UNIT II

Introduction to Number theory : Integer Arithmetic, Modular Arithmetic, Matrices, Linear Congruence, Algebraic Structures, GF(2n) Fields, Primes, Primarily Testing, Factorization, Chinese remainder Theorem, Quadratic Congruence, Exponentiation and Logarithm.

Public-key cryptography: Principles of public-key cryptography, RSA Algorithm, Diffie-Hellman Key Exchange, ELGamal cryptographic system, Elliptic Curve Arithmetic, Elliptic curve cryptography

---

Principles of public-key cryptography:

Of equal importance to conventional encryption is **public-key encryption**, which finds use in **message authentication** and key **distribution.** This section looks first at the basic concept of **public-key encryption** and takes a preliminary look at **key distribution issues**. This notes examines the two most important public-key algorithms: RSA and Diffie-Hellman.

Public-Key Encryption Structure

Public-key encryption, first publicly proposed by Diffie and Hellman in 1976 [DIFF76], is the first truly revolutionary advance in encryption in literally thousands of years. Public-key algorithms are based on mathematical functions rather than on simple operations on bit patterns, such as are used in symmetric encryption algo-rithms. More important, public-key cryptography is asymmetric, involving the use of two separate keys—in contrast to the symmetric conventional encryption, which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication.

Before proceeding, we should first mention several common misconceptions concerning public-key encryption. One is that public-key encryption is more secure from cryptanalysis than conventional encryption. In fact, the security of any encryption scheme depends on (1) the length of the key and (2) the computational work involved in breaking a cipher. There is nothing in principle about either con-ventional or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis. A second misconception is that public-key encryption is a general-purpose technique that has made conventional encryption obsolete. On the contrary, because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that conven-tional encryption will be abandoned

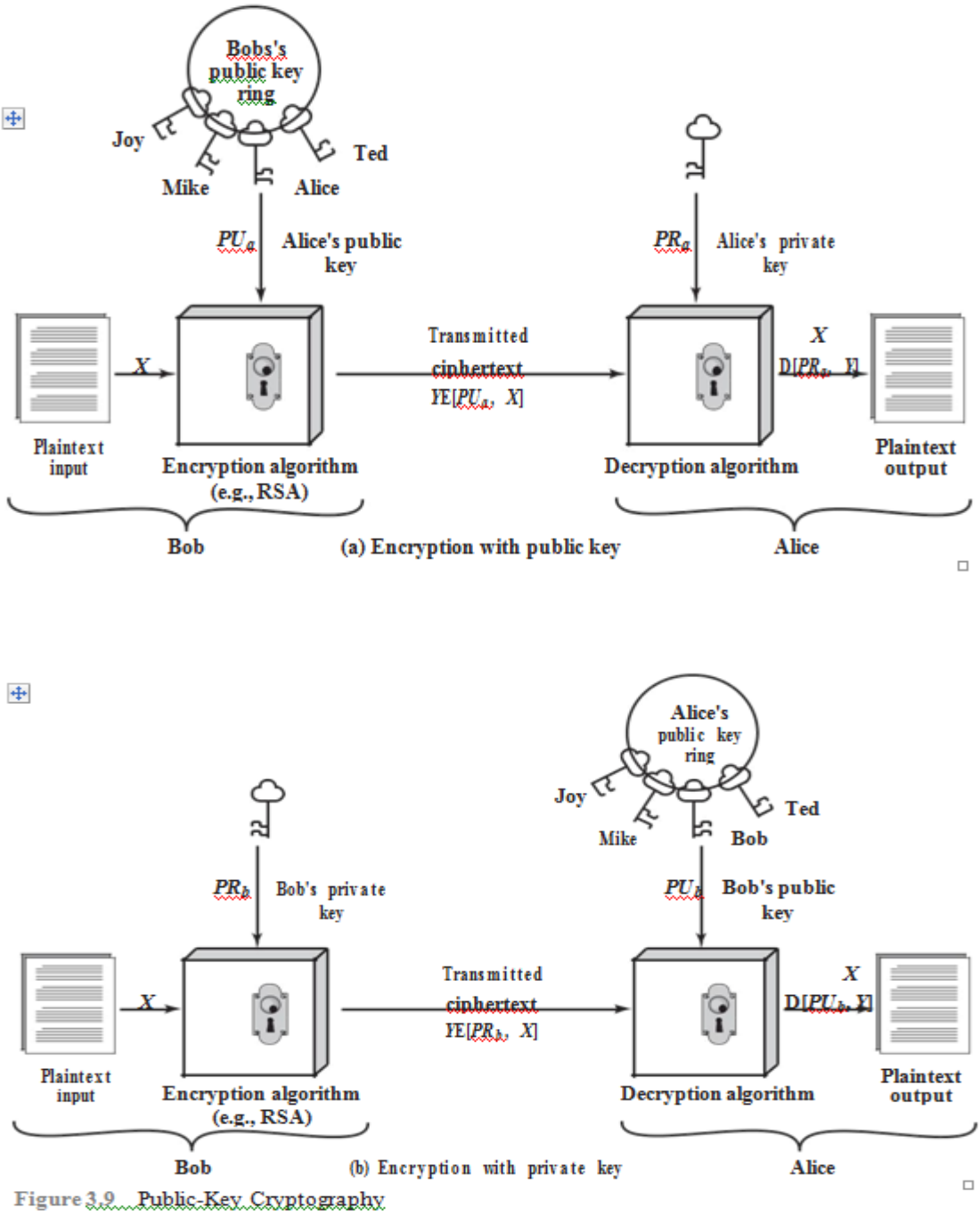

Figure 3.9 Public-Key Cryptography

A public-key encryption scheme has six ingredients (Figure 3.9a)

Plaintext: This is the readable message or data that is fed into the algorithm as input.

Encryption algorithm: The encryption algorithm performs various transforma-tions on the plaintext.

Public and private key: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact trans-formations performed by the encryption algorithm depend on the public or private key that is provided as input.

Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.

Decryption algorithm: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

As the names suggest, the public key of the pair is made public for others to use, while the private key is known only to its owner. A general-purpose public-key cryptographic algorithm relies on one key for encryption and a different but related key for decryption.

The essential steps are the following:

→Each user generates a pair of keys to be used for the encryption and decryption of messages

→Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 3.9a suggests, each user maintains a collection of public keys obtained from others.

→If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.

→When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's pri-vate key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distrib-uted. As long as a user protects his or her private key, incoming communication is secure. At any time, a user can change the private key and publish the companion public key to replace the old public key.

The key used in conventional encryption is typically referred to as a **secret key**. The two keys used for public-key encryption are referred to as the **public key** and the **private key**. Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with conventional encryption.

Applications for Public-Key Cryptosystems

. Public-key systems are characterized by the use of a cryptographic type of algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender's private key, the receiver's public key, or both to perform some type of cryptographic function. In broad terms, we can classify the use of public-key cryptosystems into three categories

Encryption/decryption: The sender encrypts a message with the recipient's public key.

Digital signature: The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

Key exchange: Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No
Elliptic curve	Yes	Yes	Yes

Table 3.2 Applications for Public-Key Cryptosystems

Requirements for Public-Key Cryptography

The cryptosystem illustrated in Figure 3.9 depends on a cryptographic algorithm based on two related keys. Diffie and Hellman postulated this system without demonstrating that such algorithms exist. However, they did lay out the conditions that such algorithms must fulfill [DIFF76]:

It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).

It is computationally easy for a sender A, knowing the public key and the mes-sage to be encrypted, M, to generate the corresponding ciphertext:

$$= \mathrm{E}(PU_b, M)$$

It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$= \mathrm{D}(PR_b, C) = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

It is computationally infeasible for an opponent, knowing the public key, PU_b, to determine the private key, PR_b.

It is computationally infeasible for an opponent, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.

We can add a sixth requirement that, although useful, is not necessary for all Public-key applications.

Either of the two related keys can be used for encryption, with the other used for decryption.

$$= \mathrm{D}[PU_b, \mathrm{E}(PR_b, M)] = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

PUBLIC KEY CRYPTOGRAPHY ALGORITHMS:

The two most widely used public-key algorithms are

1. RSA and
2. Diffie-Hellman.

In this section and then briefly introduce two other algorithms

The RSA Public-Key Encryption Algorithm

One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [RIVE78]. The RSA scheme has since that time reigned supreme as the most widely accepted and imple-mented approach to public-key encryption. **RSA** is a block cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n.

Encryption and decryption are of the following form period for some plaintext block M and ciphertext block C:

$= M^e \bmod n$

$= C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

Both sender and receiver must know the values of n and e, and only the receiver knows the value of d. This is a public-key encryption algorithm with a pub-lic key of $KU = \{e, n\}$ and a private key of $KR = \{d, n\}$. For this algorithm to be sat-isfactory for public-key encryption, the following requirements must be met

It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$.
It is relatively easy to calculate M^e and C^d for all values of $M < n$.

It is infeasible to determine d given e and n

The first two requirements are easily met. The third requirement can be met for large values of e and n.

Figure 3.10 summarizes the RSA algorithm. Begin by selecting two prime numbers p and q and calculating their product n, which is the modulus for encryp-tion and decryption. Next, we need the quantity $\mathbf{f}(n)$, referred to as the Euler totient of n, which is the number of positive integers less than n and relatively prime to n. Then select an integer e that is relatively prime to $\mathbf{f}(n)$ [i.e., the greatest com-mon divisor of e and $\mathbf{f}(n)$ is 1]. Finally, calculate d as the multiplicative inverse of e, modulo $\mathbf{f}(n)$. It can be shown that d and e have the desired properties

Suppose that user A has published its public key and that user B wishes to send the message M to A. Then B calculates $C = M^e \pmod n$ and transmits C. On receipt of this ciphertext, user A decrypts by calculating $M = C^d \pmod n$.

An example, from [SING99], is shown in Figure 3.11. For this example, the keys were generated as follows:

Select two prime numbers, $p = 17$ and $q = 11$.

Calculate $n = pq = 17 \times 11 = 187$

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\varphi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\varphi(n), e) = 1; 1 < e < \varphi(n)$
Calculate d	$de \bmod \varphi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Figure 3.10 The RSA Algorithm

Calculate **f**(n) = (p - 1)(q - 1) = 16 × 10 = 160.

Select e such that e is relatively prime to **f**(n) = 160 and less than **f**(n); we choose e = 7.
Determine d such that de mod 160 = 1 and d < 160. The correct value is d = 23, because 23 × 7 = 161 = (1 × 160) + 1.

The resulting keys are public key PU = {7, 187} and private key PR = {23, 187}. The example shows the use of these keys for a plaintext input of M = 88. For

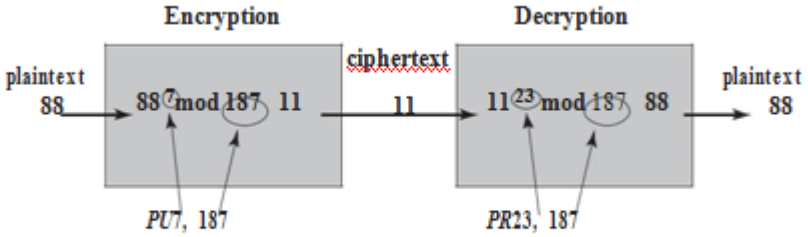

Figure 3.11 Example of RSA Algorithm

encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows:

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$88^1 \bmod 187 = 88$

$88^2 \bmod 187 = 7744 \bmod 187 = 77$

$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$

$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$11^1 \bmod 187 = 11$

$11^2 \bmod 187 = 121$

$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$

$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$

$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187$

$79{,}720{,}245 \bmod 187 = 88$

There are two possible approaches to defeating the RSA algorithm. The first is the brute-force approach: Try all possible private keys. Thus, the larger the number of bits in *e* and *d*, the more secure the algorithm. However, because the calculations involved (both in key generation and in encryption/decryption) are complex, the larger the size of the key, the slower the system will run.

Most discussions of the cryptanalysis of RSA have focused on the task of fac-toring *n* into its two prime factors. For a large *n* with large prime factors, factoring is a hard problem, but not as hard as it used to beA striking illustration of this occurred in 1977; the three inventors of RSA challenged *Scientific American* readers to decode a cipher they printed in Martin Gardner's "Mathematical Games" column [GARD77]. They offered a $100 reward for the return of a plaintext sentence, an event they predicted might not occur for some 40 quadrillion years. In April of 1994, a group working over the Internet and using over 1600 computers claimed the prize after only eight months of work [LEUT94]. This challenge used a public-key size (length of *n*) of 129 decimal digits (approximately 428 bits). This result does not

invalidate the use of RSA; it simply means that larger key sizes must be used. Currently, a 1024-bit key size (about 300 decimal digits) is considered strong enough for virtually all applications

Diffie-Hellman Key Exchange

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [DIFF76] and is generally referred to as the **Diffie-Hellman key exchange**. A number of commercial products employ this key exchange technique

The purpose of the algorithm is to enable two users to exchange a secret key securely that then can be used for subsequent encryption of messages. The algo-rithm 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

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

are distinct and consist of the integers from 1 through p - 1 in some permutation. For any integer b less than p and a primitive root a of prime number p, one can

find a unique exponent i such that

$$b = a^i \bmod p \quad 0 \ldots i \ldots (p - 1)$$

The exponent i is referred to as the discrete logarithm, or index, of b for the base a, mod p. We denote this value as $\text{dlog}_{a,p}(b)$.[5]

The Algorithm With this background, we can define the Diffie-Hellman key exchange, which is summarized in Figure 3.12. 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 $K = 1Y_B2^{X_A} \bmod q$ and user B computes the key as $K = 1Y_A2^{X_B} \bmod q$. These two calculations produce identical results:

$$= 1Y_B2^{X_A} \bmod q$$

$$1\alpha^{X_B} \bmod q2^{X_A} \bmod q$$
$$1\alpha^{X_B}2^{X_A} \bmod q$$

$$\alpha^{X_B X_A} \bmod q$$
$$1\alpha^{X_A}2^{X_B} \bmod q$$

$$1\alpha^{X_A} \bmod q2^{X_B} \bmod q$$
$$1Y_A2^{X_B} \bmod q$$

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

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

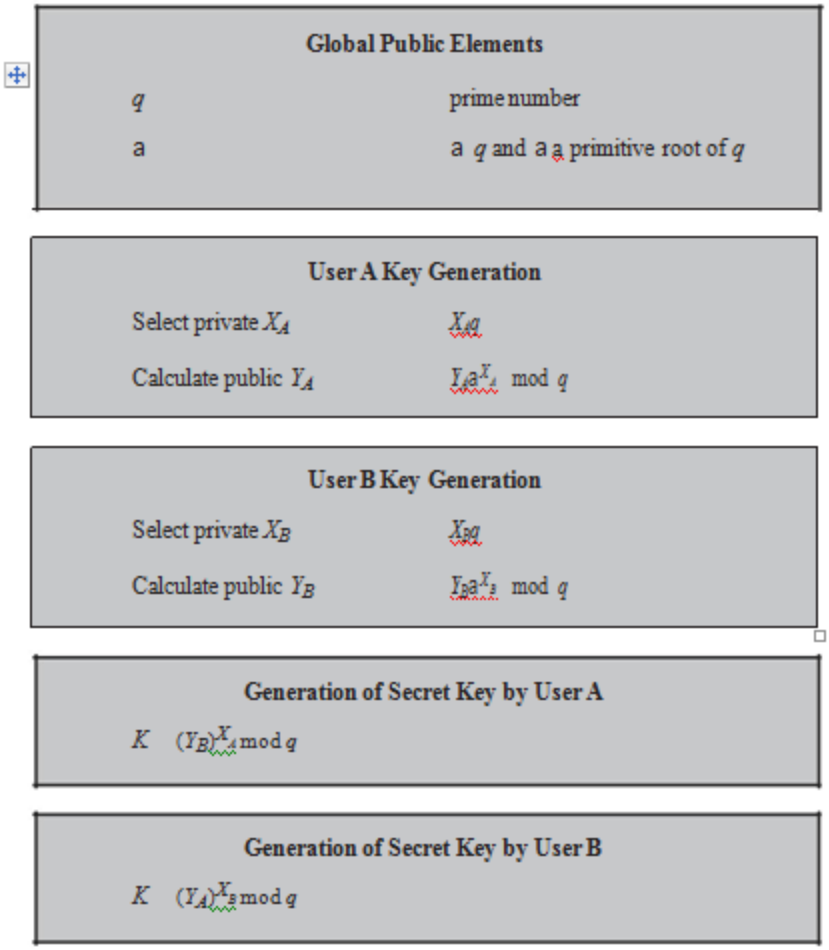

Figure 3.12 The Diffie-Hellman Key Exchange Algorithm

The adversary can then calculate the key K in the same manner as user B does. 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 calcu-late 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

= 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:

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

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

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

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

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

We assume an attacker would have available the following information:

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

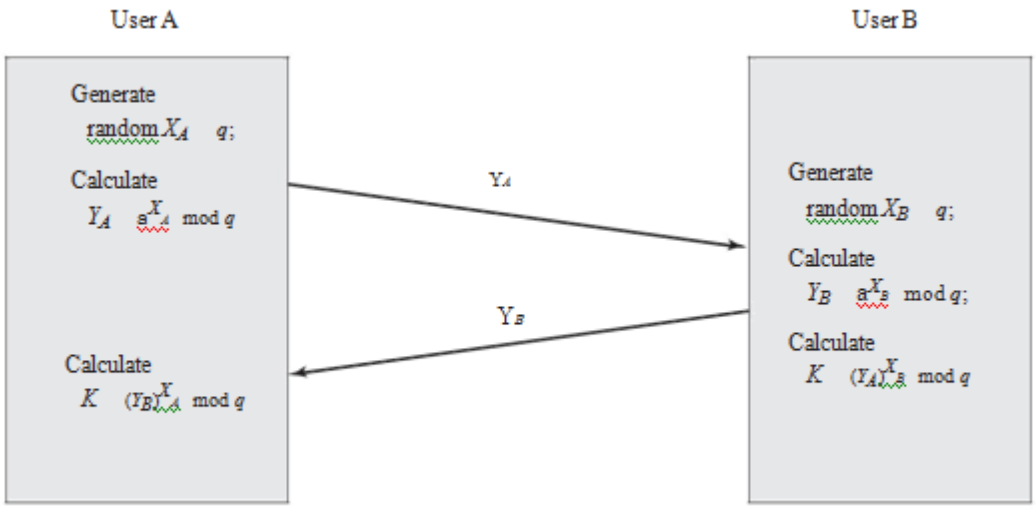

Figure 3.13 Diffie-Hellman Key Exchange

In this simple example, it would be possible to determine the secret key 160 by brute force. 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

KEY EXCHANGE PROTOCOLS Figure 3.13 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 communica-tion 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

MAN-IN-THE-MIDDLE ATTACK The protocol depicted in Figure 3.13 is insecure against a man-in-the-middle attack. Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows:

Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2}, and then computing the corresponding public keys Y_{D1} and Y_{D2}.

Alice transmits Y_A to Bob.

3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates

$K2 = 1Y\ 2^X D2 \bmod q$.

Bob receives Y_{D1} and calculates $K1 = 1Y_{D1}2^X B \bmod q$.

Bob transmits Y_B to Alice.^{A}

6. Dart h intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates

$K1 = 1Y_B2^X D1 \bmod q$.

Alice receives Y_{D2} and calculates

7. $K2 = 1Y_{D2}2^X A \bmod q$.

At this point, Bob and Alice think that they share a secret key. Instead Bob and Darth share secret key $K1$, and Alice and Darth share secret key $K2$. All future communication between Bob and Alice is compromised in the following way:

Alice sends an encrypted message M: E($K2$, M).

Darth intercepts the encrypted message and decrypts it to recover M.

Darth sends Bob E($K1$, M) or E($K1$, M'), where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates