

This chapter continues our overview of public-key encryption. We examine key distribution and management for public-key systems, including a discussion of Diffie-Hellman key exchange. Finally, we provide an introduction to elliptic curve cryptography.

10.1 KEY MANAGEMENT

In Chapter 7, we examined the problem of the distribution of secret keys. One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard:

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

We examine each of these areas in turn.

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
- Publicly available directory
- Public-key authority
- Public-key certificates

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 10.1). For example, because of the growing popularity of PGP (pretty good privacy, discussed in Chapter 15), which makes use of RSA, many PGP users have adopted the practice of appending their public



Figure 10.1 Uncontrolled Public-Key Distribution

key to messages that they send to public forums, such as USENET newsgroups and Internet mailing lists.

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. 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 (see Figure 9.3).

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 10.2). 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. 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.
5. 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

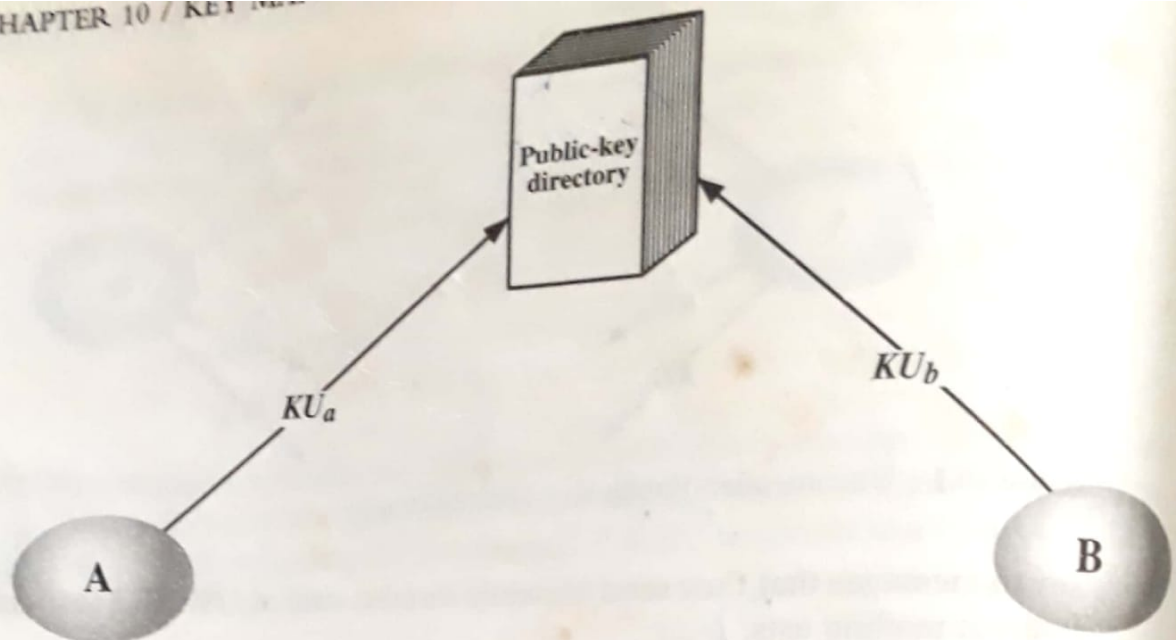


Figure 10.2 Public-Key Publication

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.

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 10.3, which is based on a figure in [POPE79]. 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 10.3) occur:

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, KR_{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, KU_b , which A can use to encrypt messages destined for B
 - The original request, 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, 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. 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. 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). Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.
7. A returns N_2 , 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 four 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 10.3 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.

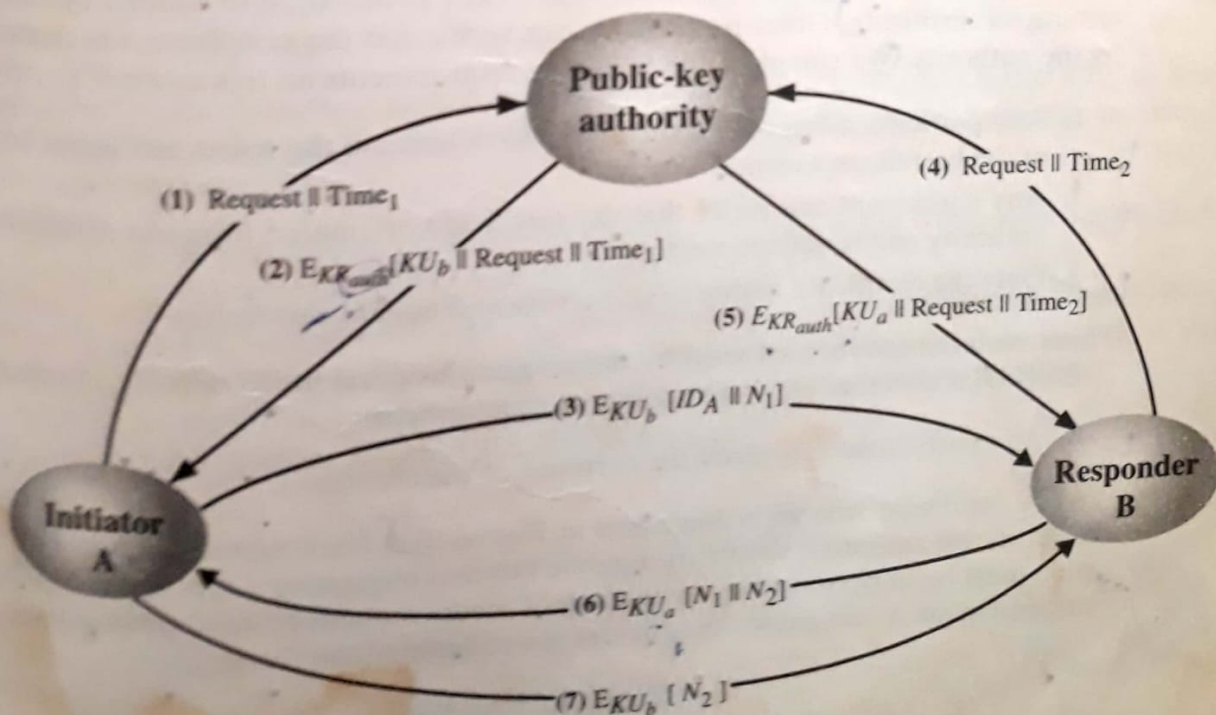


Figure 10.3 Public-Key Distribution Scenario

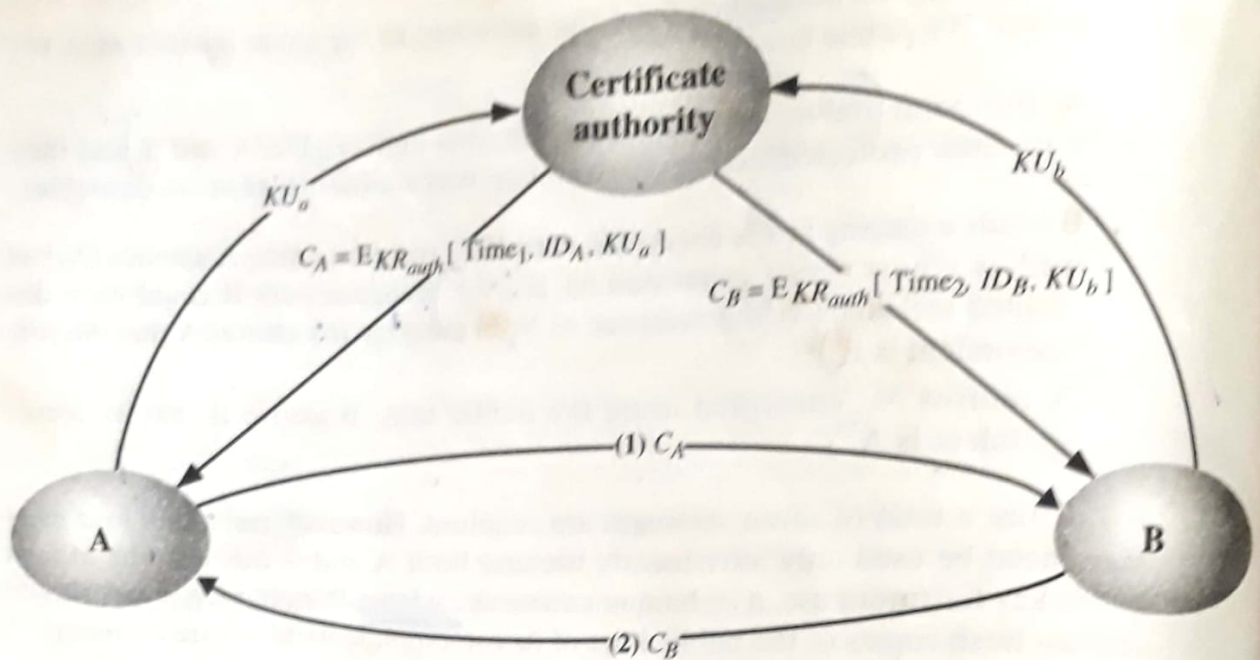


Figure 10.4 Exchange of Public-Key Certificates

An alternative approach, first suggested by Kohnfelder [KOH78], 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. Each certificate contains 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. 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.

These requirements are satisfied by the original proposal in [KOH78]. Denning [DENN83] added the following additional requirement:

4. Any participant can verify the currency of the certificate.

A certificate scheme is illustrated in Figure 10.4. 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

$$C_A = E_{KR_{auth}}[T, ID_A, KU_a]$$

where KR_{auth} is the private key used by the authority. A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

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

The recipient uses the authority's public key, KU_{auth} , to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements ID_A and KU_A provide the recipient with the name and public key of the certificate's holder. The timestamp T validates the currency of the certificate. The timestamp counters the following scenario. A's private key is learned by an opponent. A generates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the opponent replays the old certificate to B. If B then encrypts messages using the compromised old public key, the opponent can read those messages.

In this context, the compromise of a private key is comparable to the loss of a credit card. The owner cancels the credit card number but is at risk until all possible communicants are aware that the old credit card is obsolete. Thus, the timestamp serves as something like an expiration date. If a certificate is sufficiently old, it is assumed to be expired.

Public-Key Distribution of Secret Keys

Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping (Figure 9.2), tampering (Figure 9.3), or both (Figure 9.4) is possible. However, few users will wish to make exclusive use of public-key encryption for communication 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.

Simple Secret Key Distribution

An extremely simple scheme was put forward by Merkle [MERK79], as illustrated in Figure 10.5. 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. Because 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 .

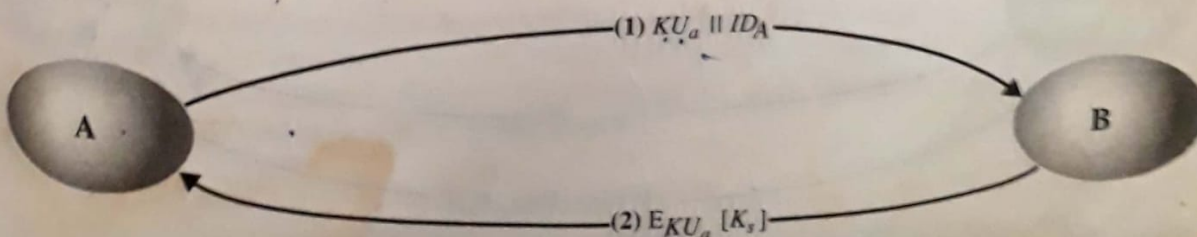


Figure 10.5 Simple Use of Public-Key Encryption to Establish a Session Key

A and B can now securely communicate using conventional encryption and the session key 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.

This protocol is vulnerable to an active attack. If an opponent, E, has control of the intervening communication channel, then E can compromise the communication in the following fashion 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 an identifier of A, ID_A .
2. E intercepts the message, creates its own public/private key pair $\{KU_e, KR_e\}$ and transmits $KU_e \parallel ID_A$ 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.

Secret Key Distribution with Confidentiality and Authentication

Figure 10.6, based on an approach suggested in [NEED78], 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 earlier in this section. Then the following steps occur:

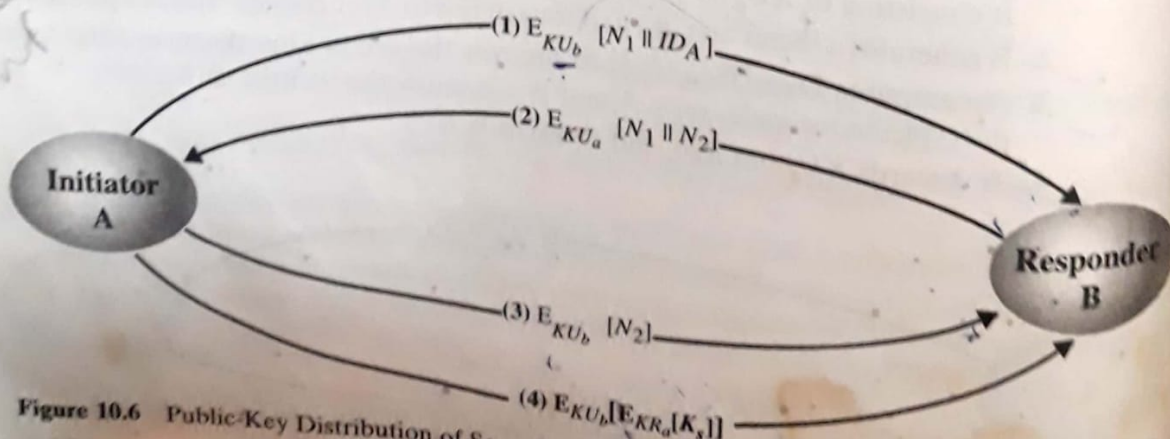


Figure 10.6 Public Key Distribution of Secret Keys

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 identify this transaction uniquely.
2. B sends a message to A encrypted with K_{U_a} and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because 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 $M = E_{K_{U_b}}[E_{K_{R_a}}[K_s]]$ 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 $D_{K_{U_a}}[D_{K_{R_b}}[M]]$ to recover the secret key.

Notice that the first three steps of this scheme are the same as the last three steps of Figure 10.3. The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

A Hybrid Scheme

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes [LE93]. This scheme retains the use of a key distribution center (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 or 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.

10.2 DIFFIE-HELLMAN KEY EXCHANGE

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [DIFF76b] and is generally referred to as Diffie-Hellman key exchange.¹ A number of commercial products employ this key exchange technique.

¹Williamson of Britain's CESG published the identical scheme a few months earlier in a classified document [WILL76] and claims to have discovered it several years prior to that; see [ELLI99] for a discussion.

The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of the keys.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way. First, we define a primitive root of a prime number p as one whose powers generate all the integers from 1 to $p - 1$. That is, if a is a primitive root of the prime number p , then the numbers

$$a \bmod p, a^2 \bmod p, \dots, a^{p-1} \bmod p$$

are distinct and consist of the integers from 1 through $p - 1$ in some permutation.

For any integer b and a primitive root a of prime number p , we can find a unique exponent i such that

$$b = a^i \bmod p \quad \text{where } 0 \leq i \leq (p - 1)$$

The exponent i is referred to as the discrete logarithm, or index, of b for the base a , $\bmod p$. This value is denoted as $\text{ind}_{a,p}(b)$. See Chapter 8 for an extended discussion of discrete logarithms.

With this background we can define the Diffie-Hellman key exchange, which is summarized in Figure 10.7. For this scheme there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q . Suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$. Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and user B computes the key as $K = (Y_A)^{X_B} \bmod q$. These two calculations produce identical results:

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= \alpha^{X_B X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned}$$

by the rules of modular arithmetic

The result is that the two sides have exchanged a secret key. Furthermore, because X_A and X_B are private, an opponent only has the following ingredients to work with: q , α , Y_A , and Y_B . Thus, the opponent is forced to take a discrete logarithm to determine the key. For example, attacking the secret key of user B, the opponent must compute

$$X_B = \text{ind}_{\alpha,q}(Y_B)$$

Global Public Elements

q
 α Prime number
 $\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A $X_A < q$
 Calculate public Y_A $Y_A = \alpha^{X_A} \bmod q$ 11

User B Key Generation

Select private X_B $X_B < q$
 Calculate public Y_B $Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A

$$K = (Y_B)^{X_A} \bmod q$$

Generation of Secret Key by User B

$$K = (Y_A)^{X_B} \bmod q$$

Figure 10.7 The Diffie-Hellman Key Exchange Algorithm

The opponent can then calculate the key K in the same manner as user B calculates it. The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, [it is very difficult to calculate discrete logarithms]. For large primes, the latter task is considered infeasible.

Here is an example. Key exchange is based on the use of the prime number $q = 353$ and a primitive root of 353, in this case $\alpha = 3$. A and B select secret keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

$$\text{A computes } Y_A = 3^{97} \bmod 353 = 40.$$

$$\text{B computes } Y_B = 3^{233} \bmod 353 = 248.$$

After they exchange public keys, each can compute the common secret key:

$$\text{A computes } K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160.$$

$$\text{B computes } K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160.$$

We assume an attacker would have available the following information:

$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$

In this simple example, it would be possible by brute force to determine the secret key 160. In particular, an attacker E can determine the common key by discovering a solution to the equation $3^a \bmod 353 = 40$ or the equation $3^b \bmod 353 = 248$. The brute-force approach is to calculate powers of 3 modulo 353, stopping when the result equals either 40 or 248. The desired answer is reached with the exponent value of 97, which provides $3^{97} \bmod 353 = 40$.

With larger numbers, the problem becomes impractical.

Figure 10.8 shows a simple protocol that makes use of the Diffie-Hellman calculation. Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key X_A , calculate Y_A , and send that to user B. User B responds by generating a private value X_B , calculating Y_B , and sending Y_B to user A. Both users can now calculate the key. The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q and α and include those in the first message.

As an example of another use of the Diffie-Hellman algorithm, suppose that a group of users (e.g., all users on a LAN) each generate a long-lasting private value X_A and calculate a public value Y_A . These public values, together with global public values for q and α , are stored in some central directory. At any time, user B can access user A's public value, calculate a secret key, and use that to send an encrypted message to user A. If the central directory is trusted, then this form of communication provides both confidentiality and a degree of authentication. Because only A and B can determine the key, no other user can read the message (confidentiality). Recipient A knows that only user B could have created a message using this key (authentication). However, the technique does not protect against replay attacks.

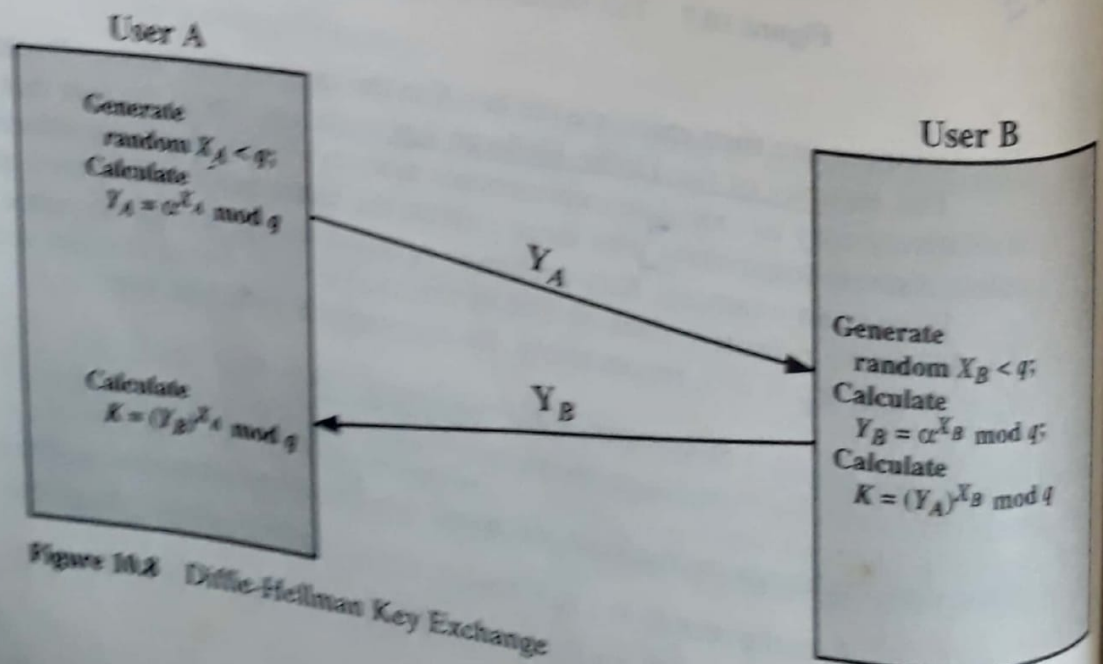


Figure 10.8 Diffie-Hellman Key Exchange