UNIT III **PUBLIC KEY CRYPTOGRAPHY**

MATHEMATICS OF ASYMMETRIC KEY CRYPTOGRAPHY: Primes – Primality Testing – Factorization – Euler‘s totient function, Fermat‘s and Euler‘s Theorem – Chinese Remainder Theorem – Exponentiation and logarithm – ASYMMETRIC KEY CIPHERS: RSA cryptosystem – Key distribution – Key management – Diffie Hellman key exchange – ElGamal cryptosystem – Elliptic curve arithmetic-Elliptic curve cryptography

MATHEMATICS OF ASYMMETRIC KEY CRYPTOGRAPHY

- ❖ Primes
- ❖ Primality Testing
- ❖ Factorization
- ❖ Euler‘s totient function, Fermat‘s and Euler‘s Theorem
- ❖ Chinese Remainder Theorem
- ❖ Exponentiation and logarithm

Prime numbers

- ✔ An integer $p>1$ is a prime number if and only if its only divisors are ±1 and ±p.
- ✔ Any integer $a>1$ can be factored in a unique way as

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t}$$

where $p_1<p_2<\ldots\ldots p_t$ are prime numbers and where each a_i is a positive integer.

- ✔ This is known as the fundamental theorem of arithmetic.

$$91 = 7 \times 13$$
$$3600 = 2^4 \times 3^2 \times 5^2$$
$$11011 = 7 \times 11^2 \times 13$$

- ✔ It is useful for what follows to express another way. If P is the set of all prime numbers, then any positive integer a can be written uniquely in the following form:

$$a = \prod_{p \in P} p^{a_p} \quad \text{where each } a_p \geq 0$$

- ✔ The right-hand side is the product over all possible prime numbers p; for any particular value of a, most of the exponents a_p will be 0.
- ✔ It is easy to determine the greatest common divisor of two positive integers if we express each integer as the product of primes.

$$300 = 2^2 \times 3^1 \times 5^2$$
$$18 = 2^1 \times 3^2$$
$$\gcd(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$$

- ✔ The following relationship always holds:

If $k = \gcd(a, b)$, then $k_p = \min(a_p, b_p)$ for all p.

Primality Testing:

- ✔ A primality test is an algorithm for determining whether an input number is prime. Among other fields of mathematics, it is used for cryptography.
- ✔ Primality tests do not generally give prime factors, only stating whether the input number is prime or not.
- ✔ Some primality tests *prove* that a number is prime

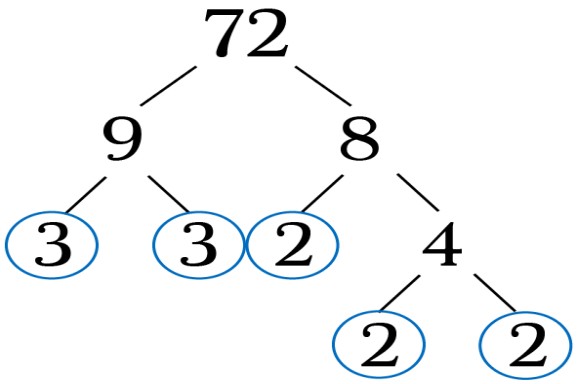
The Sieve of Eratosthenes:

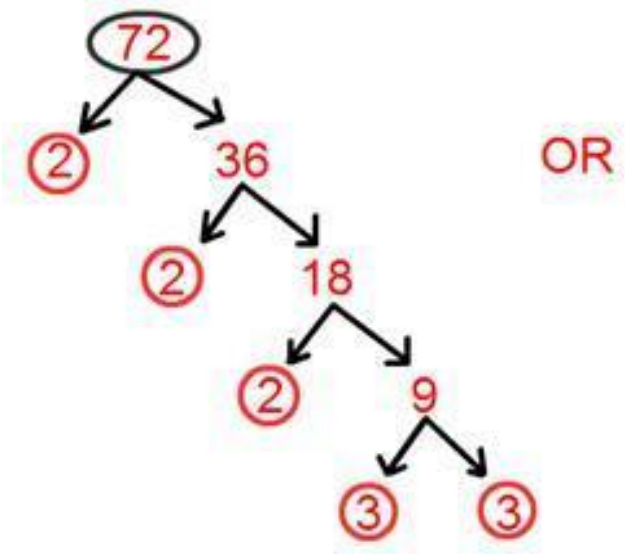
- ✔ Here is a systematic way of writing down all the prime numbers, and all the composite numbers, up to 100.
- ✔ Write down all the whole numbers up to 100.
- ✔ Cross out 0 and 1 — they are exceptions.
- ✔ Circle the next number 2, and then cross out every multiple of 2
- ✔ Circle the next number not crossed out (which is 3), and then cross out multiples of 3.
- ✔ Continue until all numbers are either circled or crossed out.

- ✔ The 25 circled numbers are the primes up to 100, and the 74 crossed-out numbers (not including 0 and 1) are the composite numbers up to 100.
- ✔ It is obvious fact that all primes end in 1, 3, 7 or 9, except for 2 and 5, there are no other obvious patterns.

Prime Factorization:

- ✔ Prime factorization is used to "break down" or express a given number as a product of prime numbers.
- ✔ If a prime number occurs more than once in the factorization, it is usually expressed in exponential form to make it look more compact.





OR

2	72
2	36
2	18
3	9
3	3
	1

Fermat's and Euler's theorem

- ✔ Two theorems that play important roles in public-key cryptography are Fermat's theorem and Euler's theorem.

Fermat's Theorem

Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p, then

$$a^{p-1} \equiv 1 \pmod{p} \qquad (8.2)$$

Proof: Consider the set of positive integers less than p: {1, 2, ….. p - 1} and multiply each element by a, modulo p, to get the set X = {a mod p, 2a mod p, ……(p - 1)a mod p}. Multiplying the numbers in both sets (p and X) and taking the result mod p yields

$$a \times 2a \times \cdots \times (p-1)a \equiv [(1 \times 2 \times \cdots \times (p-1)] \pmod{p}$$
$$a^{p-1}(p-1)! \equiv (p-1)! \pmod{p}$$

We can cancel the (p - 1)! term because it is relatively prime to p. This yields Equation (8.2), which completes the proof.

$$a^{p-1} \equiv 1 \pmod{p}$$

An alternative form of Fermat's theorem is also useful: If p is prime and a is a positive integer, then

$$a^p \equiv a \pmod{p} \qquad (8.3)$$

Example 1: $5^{18} \bmod 19$

- ✔ If we don't know fermats theorem or congruent it is difficult for computing $5^{18} \bmod 19$
- ✔ So convert this into fermats formula
- ✔ Now observe the statement
- ✔ P=19 and a=5
- ✔ Both are reatively prime i.e gcd(19,5)=1
- ✔ Hence it can be represented as

$$5^{18} \bmod 19 = 1$$

Example 2: $5^{19} \bmod 19 = 5$ [$a^p \equiv a \pmod{p}$]

Example 3: $5^{20} \bmod 19$

- ✔ $5^{20} \bmod 19 \Rightarrow 5^{19} \times 5^1 \bmod 19$
  - $5^{19} \times 5^1 \bmod 19$
  - $5^{19} \bmod 19 \times 5^1 \bmod 19$(Commutative law)
  - 5x5 mod 19 =25mod19
- ✔ Again perform mod operation

- 25mod19=6

✔ We should apply mod operation until the result is less than mod value

Euler's Totient Function

Before presenting Euler's theorem, we need to introduce an important quantity in number theory, referred to as **Euler's totient function**, written φ(*n*), and defined as the number of positive integers less than *n* and relatively prime to *n*. By convention,

$$\varphi(1) = 1.$$

Table 8.2 lists the first 30 values of φ (*n*). The value φ(1) is without meaning but is defined to have the value 1.

It should be clear that, for a prime number *p*,

$$\varphi(p) = p - 1$$

Now suppose that we have two prime numbers *p* and *q* with $p \neq q$. Then we can show that, for $n = pq$,

$$\varphi(n) = \varphi(pq) = \varphi(p) * \varphi(q) = (p - 1) * (q - 1)$$

Table 8.2 Some Values of Euler's Totient Function $\phi(n)$

n	$\phi(n)$	n	$\phi(n)$	n	$\phi(n)$
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

$$\phi(21) = \phi(3) \times \phi(7) = (3 - 1) \times (7 - 1) = 2 \times 6 = 12$$
where the 12 integers are $\{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$.

Problem 1: Φ(10)

✔ To determine Φ(*n*), we list all the positive integers less than n that are relatively prime to it.

✔ First list all the positive integers less than n

- Φ(10)={1,2,3,4,5,6,7,8,9}

- ✔ Next choose only the positive integers that is relatively prime (gcd should be 1 i.e common factor should be 1)
  - Φ(10) = {1,3,7,9}
- ✔ Finally consider the number of positive integers .Hence
  - Φ(10) =4

DETERMINE $\phi(37)$ AND $\phi(35)$.

Because 37 is prime, all of the positive integers from 1 through 36 are relatively prime to 37. Thus $\phi(37) = 36$.

To determine $\phi(35)$, we list all of the positive integers less than 35 that are relatively prime to it:

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18
19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34

There are 24 numbers on the list, so $\phi(35) = 24$.

Note: It should be clear that, for a prime number *p*,

$$\Phi(p) = p - 1$$

Problem 2: Φ(5)

Φ(5) ={1,2,3,4}

=4

But it is clear that, for a prime number *p*,

$$\Phi(p) = p - 1$$

coming to the same example

Φ(5) =5 – 1

=4

Euler's Theorem

Euler's theorem states that for every *a* and *n* that are relatively prime:

$$a^{\phi(n)} \equiv 1 (\text{mod } n) \qquad \textbf{(8.4)}$$

As is the case for Fermat's theorem, an alternative form of the theorem is also useful:

$$a^{\phi(n)+1} \equiv a (\text{mod } n) \qquad \textbf{(8.5)}$$

we have $a^{\Phi(n)} \equiv 1 (\text{mod } n)$

if n is prime, Φ(n)=n-1

Problem: a=2; n=17

$a^{\Phi(n)} \equiv 1 (\text{mod } n)$ =>if n=17 Φ(n)=16
$2^{16} \text{ mod } 17 = 1$ (By Fermat's theorem)

Testing for primality

Miller-Rabin Algorithm

The algorithm due to Miller and Rabin is typically used to test a large number for primality. Before explaining the algorithm, we need some back-ground. First, any positive odd integer *n* Ú 3 can be expressed as

$$n - 1 = 2^k q \qquad \text{with } k > 0, q \text{ odd}$$

T o see this, note that *n* - 1 is an even integer. Then, divide (*n* - 1) by 2 until the result is an odd number *q*, for a total of *k* divisions. If *n* is expressed as a binary number, then the result is achieved by shifting the number to the right until the rightmost digit is a 1, for a total of *k* shifts.

Details of the Algorithm

The procedure TEST takes a candidate integer *n* as input and returns the result •composite if *n* is definitely not a prime, and the result inconclusive if *n* may or may not be a prime.

```
TEST (n)
1. Find integers k, q, with k > 0, q odd, so that
   (n − 1 = 2^k q);
2. Select a random integer a, 1 < a < n − 1;
3. if a^q mod n = 1 then return("inconclusive");
4. for j = 0 to k − 1 do
5. if a^(2^j q) mod n = n − 1 then return("inconclusive");
6. return("composite");
```

The Chinese remainder theorem

One of the most useful results of number theory is the **Chinese remainder theorem** (CRT).[8] In essence, the CRT says it is possible to reconstruct integers in a certain range from their residues modulo a set of pairwise relatively prime moduli.

The CRT can be stated in several ways.

$$M = \prod_{i=1}^{k} m_i$$

where the m_i are pairwise relatively prime;
that is, gcd(m_i, m_j) = 1 for 1≤i, j≤k and i≠j.

- ✔ Let $n_1,n_2,n_3….nr$ be pairwise relatively prime positive integer. Then the linear system of congruences

$x \equiv a_1(\mathrm{mod} n_1)$
$x \equiv a_2(\mathrm{mod} n_2)$
$x \equiv a_3(\mathrm{mod} n_3)$
.
.
$x \equiv a_r(\mathrm{mod} n_r)$
has a unique solution modulo $n_1,n_2,n_3….n_r$

- ✔ To find the unique solution we have

$x=(a_1M_1y_1 + a_2M_2y_2+ \cdots\cdots a_rM_ry_r)(\mathrm{mod}\ M)$

where M= n1 x n2 x n3 x….n r

$M_1= \frac{M}{n1}$
$M_2= \frac{M}{n2}$
.
.
$M_r= \frac{M}{nr}$

- ✔ To find y1,y2…..yr satisfying

$M_1y_1 \equiv 1(\mathrm{mod} n_1)$
$M_2y_2 \equiv 1(\mathrm{mod} n_2)$
.
.
.
$Mryr \equiv 1(\mathrm{mod} n_r)$

Problem 1:

Find the solution of $x \equiv 1(\mathrm{mod}3)$, $x \equiv 2(\mathrm{mod}4)$ and $x \equiv 3(\mathrm{mod}5)$

Soln:

Given a1=1;a2=2; a3=3
n1=3;n2=4;n3=5 should be pairwise relatively prime to find unique value of x?

- ✔ To find the unique solution we have

$x=(a_1M_1y_1 + a_2M_2y_2+ \cdots\cdots a_rM_ry_r)(\mathrm{mod}\ M)$

where M= n1 x n2 x n3 x….n r

$M_1 = \frac{M}{n1}$

$M_2 = \frac{M}{n2}$

.

.

$M_r = \frac{M}{nr}$

Step:1

Compute M

M=n1 x n2 x n3

=3 x 4 x 5

=60

Step 2: Compute M1,M2,M3

$M_1 = \frac{M}{n1} = \frac{3*4*5}{3} = 20$

$M_2 = \frac{M}{n2} = \frac{3*4*5}{4} = 15$

$M_3 = \frac{M}{n3} = \frac{3*4*5}{5} = 12$

Step :3 To find y1,y2,y3 satisfying

$M_1y_1 \equiv 1(\text{mod} n_1)$

$M_2y_2 \equiv 1(\text{mod} n_2)$

$M_3y_3 \equiv 1(\text{mod} n_3)$

$M_1y_1 \equiv 1(\text{mod} n_1)$

$20\ y1 \equiv 1(\text{mod}3)$

$20*2 = 1 \bmod 3$

y1=2

$M_2y_2 \equiv 1(\text{mod} n_2)$

$15\ y2 \equiv 1(\text{mod}4)$

y2=3

$M_3y_3 \equiv 1(\text{mod} n_3)$

$12\ y3 \equiv 1(\text{mod}5)$

$2y3 \equiv 1(\text{mod} n5)$

y3=3

By Chinese Remainder Theorem the unique solution is

$x=(a_1M_1y_1 + a_2M_2y_2 + a_3M_3y_3) (\text{mod } M)$

$=(1x20x2 + 2x15x3 + 3x12x3) \bmod 60$

$=(40+90+108) \bmod 60$

=238 mod 60
=58

ASYMMETRIC KEY CIPHERS

- RSA cryptosystem
- Key distribution
- Key management
- Diffie Hellman key exchange
- ElGamal cryptosystem
- Elliptic curve arithmetic
- Elliptic curve cryptography

Public key cryptography

Table 9.1 Terminology Related to Asymmetric Encryption

Asymmetric Keys Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.
Public Key Certificate A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.
Public Key (Asymmetric) Cryptographic Algorithm A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.
Public Key Infrastructure (PKI) A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

- ❖ Principles of public key cryptosystems
- ❖ The RSA algorithm

Principles of public key cryptosystems

- ✔ Public-Key Cryptosystems
- ✔ Applications for Public-Key Cryptosystems
- ✔ Requirements for Public-Key Cryptography

- ✔ Public-Key Cryptanalysis

Public-Key Cryptosystems

- ✔ Asymmetric algorithms rely on one key for encryption and a different but related key for decryption.
- ✔ These algorithms have the following important characteristic.
    - o It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms, such as RSA, also exhibit the following characteristic.

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

A **public-key encryption** scheme has six ingredients (Figure 9.1a; compare with Figure 2.1).

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

2. **Encryption algorithm:**
   The encryption algorithm performs various transfor-mations on the plaintext.

3. **Public and private keys:**
   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 algorithm depend on the public or private key that is provided as input.

4. **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.

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

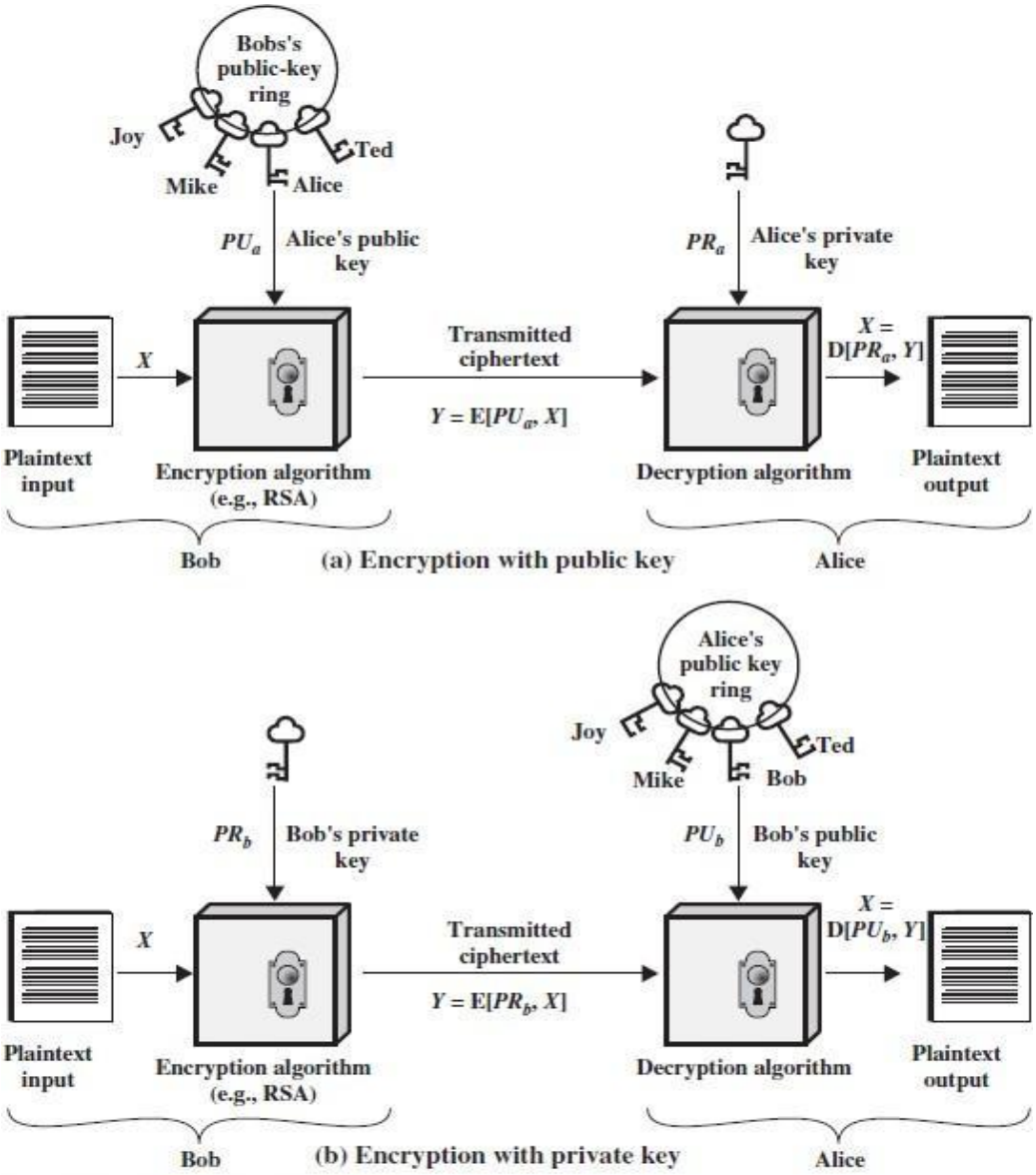


Figure 9.1 Public-Key Cryptography

✔ The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryp-tion of messages.
2. 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 9.1a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the mes-sage using Alice's public key.

4. 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 distributed.

✔ As long as a user's private key remains protected and secret, incoming communication is secure.

✔ At any time, a system can change its private key and publish the companion public key to replace its old public key.

✔ To discriminate between the two, we refer to the key used in symmetric encryption as a **secret key**.

✔ The two keys used for asymmetric 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 symmetric encryption.

Let us take a closer look at the essential elements of a public-key encryption scheme, using Figure 9.2 (compare with Figure 2.2). There is some source A that produces a message in plaintext, $X = [X_1, X_2, \ldots, X_M]$. The M elements of X are letters in some finite alphabet. The message is intended for destination B. B generates a related pair of keys: a public key, PU_b, and a private key, PR_b. PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.

Table 9.2 Conventional and Public-Key Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

With the message X and the encryption key PU_b as input, A forms the ciphertext $Y = [Y_1,$

$Y_2, \ldots, Y_N]$:

$$Y = \mathrm{E}(PU_b, X)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = \mathrm{D}(PR_b, Y)$$

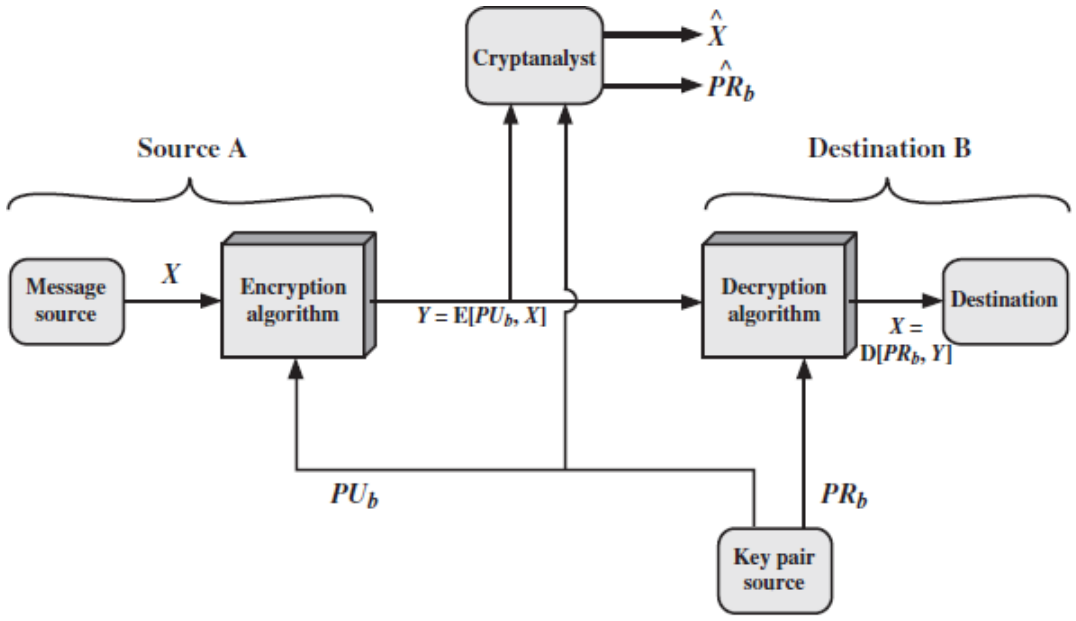


Figure 9.2 Public-Key Cryptosystem: Secrecy

Figures 9.1b and 9.3 show the use of public-key encryption to provide authentication:

$$Y = \mathrm{E}(PR_a, X)$$
$$X = \mathrm{D}(PU_a, Y)$$

In this case, A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a **digital signature**. In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

In the preceding scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document must be kept in plaintext to

be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the sender's private key, it serves as a signature that verifies origin, content, and sequencing.

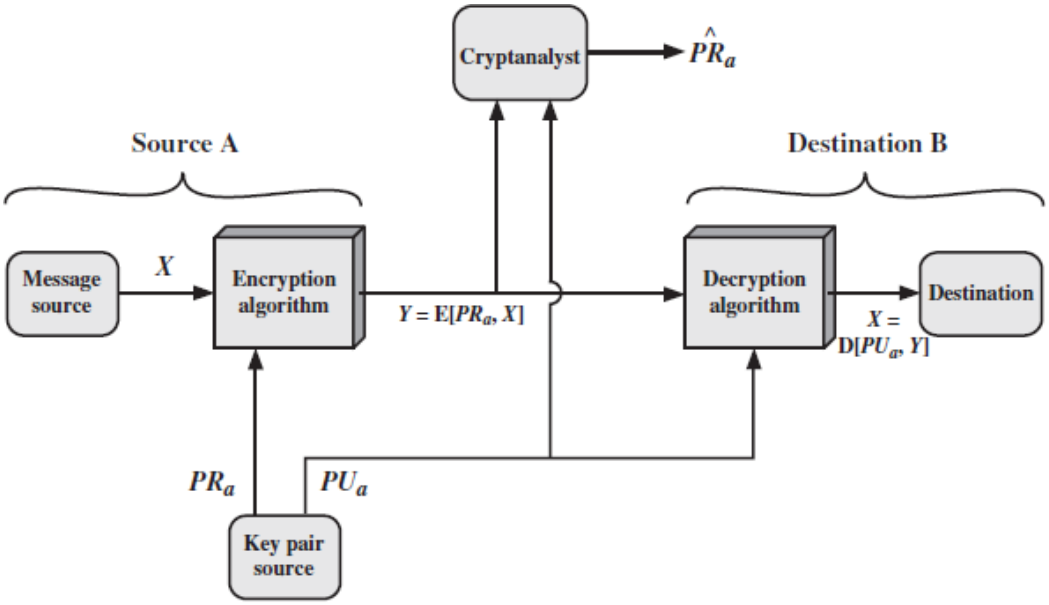


Figure 9.3 Public-Key Cryptosystem: Authentication

It is important to emphasize that the encryption process depicted in Figures 9.1b and 9.3 does not provide confidentiality. That is, the message being sent is safe from alteration but not from eavesdropping. This is obvious in the case of a signature based on a portion of the message, because the rest of the message is transmitted in the clear. Even in the case of complete encryption, as shown in Figure 9.3, there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme (Figure 9.4):

$$Z = E(PU_b, E(PR_a, X))$$
$$X = D(PU_a, D(PR_b, Z))$$

In this case, we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided. The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

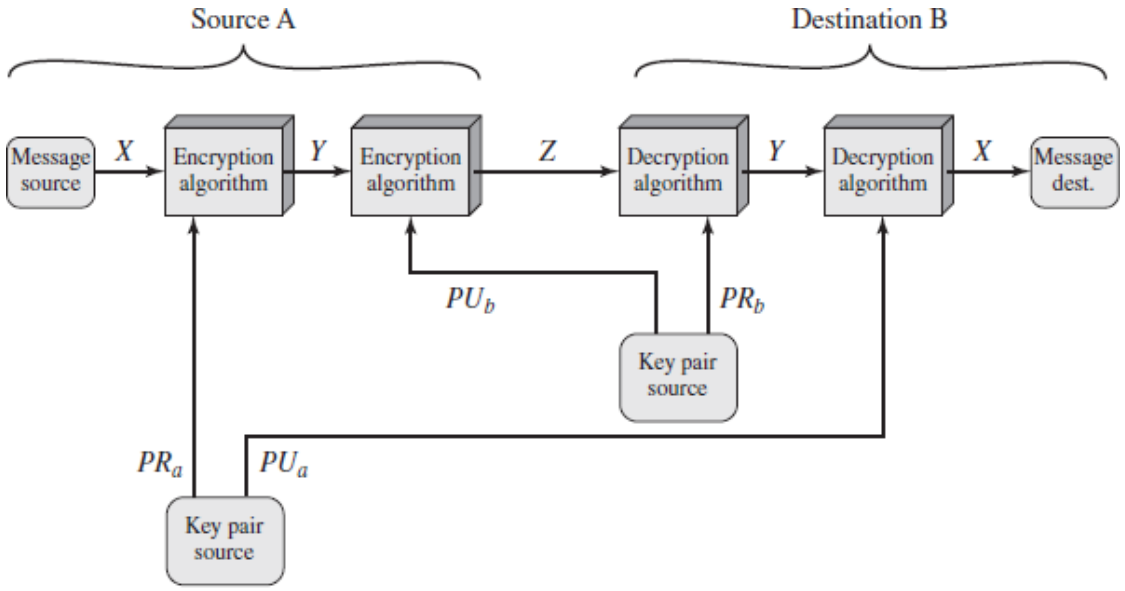


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

Applications for Public-Key Cryptosystems

✔ Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly.

Table 9.3 Applications for Public-Key Cryptosystems

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

✔ Depending on the application, the sender uses either the sender's private key or 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:

1. **Encryption/decryption:**
   The sender encrypts a message with the recipient's public key.
2. **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.

3. **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.

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a pair (public key *PUb*, private key *PRb*).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, *M*, to generate the corresponding ciphertext:

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

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

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

4. It is computationally infeasible for an adversary, knowing the public key, *PUb*, to determine the private key, *PRb*.
5. It is computationally infeasible for an adversary, knowing the public key, *PUb*, and a ciphertext, *C*, to recover the original message, *M*.
6. The two keys can be applied in either order:

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

Public-Key Cryptanalysis

As with symmetric encryption, a public-key encryption scheme is vulnerable to a brute-force attack. The countermeasure is the same: Use large keys. However, there is a tradeoff to be considered. Public-key systems depend on the use of some sort of invertible mathematical function. The complexity of calculating these functions may not scale linearly with the number of bits in the key but grow more rapidly than that. Thus, the key size must be large enough to make brute-force attack impractical but small enough for practical encryption and decryption. In practice, the key sizes that have been proposed do make brute-force attack impractical but result in encryption/ decryption speeds that are too slow for general-purpose use. Instead, as was mentioned earlier, public-key encryption is currently confined to key management and signature applications.

Another form of attack is to find some way to compute the private key given the public key. To date, it has not been mathematically proven that this form of attack is infeasible for a particular public-key algorithm. Thus, any given algorithm, including the widely used RSA algorithm, is suspect. The history of cryptanalysis shows that a problem that seems insoluble from one perspective can be found to have a solution if looked at in an entirely different way.

Finally, there is a form of attack that is peculiar to public-key systems. This is, in essence, a probable-message attack. Suppose, for example, that a message were to be sent that consisted

solely of a 56-bit DES key. An adversary could encrypt all possible 56-bit DES keys using the public key and could discover the encrypted key by matching the transmitted ciphertext. Thus, no matter how large the key size of the public-key scheme, the attack is reduced to a brute-force attack on a 56-bit key. This attack can be thwarted by appending some random bits to such simple messages.

The RSA algorithm

- ✔ The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.
- ✔ The **RSA** scheme is a cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n.
- ✔ A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}.

Description of the Algorithm

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to $\log_2(n) + 1$; in practice, the block size is i bits, where $2^i < n \ldots < 2^{i+1}$. Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C.

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

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met.

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

For now, we focus on the first requirement and consider the other questions later. We need to find a relationship of the form

$$M^{ed} \bmod n = M$$

The preceding relationship holds if e and d are multiplicative inverses modulo $\varphi(n)$, where $\varphi(n)$ is the Euler totient function. It is that for p, q prime,

$$\varphi(pq) = (p - 1)(q - 1)$$

The relationship between e and d can be expressed as

$$ed \bmod \phi(n) = 1 \qquad (9.1)$$

This is equivalent to saying

$$ed \equiv 1 \bmod \phi(n)$$
$$d \equiv e^{-1} \bmod \phi(n)$$

That is, e and d are multiplicative inverses mod $\varphi(n)$. Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\varphi(n)$. Equivalently,

$$\gcd(\varphi(n), d) = 1$$

Equation (9.1) satisfies the requirement for RSA.

are now ready to state the RSA scheme. The ingredients are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

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

Figure 9.5 summarizes the RSA algorithm.

Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Figure 9.5 The RSA Algorithm

For this example, the keys were generated as follows.

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm (Chapter 4).

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 encryption, we need to calculate $C = 887 \bmod 187$. 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 $M = 11^{23} \bmod 187$:

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

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

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

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

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

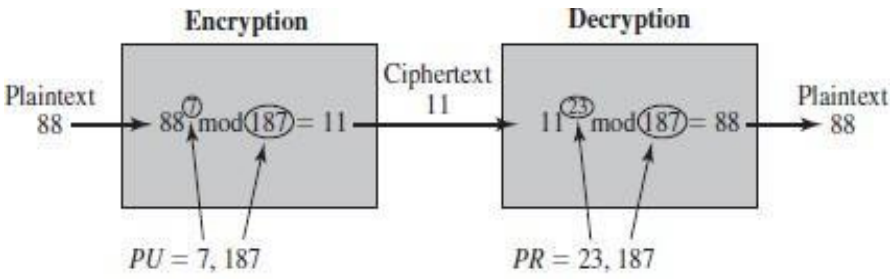


Figure 9.6 Example of RSA Algorithm

Key management

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

❖ 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).

Figure 10.1. Uncontrolled Public-Key Distribution

Merit:

- This approach is convenient.

Demerit:

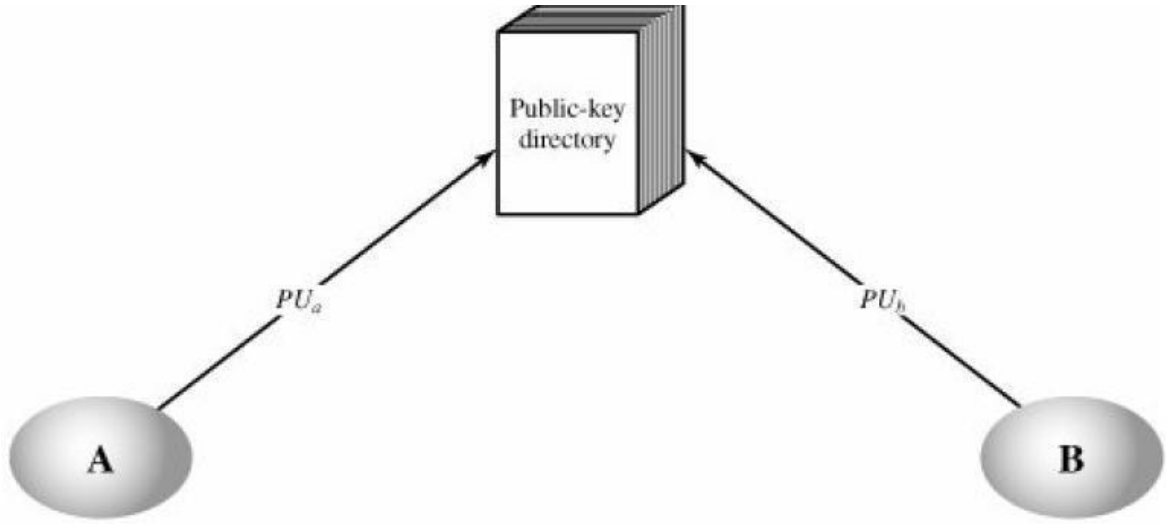
- Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until

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.

✔ **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).

Figure 10.2. Public-Key Publication



- Such a scheme would include the following elements:
  - 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.
  - Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

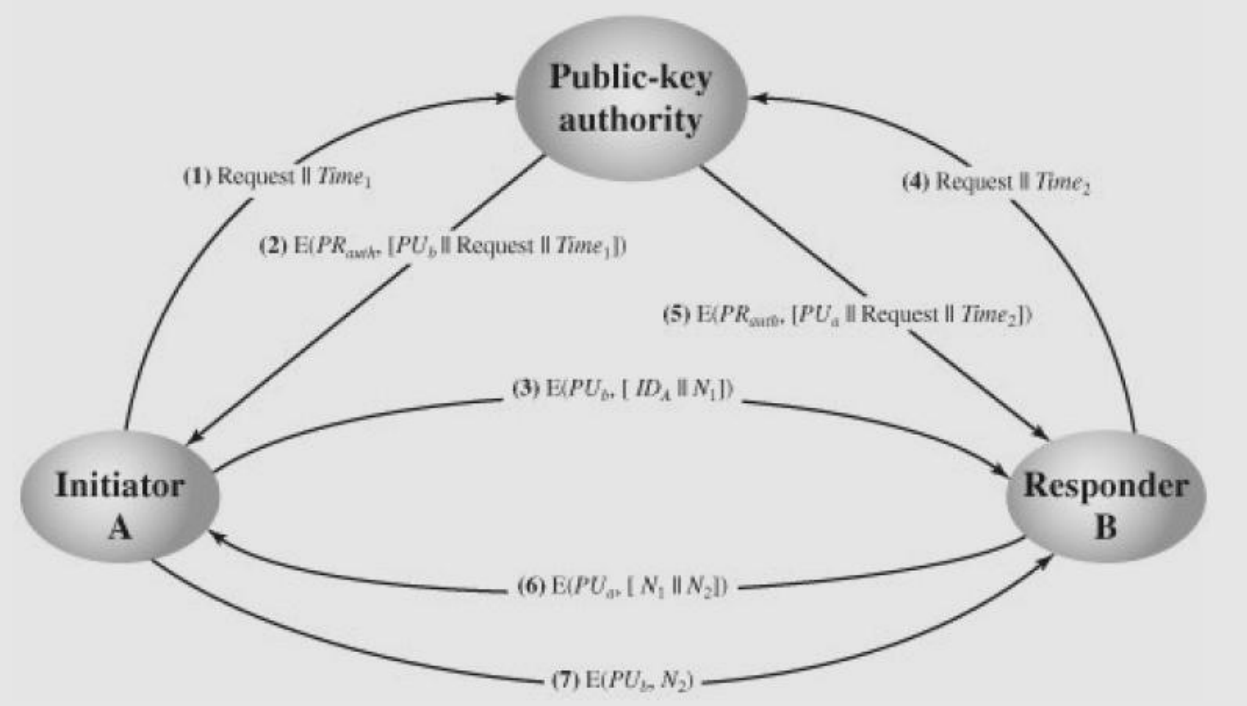
Merit:

- This scheme is clearly more secure than individual public announcements.

✔ **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.
- 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.

Figure 10.3. Public-Key Distribution Scenario



✔ 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, 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, 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.

5. 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 PU_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.

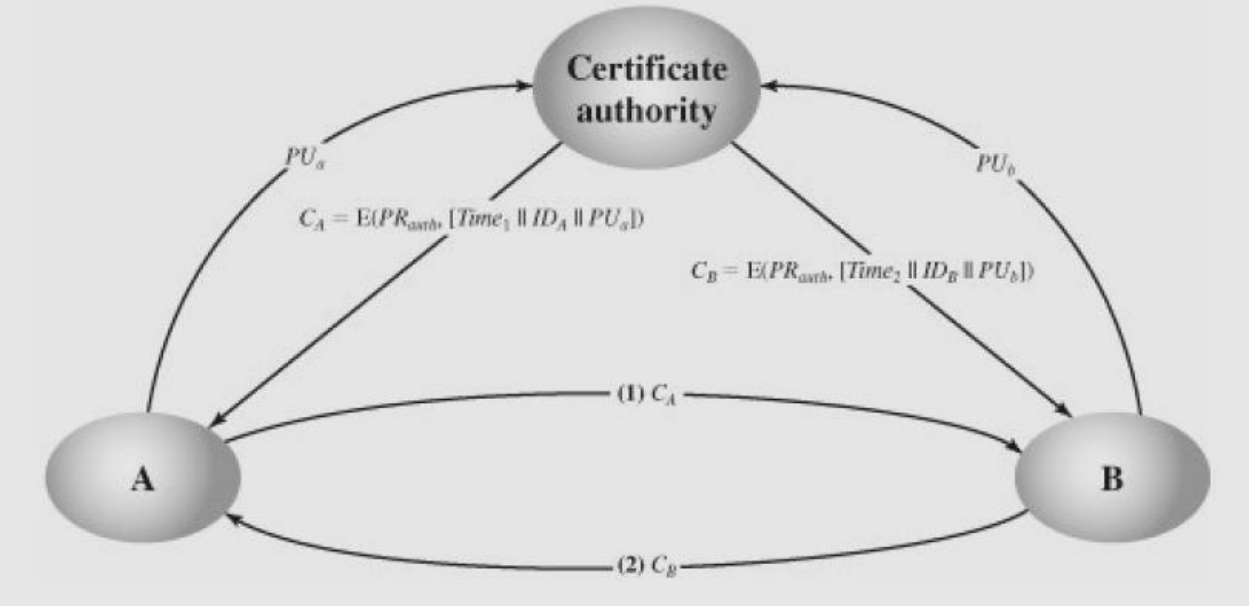
✔ **Public-Key Certificates**

- An alternative approach compared to the previous one 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 plus an identifier of the key owner, with 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 needed 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 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.

Figure 10.4. Exchange of Public-Key Certificates



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, who reads and verifies the certificate as follows:

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

The recipient uses the authority's public key, PU_{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 PU_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 adversary. A generates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the adversary replays the old certificate to B. If B then encrypts messages using the compromised old public key, the adversary can read those messages.

❖ Distribution of Secret Keys Using Public-Key Cryptography

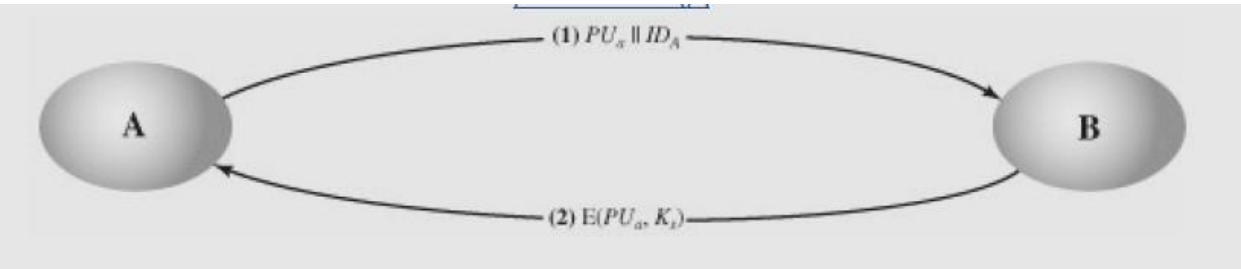
✔ Simple secret key distribution

✔ Secret Key Distribution with Confidentiality and Authentication

✔ Simple Secret Key Distribution

- An extremely simple scheme was put forward by Merkle as illustrated in Figure 10.5.

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



- 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, 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.

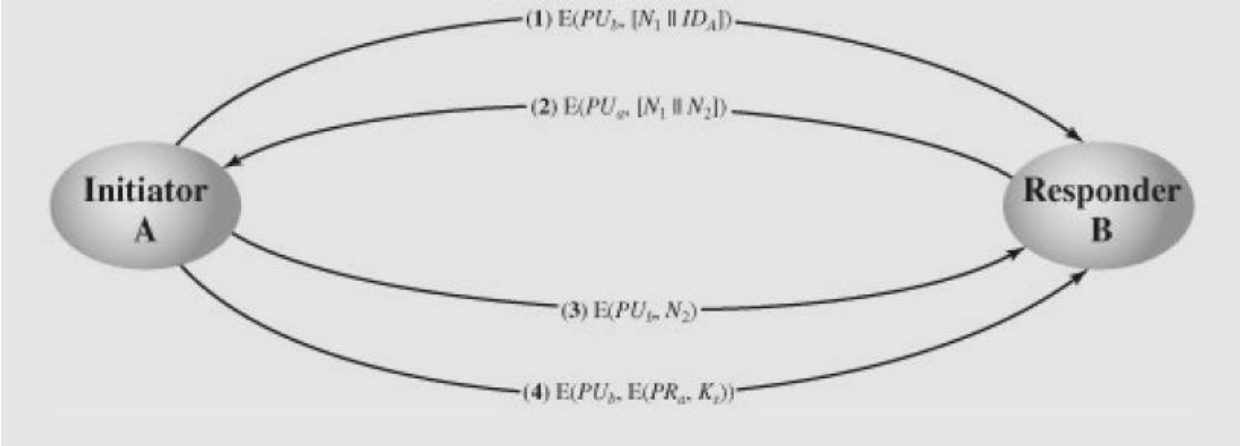
✔ Secret Key Distribution with Confidentiality and Authentication

- Figure 10.6, 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:

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 PU_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 $M = E(PU_b, E(PR_a, K_s))$ 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.

Figure 10.6. Public-Key Distribution of Secret Keys



Diffie Hellman Key exchange

- ✔ The first published public-key algorithm appeared by Diffie and Hellman that defined public-key cryptography and is generally referred to as Diffie-Hellman key exchange.
- ✔ The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages.
- ✔ The algorithm itself is limited to the exchange of secret values.
- ✔ 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. Recall that a primitive root of a prime number *p* is one whose powers modulo *p* 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

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

  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

$$b \equiv a^i \pmod{p} \qquad \text{where } 0 \leq i \leq (p-1)$$

- ✔ The exponent *i* is referred to as the **discrete logarithm** of *b* for the base *a*, mod *p*.

The Algorithm

- ✔ Figure 10.1 summarizes the Diffie-Hellman key exchange algorithm.
- ✔ For this scheme, there are two publicly known numbers: a prime number *q* and an integer a that is a primitive root of *q*.
- ✔ Suppose the users A and B wish to create a shared key.
- ✔ User A selects a random integer $X_A < q$ and computes $Y_A = a^{XA} \bmod q$.
- ✔ Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = a^{XB} \bmod q$.
- ✔ Each side keeps the *X* value private and makes the *Y* value available publicly to the other side.
- ✔ Thus, X_A is A's private key and Y_A is A's corresponding public key, and similarly for B.
- ✔ User A computes the key as $K = (Y_B)^{XA} \bmod q$ and user B computes the key as $K = (Y_A)^{XB}$

mod q.

- ✔ These two calculations produce identical results:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{by the rules of modular arithmetic} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

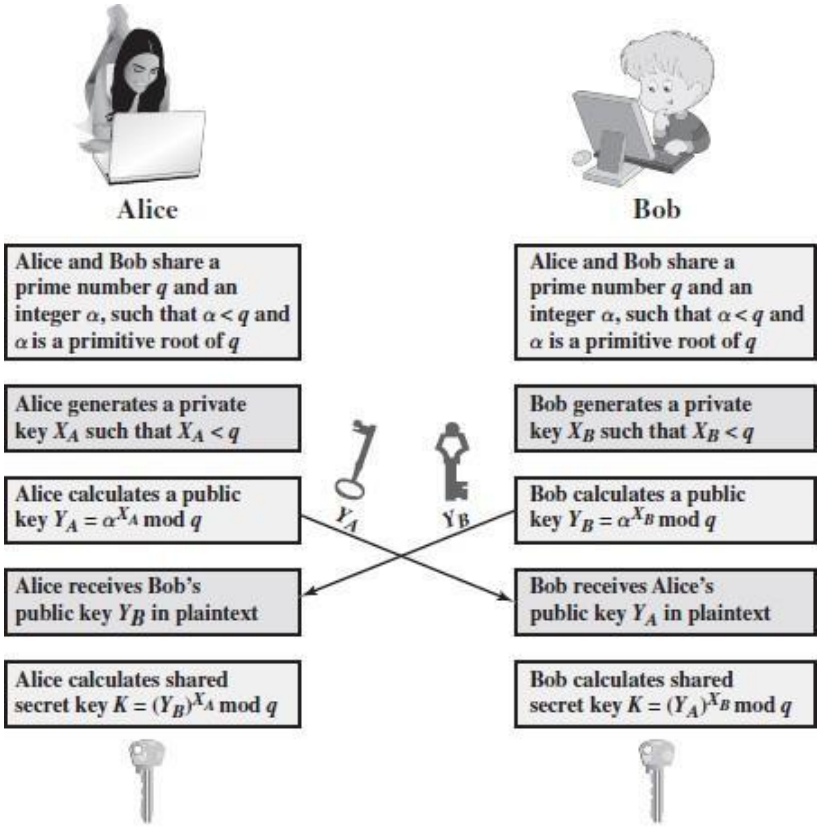


Figure 10.1 The Diffie-Hellman Key Exchange

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 a = 3. A and B select private 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:

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

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A

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

Calculation of Secret Key by User B

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

Figure 10.1 The Diffie-Hellman Key Exchange Algorithm

The key exchange protocols

- ✔ Figure 10.2 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 and 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.

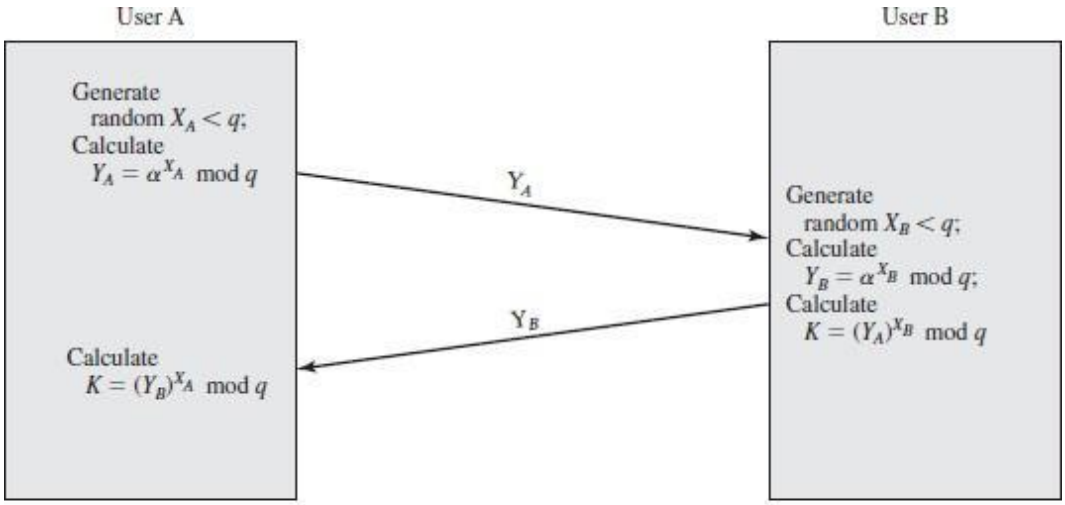


Figure 10.2 Diffie-Hellman Key Exchange

Elliptic Curve Arithmetic

Abelian Groups

An **abelian group** *G*, sometimes denoted by {*G*, . }, is a set of elements with a binary operation, denoted by ., that associates to each ordered pair (*a*, *b*) of elements in *G* an element (*a* · *b*) in *G*, such that the following axioms are obeyed:

(A1) Closure:	If *a* and *b* belong to *G*, then $a \cdot b$ is also in *G*.
(A2) Associative:	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all *a*, *b*, *c* in *G*.
(A3) Identity element:	There is an element e in *G* such that $a \cdot e = e \cdot a = a$ for all *a* in *G*.
(A4) Inverse element:	For each *a* in *G* there is an element *a′* in *G* such that $a \cdot a' = a' \cdot a = e$.
(A5) Commutative:	$a \cdot b = b \cdot a$ for all *a*, *b* in *G*.

An **elliptic curve** is defined by an equation in two variables with coefficients. For cryptography, the variables and coefficients are restricted to elements in a finite field, which results in the definition of a finite abelian group.

Elliptic Curves over Real Numbers

Elliptic curves are not ellipses. They are so named because they are described by cubic equations, similar to those used for calculating the circumference of an ellipse. In general, cubic equations for elliptic curves take the following form, known as a **Weierstrass equation:**

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where *a*, *b*, *c*, *d*, *e* are real numbers and *x* and *y* take on values in the real numbers.

Elliptic curve cryptography

The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiple additions is the counterpart of modular exponentiation.

Analog of Diffie-Hellman Key Exchange

- Key exchange using elliptic curves can be done in the following manner. First pick a large integer q, which is either a prime number p or an integer of the form 2^m, and elliptic curve parameters a and b.
- This defines the elliptic group of points $E_q(a, b)$. Next, pick a *base point* $G = (x_1, y_1)$ in $E_p(a, b)$ whose order is a very large value n.
- The **order** n of a point G on an elliptic curve is the smallest positive integer n such that $nG = 0$ and G are parameters of the cryptosystem known to all participants.

Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

Figure 10.7 ECC Diffie-Hellman Key Exchange

- A key exchange between users A and B can be accomplished as follows (Figure 10.7).

1. A selects an integer n_A less than n. This is A's private key. A then generates a public key $P_A = n_A * G$; the public key is a point in $E_q(a, b)$.
2. B similarly selects a private key n_B and computes a public key P_B.
3. A generates the secret key $k = n_A * P_B$. B generates the secret key $k = n_B * P_A$.

   The two calculations in step 3 produce the same result because

$$n_A * P_B = n_A * (n_B * G) = n_B * (n_A * G) = n_B * P_A$$

Elliptic Curve Encryption/Decryption

- ✔ As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group $Eq(a, b)$ as parameters.
- ✔ Each user A selects a private key n_A and generates a public key $P_A = nA * G$.
- ✔ To encrypt and send a message P_m to B, A chooses a random positive integer k and

$$C_m = \{kG, P_m + kP_B\}$$

  produces the ciphertext C_m consisting of the pair of points:

- ✔ Note that A has used B's public key P_B.
- ✔ To decrypt the ciphertext, B multiplies the first point in the pair by B's private key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

ElGamal cryptosystem

- The ElGamal cryptosystem is used in some form in a number of standards including the digital signature standard (DSS).
- As with Diffie-Hellman, the global elements of ElGamal are a prime number q and **α,** which is a primitive root of q.

- User A generates a private/public key pair as follows:

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y^A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range $0 \leq M \leq q - 1$. Longer messages are sent as a sequence of blocks, with each block being an integer less than q.
2. Choose a random integer k such that $1 \leq k \leq q - 1$.
3. Compute a one-time key $K = (Y_A)^k \bmod q$.
4. Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 = \alpha^k \bmod q; \quad C_2 = KM \bmod q$$

User A recovers the plaintext as follows:

1. Recover the key by computing $K = (C_1)^{X_A} \bmod q$.
2. Compute $M = (C_2 K^{-1}) \bmod q$.

- We can restate the ElGamal process as follows, using Figure 10.3.

1. Bob generates a random integer k.
2. Bob generates a one-time key K using Alice's public-key components Y_A, q, and k.
3. Bob encrypts k using the public-key component α, yielding C_1. C_1 provides sufficient information for Alice to recover K.
4. Bob encrypts the plaintext message M using K.
5. Alice recovers K from C_1 using her private key.
6. Alice uses K^{-1} to recover the plaintext message from C_2.

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{XA} \bmod q$
Public key	$PU = \{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{XA} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

Figure 10.3 The ElGamal Cryptosystem

- **Example:**

For example, let us start with the prime field GF(19); that is, q 19. It has primitive roots {2, 3, 10, 13, 14, 15}. We choose α=10.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^5 \bmod 19 = 3$ (see Table 8.3).
3. Alice's private key is 5; Alice's pubic key is $\{q, \alpha, Y_A\} = \{19, 10, 3\}$.

Suppose Bob wants to send the message with the value $M = 17$. Then,

1. Bob chooses $k = 6$.
2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$.
3. So
   $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$
   $C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext $(11, 5)$.

For decryption:

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2 K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$.