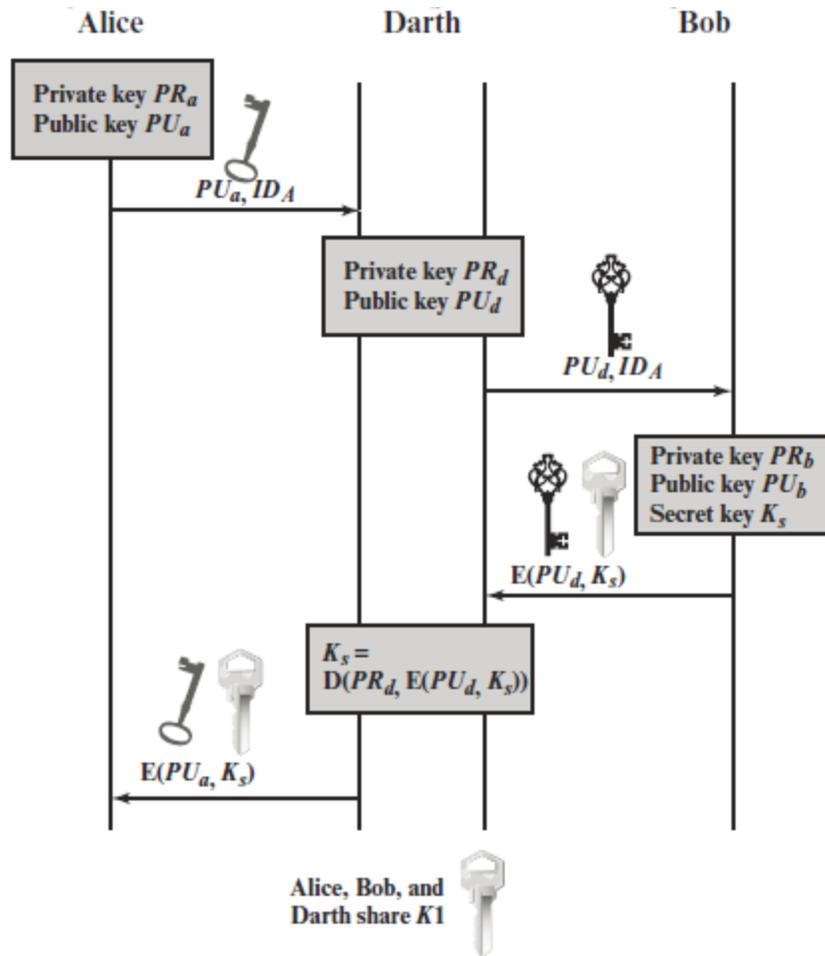


Figure 4.4: Another Man-in-the-Middle Attack

Secret Key Distribution with Confidentiality and Authentication

Figure 4.5, provides protection against both active and passive attacks. We begin at a point when it is assumed that A and B have exchanged public keys by one of the schemes described subsequently in this chapter. 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 ($N1$), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PU_a and containing A's nonce ($N1$) as well as a new nonce generated by B ($N2$). Because only B could have decrypted message (1), the presence of $N1$ in message (2) assures A that the correspondent is B.
3. A returns $N2$, encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_s and sends $M = E(PU_b, E(PR_a, K_s))$ 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 $D(PU_a, D(PR_b, M))$ to recover the secret key.

The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

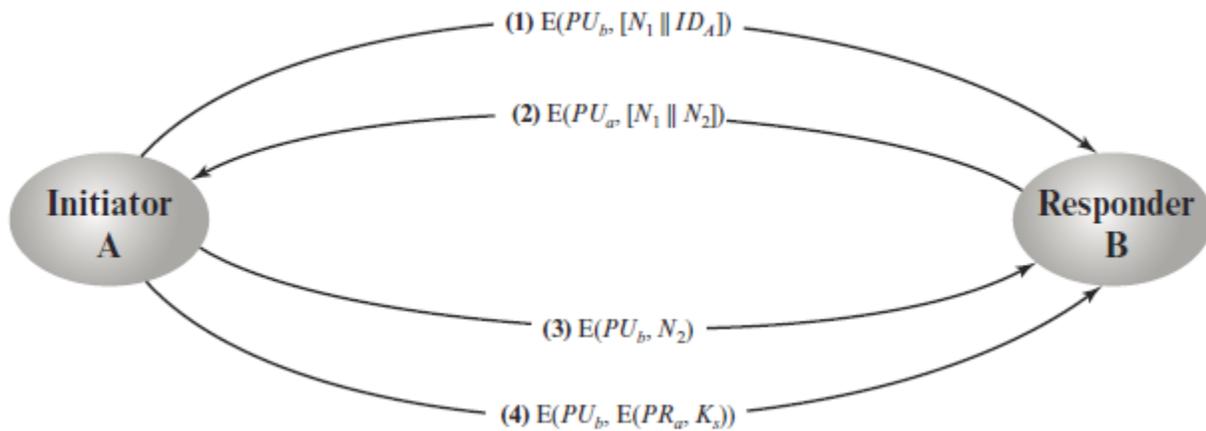


Figure 4.5: Public-Key Distribution of Secret Keys

4.3 DISTRIBUTION OF PUBLIC KEYS

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

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 4.6).



Figure 4.6: Uncontrolled Public-Key Distribution

Although this approach is convenient, it has a major weakness. Anyone can forge such a public key 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. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

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 organization (Figure 4.7). Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. 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.
3. 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.
4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

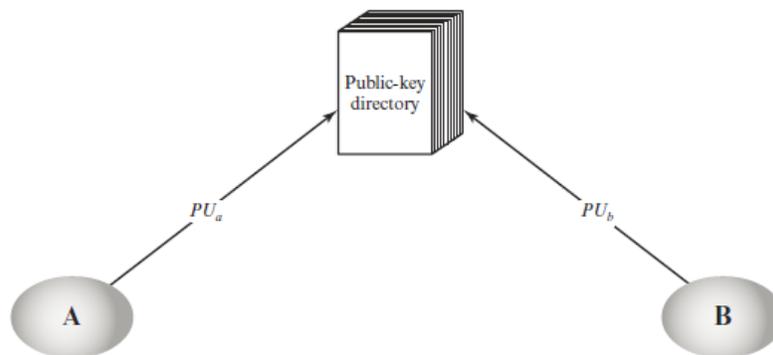


Figure 4.7: Public-Key Publication

This scheme is clearly more secure than individual public announcements but still has vulnerabilities. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary 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 adversary to tamper with the records kept by the authority.

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. A typical scenario is illustrated in Figure 4.8. As before, the scenario 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. The following steps

(matched by number to Figure 4.8) occur.

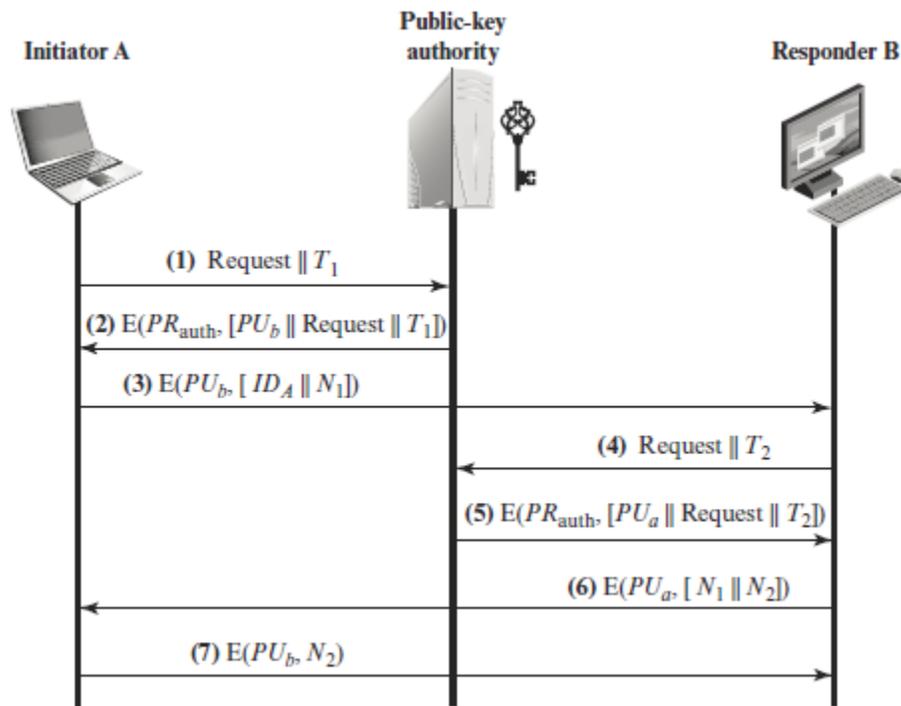


Figure 4.8: Public-Key Distribution Scenario

1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.
2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth} . Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:
 - B's public key, PU_b , which A can use to encrypt messages destined for B
 - The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
 - The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key
3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
4. 5. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

6. sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new

nonce generated by B (N2). Because only B could have decrypted message (3), the presence of N1 in message (6) assures A that the correspondent is B.

7. A returns N2, which is encrypted using B's public key, to assure B that its correspondent is A.

Thus, a total of seven messages are required. However, the initial five messages need be used only infrequently because both A and B can save the other's public key for future use—a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

Public-Key Certificates

The scenario of Figure 4.8 is attractive, yet it has some drawbacks. The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

An alternative approach, is to use certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party. Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community. A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. We can place the following requirements on this 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 time validity of the certificate.

A certificate scheme is illustrated in Figure 4.9. Each participant applies to the certificate authority, supplying a public key and requesting a certificate. Application must be in person or by some form of secure authenticated communication. For participant A, the authority provides a certificate of the form

$$C_A = E(PR_{auth}[T || ID_A || PU_a])$$

where PR_{auth} is the private key used by the authority and T is a timestamp. A may then pass this

certificate on to any other participant.

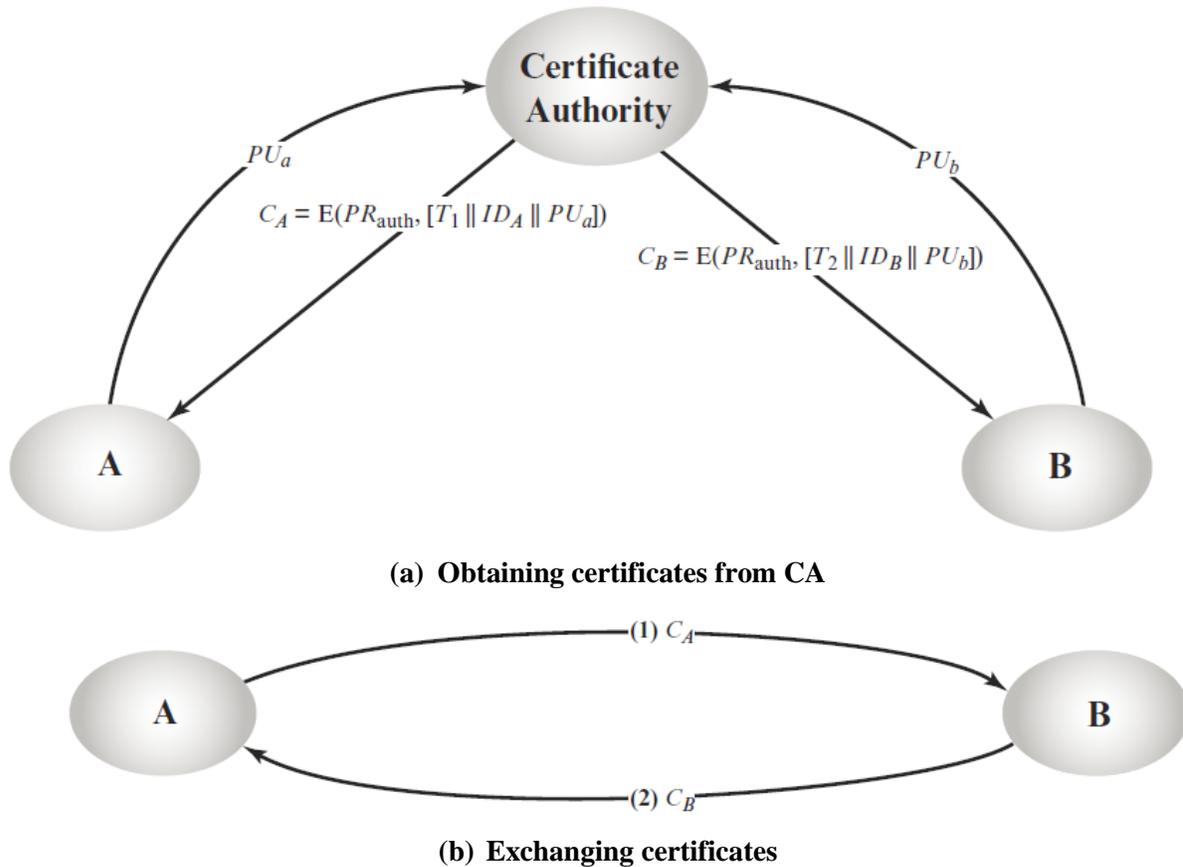


Figure 4.9: Exchange of Public-Key Certificates

One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP security, transport layer security (TLS), and S/MIME.

4.4 X.509 CERTIFICATES

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific digital signature algorithm nor a specific hash function. Figure 4.10 illustrates the overall X.509 scheme for generation of a public-key certificate. The certificate for Bob's public key includes unique identifying information for Bob, Bob's public key, and identifying information about the CA, plus other information as explained subsequently. This information is then signed by computing a hash value of the information and generating a digital signature using the hash value and the CA's private key. X.509 indicates that the signature is formed by encrypting the hash value.

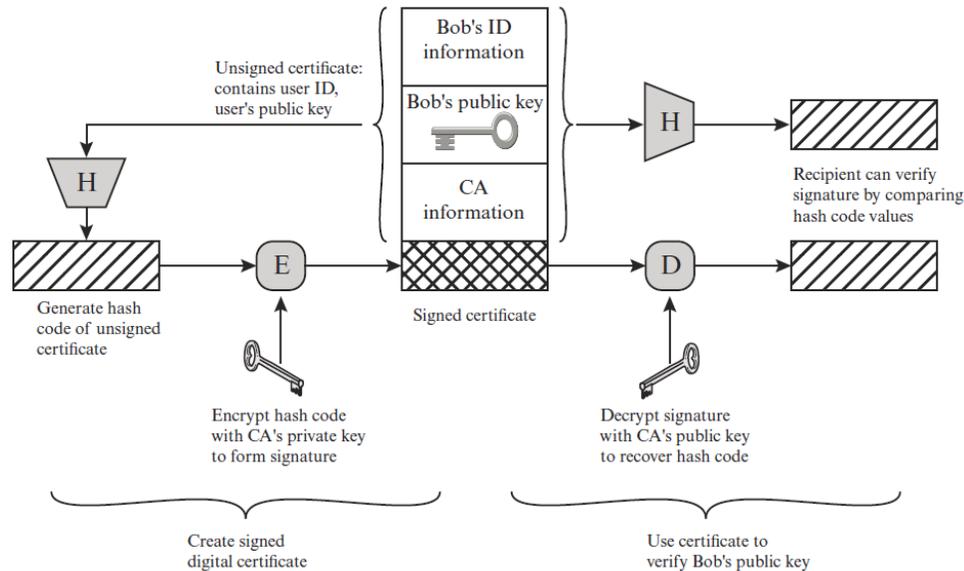


Figure 4.10: X.509 Public-Key Certificate Use

Certificates

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification 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 public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

Figure 4.11a shows the general format of a certificate, which includes the following elements.

- **Version:** Differentiates among successive versions of the certificate format; the 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 version 3. Although the X.509 specification is currently at version 7, no changes have been made to the fields that make up the certificate since version 3.
- **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate.
- **Signature algorithm identifier:** The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.
- **Issuer name:** X.500 name of the CA that created and signed this certificate.
- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
- **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.

- Subject’s public-key information: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

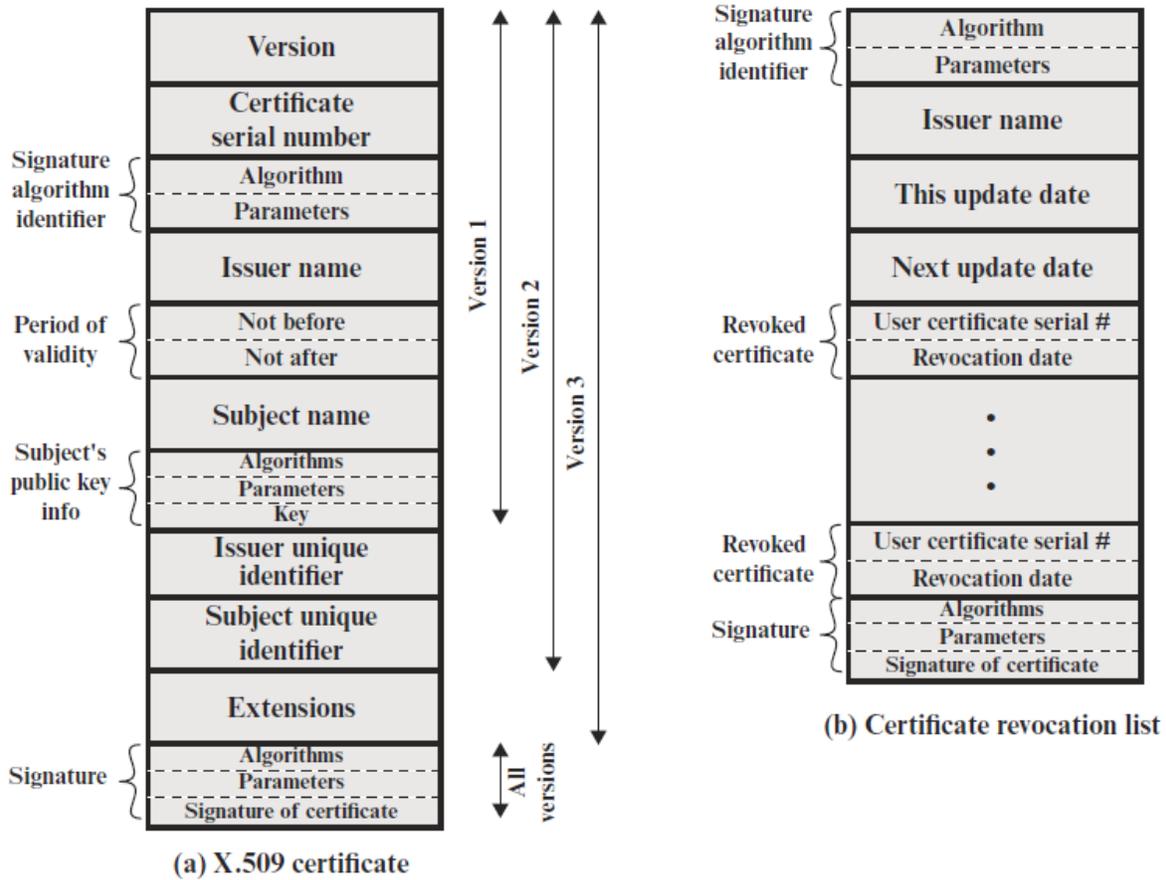


Figure 4.11: X.509 Formats

- 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 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. Extensions were added in version 3 and are discussed later in this section.
- Signature: Covers all of the other fields of the certificate. One component of this field is the digital signature applied to the other fields of the certificate. This field includes the signature algorithm identifier.

The unique identifier fields were added in version 2 to handle the possible reuse of subject and/or issuer names over time. These fields are rarely used.