Module - III

ELLIPTIC CURVE ARITHMETIC

Most of the products and standards that use public-key cryptography for encrypt- tion and digital signatures use RSA. As we have seen, the key length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA. This burden has ramifications, especially for electronic commerce sites that conduct large numbers of secure transactions. A competing system challenges RSA: elliptic curve cryptography (ECC). ECC is showing up in standardization efforts, including the IEEE P1363 Standard for Public-Key Cryptography.

The principal attraction of ECC, compared to RSA, is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead. On the other hand, although the theory of ECC has been around for some time, it is only recently that products have begun to appear and that there has been sustained cryptanalytic interest in probing for weaknesses. Accordingly, the confidence level in ECC is not yet as high as that in RSA.

ECC is fundamentally more difficult to explain than either RSA or Diffie- Hellman, and a full mathematical description is beyond the scope of this book. This section and the next give some background on elliptic curves and ECC. We begin with a brief review of the concept of Abelian group. Next, we examine the concept of elliptic curves defined over the real numbers. This is followed by a look at elliptic curves defined over finite fields. Finally, we are able to examine elliptic curve ciphers.

Abelian Groups

Recall from Chapter 4 that an **abelian group** G, sometimes denoted by $\{G, \bullet\}$, is a set of elements with a binary operation, denoted by $\bullet$, that associates to each ordered pair (a, b) of elements in G an element $(a \bullet b)$ in G, such that the following axioms are obeyed:[3]

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

A number of public-key ciphers are based on the use of an abelian group. For example, Diffie-Hellman key exchange involves multiplying pairs of nonzero integers modulo a prime number q. Keys are generated by exponentiation over the group, with exponentiation defined as repeated multiplication. For example,

$a^k \bmod q = \underbrace{(a \times a \times \ldots \times a)}_{k \text{ times}} \bmod q$. To attack Diffie-Hellman, the attacker must determine k given a and a^k; this is the discrete logarithm problem.

For elliptic curve cryptography, an operation over elliptic curves, called addi- tion, is used. Multiplication is defined by repeated addition. For example,

$$a \times k = \underbrace{(a + a + \ldots + a)}_{k \text{ times}}$$

where the addition is performed over an elliptic curve. Cryptanalysis involves determining k given a and (a * k).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. Before looking at this, we first look at elliptic curves in which the variables and coefficients are real numbers. This case is perhaps easier to visualize.

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:

y2 + ax + by = x3 + cx2 + dx + e

Where a, b, c, d, e are real numbers and x and y take on values in the real numbers.4 for our purpose, it is sufficient to limit ourselves to equations of the form

$$y^2 = x^3 + ax + b \qquad \textbf{(10.1)}$$

Such equations are said to be cubic, or of degree 3, because the highest exponent they contain is a 3. Also in cluded in the definition of an elliptic curve is a single elementdenoted *O* and called the *point at infinity* or t he *zero point*, which we discuss subsequently. To plot such a curve, we need to compute

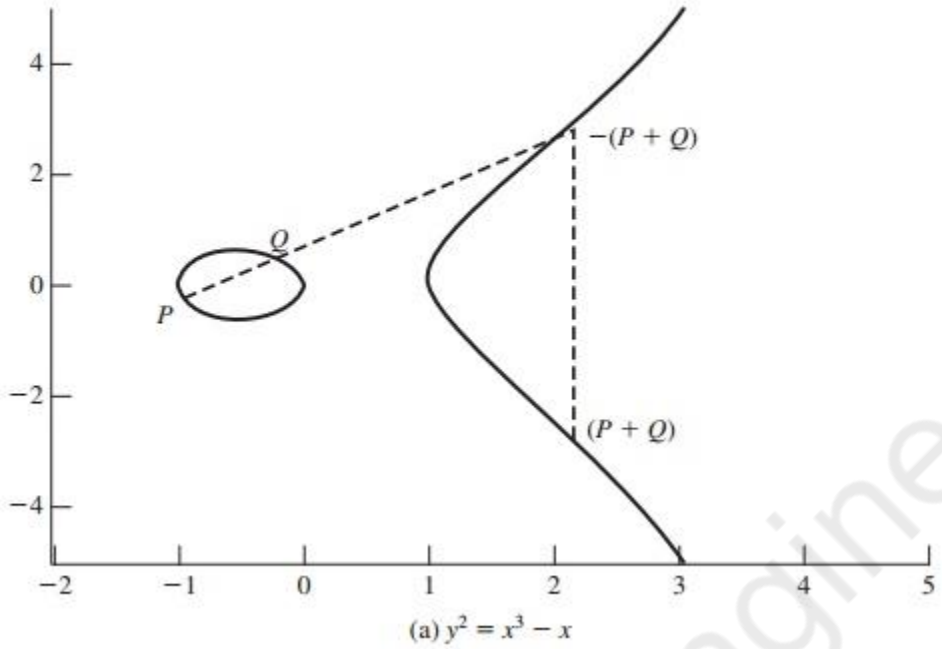
$$y = \sqrt{x^3 + ax + b}$$

For given values of a and b, the plot consists of positive and negative values of y for each value of x. Thus, each curve is symmetric about y = 0. Figure 10.4 shows two examples of elliptic curves. As you can see, the formula sometimes produces weird- looking curves.

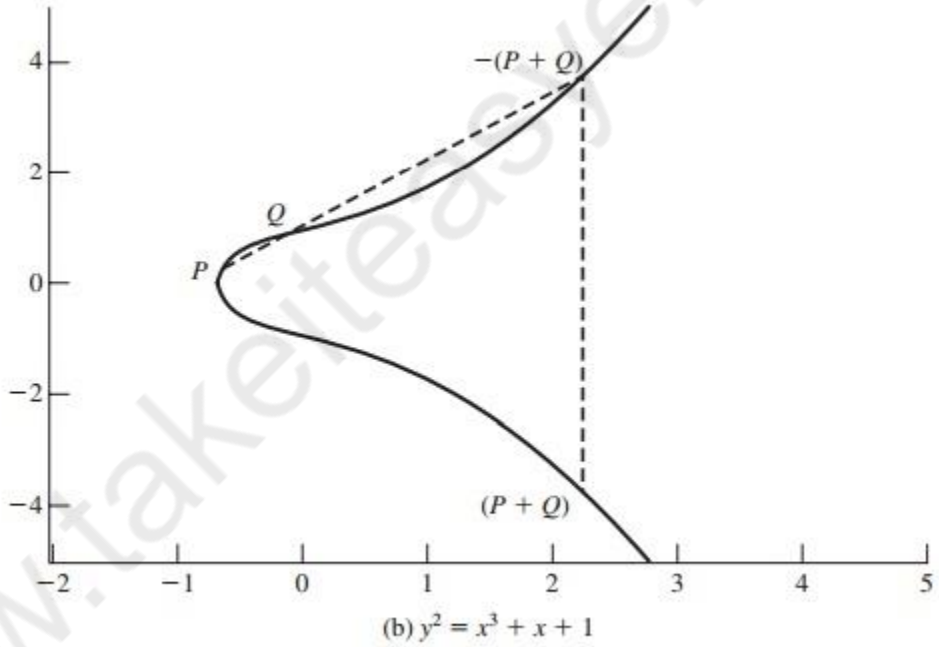
Now, consider the set of points E (a, b) consisting of all of the points (x, y) that satisfy Equation (10.1) together with the element O. Using a different value of the pair (a, b) results in a different set E (a, b). Using this terminology, the two curves in Figure 10.4 depict the sets E (-1, 0) and E (1, 1), respectively.

GEOMETRIC DESCRIPTION OF ADDITION It can be shown that a group can be defined based on the set E (a, b) for specific values of a and b in Equation (10.1) provided the following condition is met:

$$4a^3 + 27b^2 \neq 0 \qquad \textbf{(10.2)}$$



(a) $y^2 = x^3 - x$



(b) $y^2 = x^3 + x + 1$

Figure 10.4 Example of Elliptic Curves

To define the group, we must define an operation, called addition and denoted by +, for the set E (a, b), where a and b satisfy Equation (10.2). In geometric terms, the rules for addition can be stated as follows: If three points on an elliptic curve lie on a straight line, their sum is O. From this definition, we can define the rules of addition over an elliptic curve.

1. O serves as the additive identity. Thus O = -O; for any point P on the elliptic curve, P + O = P. In what follows, we assume P! = O and Q! = O.

The negative of a point P is the point with the same x coordinate but the negative of the y coordinate; that is, if P = (x, y), then -P = (x, -y). Note that these two points can be joined by a vertical line. Note that P + (-P) = P - P = O.

1. To add two points P and Q with different x coordinates, draw a straight line between them and find the third point of intersection R. It is easily seen that there is a unique point R that is the point of intersection (unless the line is tan- gent to the curve at either P or Q, in which case we take R = P or R = Q, respectively). To form a group structure, we need to define addition on these three points: P + Q = -R. That is, we define P + Q to be the mirror image (with respect to the x axis) of the third point of intersection. Figure 10.4 illustrates this construction.

2. The geometric interpretation of the preceding item also applies to two points, P and -P, with the same x coordinate. The points are joined by a vertical line, which can be viewed as also intersecting the curve at the infinity point. We there- fore have P + (-P) = O, which is consistent with item (2).

3. To double a point Q, draw the tangent line and find the other point of inter- section S. Then Q + Q = 2Q = -S.

With the preceding list of rules, it can be shown that the set E(a, b) is an abelian group.

ALGEBRAIC DESCRIPTION OF ADDITION In this subsection, we present some results that enable calculation of additions over elliptic curves.5 For two distinct points, P = (xP , yP) and Q = (xQ, yQ), that are not negatives of each other, the slope of the line l that joins them is Δ = (yQ - yP)>(xQ - xP) . There is exactly one other point where l intersects the elliptic curve, and that is the negative of the sum of P and Q.

After some algebraic manipulation, we can express the sum R = P + Q as

$$x_R = \Delta^2 - x_P - x_Q$$
$$y_R = -y_P + \Delta(x_P - x_R) \qquad \textbf{(10.3)}$$

We also need to be able to add a point to itself: $P + P = 2P = R$. When $y_P \neq 0$, the expressions are

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P$$
$$y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P \qquad \textbf{(10.4)}$$

Elliptic Curves over Zp

Elliptic curve cryptography makes use of elliptic curves in which the variables and coefficients are all restricted to elements of a finite field. Two families of elliptic curves are used in cryptographic applications: prime curves over Zp and binary curves over GF(2m). For a prime curve over Zp, we use a cubic equation in which the variables and coefficients all take on values in the set of integers from 0 through p - 1 and in which calculations are performed modulo p. For a binary curve defined over GF(2m), the variables and coefficients all take on values in GF(2m) and in cal- culations are performed over GF(2m). [FERN99] points out that prime curves are best for software applications, because the extended bit-fiddling operations needed by binary curves are not required; and that binary curves are best for hardware applications, where it takes remarkably few logic gates to create a powerful, fast cryptosystem. We examine these two families in this

section and the next.

There is no obvious geometric interpretation of elliptic curve arithmetic over finite fields. The algebraic interpretation used for elliptic curve arithmetic over real numbers does readily carry over, and this is the approach we take.

For elliptic curves over Zp, as with real numbers, we limit ourselves to equa- tions of the form of Equation (10.1), but in this case with coefficients and variables limited to Zp:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \qquad \textbf{(10.5)}$$

For example, Equation (10.5) is satisfied for $a = 1$, $b = 1$, $x = 9$, $y = 7$, $\alpha = 1$ $p = 23$:

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3 = 3$$

Now consider the set Ep(a, b) consisting of all pairs of integers (x, y) that sat- isfy Equation (10.5), together with a point at infinity O. The coefficients a and b and the variables x and y are all elements of Zp.

For example, let p = 23 and consider the elliptic curve y2 = x3 + x + 1. In

this case, a = b = 1. Note that this equation is the same as that of Figure 10.4b. The figure shows a continuous curve with all of the real points that satisfy the equation. For the set E23(1, 1), we are only interested in the nonnegative integers in the quadrant from (0, 0) through (p - 1, p - 1) that satisfy the equation mod p. Table 10.1 lists the points (other than O) that are part of E23(1, 1). Figure 10.5 plots the points of E23(1, 1); note that the points, with one exception, are symmetric about y = 11.5.

Table 10.1 Points on the Elliptic Curve E_{23} (1,1)

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

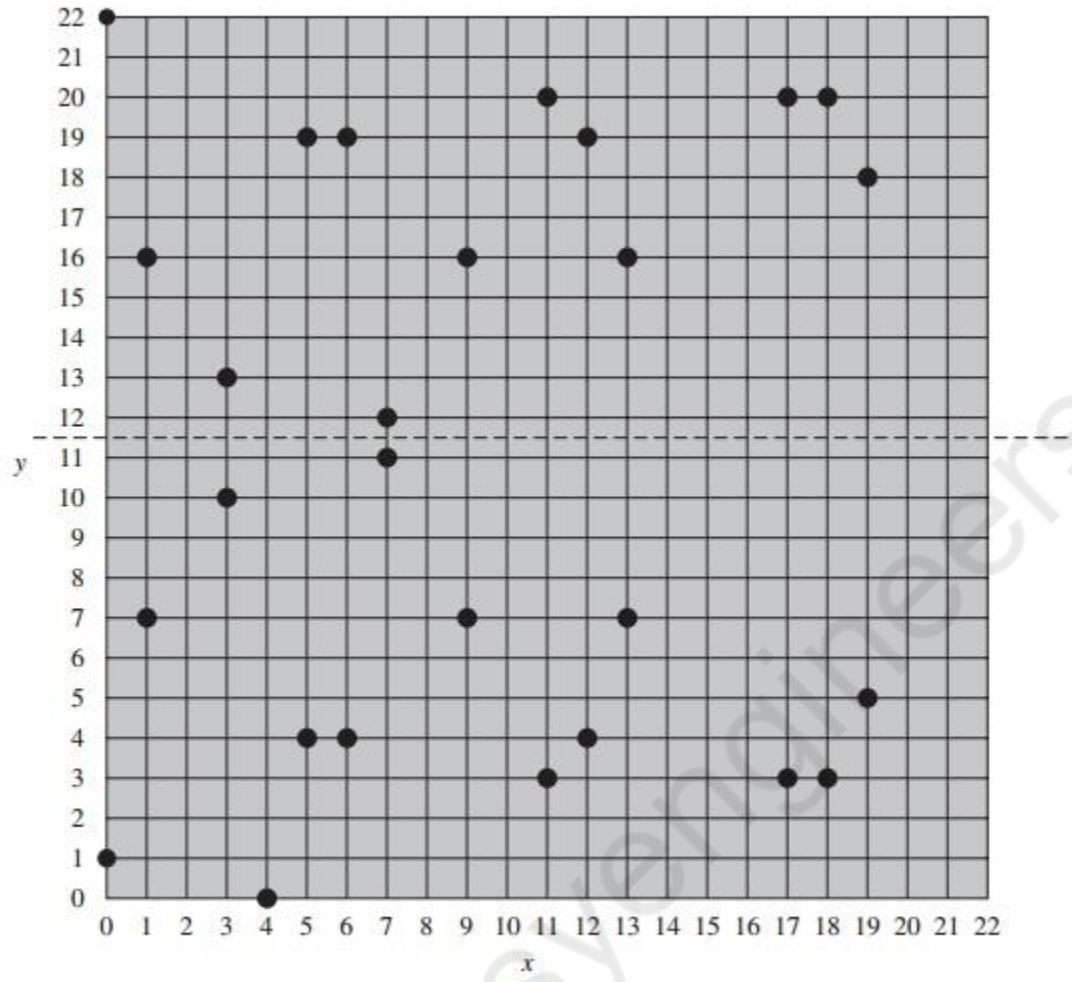


Figure 10.5 The Elliptic Curve $E_{23}(1, 1)$

It can be shown that a finite abelian group can be defined based on the set Ep (a, b) provided that (x3 + ax + b) mod p has no repeated factors. This is equivalent to the condition Note that Equation (10.6) has the same form as Equation (10.2).

The rules for addition over Ep (a, b), correspond to the algebraic technique described for elliptic curves defined over real numbers. For all points P, Q Î Ep (a, b):

1. $P + O = P$.
2. If $P = (x_P, y_P)$, then $P + (x_P, -y_P) = O$. The point $(x_P, -y_P)$ is the negative of P, denoted as $-P$. For example, in $E_{23}(1, 1)$, for $P = (13, 7)$, we have $-P = (13, -7)$. But $-7 \bmod 23 = 16$. Therefore, $-P = (13, 16)$, which is also in $E_{23}(1, 1)$.
3. If $P = (x_p, y_p)$ and $Q = (x_Q, y_Q)$ with $P \neq -Q$, then $R = P + Q = (x_R, y_R)$ is determined by the following rules:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$

$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

where

$$\lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P}\right) \bmod p & \text{if } P \neq Q \\ \left(\frac{3x_P^2 + a}{2y_P}\right) \bmod p & \text{if } P = Q \end{cases}$$

4. Multiplication is defined as repeated addition; for example, $4P = P + P + P + P$.

For example, let $P = (3, 10)$ and $Q = (9, 7)$ in $E_{23}(1, 1)$. Then

$$\lambda = \left(\frac{7 - 10}{9 - 3}\right) \bmod 23 = \left(\frac{-3}{6}\right) \bmod 23 = \left(\frac{-1}{2}\right) \bmod 23 = 11$$

$$x_R = (11^2 - 3 - 9) \bmod 23 = 109 \bmod 23 = 17$$

$$y_R = (11(3 - 17) - 10) \bmod 23 = -164 \bmod 23 = 20$$

So $P + Q = (17, 20)$. To find $2P$,

$$\lambda = \left(\frac{3(3^2) + 1}{2 \times 10}\right) \bmod 23 = \left(\frac{5}{20}\right) \bmod 23 = \left(\frac{1}{4}\right) \bmod 23 = 6$$

The last step in the preceding equation involves taking the multiplicative inverse of 4 in Z23. This can be done using the extended Euclidean algorithm defined in Section 4.4. To confirm, note that (6 * 4) mod 23 = 24 mod 23 = 1.

xR = (62 - 3 - 3) mod 23 = 30 mod 23 = 7

yR = (6(3 - 7) - 10) mod 23 = (-34) mod 23 = 12

and 2P = (7, 12).

For determining the security of various elliptic curve ciphers, it is of some inter- est to know the number of points in a finite abelian group defined over an elliptic curve. In the case of the finite group EP(a, b), the number of points N is bounded by Note that the number of points in Ep(a, b) is approximately equal to the number of elements in Zp, namely p elements.

Elliptic Curves over GF(2m)

Recall from Chapter 4 that a finite field GF(2m) consists of 2m elements, together with addition and multiplication operations that can be defined over polynomials. For elliptic curves over GF(2m), we use a cubic equation in which the variables and coefficients all take on values in GF(2m) for some number m and in which calcula- tions are performed using the rules of arithmetic in GF(2m).

It turns out that the form of cubic equation appropriate for cryptographic appli- cations for elliptic curves is somewhat different for GF(2m) than for Zp. The form is

$$y^2 + xy = x^3 + ax^2 + b \qquad \textbf{(10.7)}$$

Table 10.2 Points on the Elliptic Curve $E_{24}(g^4, 1)$

$(0, 1)$	(g^5, g^3)	(g^9, g^{13})
$(1, g^6)$	(g^5, g^{11})	(g^{10}, g)
$(1, g^{13})$	(g^6, g^8)	(g^{10}, g^8)
(g^3, g^8)	(g^6, g^{14})	$(g^{12}, 0)$
(g^3, g^{13})	(g^9, g^{10})	(g^{12}, g^{12})

where it is understood that the variables x and y and the coefficients a and b are elements of GF(2m) and that calculations are performed in GF(2m).

Now consider the set E2m(a, b) consisting of all pairs of integers (x, y) that satisfy Equation (10.7), together with a point at infinity O.

For example, let us use the finite field GF(24) with the irreducible polynomial f(x) = x4 + x + 1. This yields a generator g that satisfies f(g) = 0 with a value of g4 = g + 1 , or in binary, g = 0010. We can develop the powers of g as follows.

$g^0 = 0001$	$g^4 = 0011$	$g^8 = 0101$	$g^{12} = 1111$
$g^1 = 0010$	$g^5 = 0110$	$g^9 = 1010$	$g^{13} = 1101$
$g^2 = 0100$	$g^6 = 1100$	$g^{10} = 0111$	$g^{14} = 1001$
$g^3 = 1000$	$g^7 = 1011$	$g^{11} = 1110$	$g^{15} = 0001$

For example, $g^5 = (g^4)(g) = g^2 + g = 0110$.

Now consider the elliptic curve $y^2 + xy = x^3 + g^4x^2 + 1$. In this case, $a = g^4$ and $b = g^0 = 1$. One point that satisfies this equation is (g^5, g^3):

$$(g^3)^2 + (g^5)(g^3) = (g^5)^3 + (g^4)(g^5)^2 + 1$$
$$g^6 + g^8 = g^{15} + g^{14} + 1$$
$$1100 + 0101 = 0001 + 1001 + 0001$$
$$1001 = 1001$$

Table 10.2 lists the points (other than O) that are part of E241g4, 12 . Figure 10.6 plots the points of E241g4, 12 .It can be shown that a finite abelian group can be defined based on the set E2m(a, b), provided that b != 0. The rules for addition can be stated as follows. For all points P, Q Î E2m(a, b):

1. $P + O = P$.
2. If $P = (x_P, y_P)$, then $P + (x_P, x_P + y_P) = O$. The point $(x_P, x_P + y_P)$ is the negative of P, which is denoted as $-P$.
3. If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ with $P \neq -Q$ and $P \neq Q$, then $R = P + Q = (x_R, y_R)$ is determined by the following rules:

$$x_R = \lambda^2 + \lambda + x_P + x_Q + a$$
$$y_R = \lambda(x_P + x_R) + x_R + y_P$$

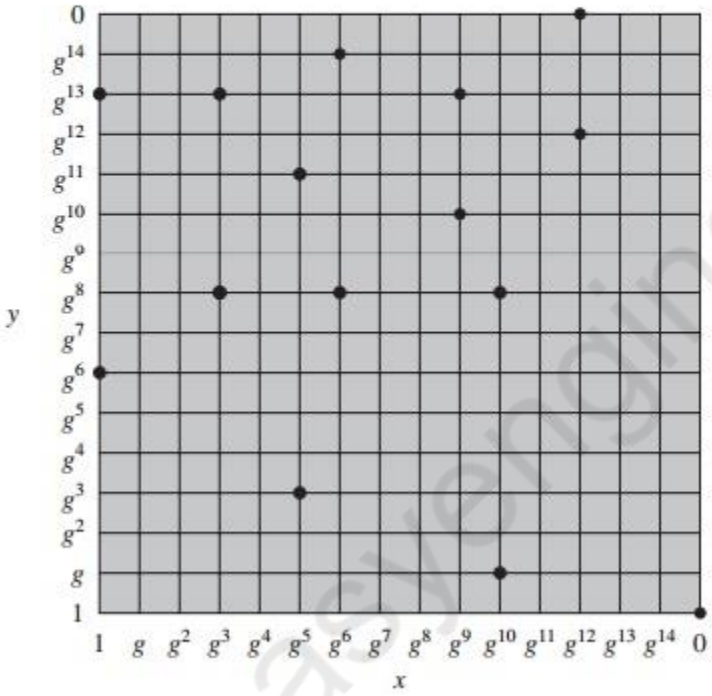


Figure 10.6 The Elliptic Curve $E_{2^4}(g^4, 1)$

where

$$\lambda = \frac{y_Q + y_P}{x_Q + x_P}$$

4. If $P = (x_P, y_P)$ then $R = 2P = (x_R, y_R)$ is determined by the following rules:

$$x_R = \lambda^2 + \lambda + a$$
$$y_R = x_P^2 + (\lambda + 1x)_R$$

where

$$\lambda = x_P + \frac{y_P}{x_P}$$

ELLIPTIC CURVE CRYPTOGRAPHY

The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiple addition is the counterpart of modular exponentiation. To form a cryptographic system using elliptic curves, we need to find a "hard problem" corresponding to factoring the product of two primes or taking the discrete logarithm.

Consider the equation Q = kP where Q, P Î EP(a, b) and k < p. It is relatively easy to calculate Q given k and P, but it is relatively hard to determine k given Q and P. This is called the discrete logarithm problem for elliptic curves.

We give an example taken from the Certicom Web site(www.certicom.com).

Consider the group E23(9,17). This is the group defined by the equation y2 mod 23 = (x3 + 9x + 17) mod 23 . What is the discrete logarithm k of Q = (4, 5) to the base P = (16, 5)? The brute-force method is to compute multiples of P until Q is found. Thus,

P = (16, 5); 2P = (20, 20); 3P = (14, 14); 4P = (19, 20); 5P = (13, 10);

6P = 17, 32; 7P = 18, 72; 8P = (12, 17); 9P = (4, 5)

Because 9P = (4, 5) = Q, the discrete logarithm Q = (4, 5) to the base P = (16, 5) is k = 9. In a real application, k would be so large as to make the brute- force approach infeasible.

In the remainder of this section, we show two approaches to ECC that give the flavor of this technique.

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 2m, and elliptic curve parameters a and b for Equation (10.5) or Equation (10.7). This defines the elliptic group of points Eq(a, b). Next, pick a base point G = (x1, y1) in Ep(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.

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

1. A selects an integer nA less than n. This is A's private key. A then generates a public key PA = nA * G; the public key is a point in Eq(a, b).

2. B similarly selects a private key nB and computes a public key PB.

3. A generates the secret key k = nA * PB. B generates the secret key k = nB * PA.

The two calculations in step 3 produce the same result because

nA * PB = nA * (nB * G) = nB * (nA * G) = nB * PA

To break this scheme, an attacker would need to be able to compute k given Gand kG, which is assumed to be hard.

As an example,6 take p = 211; Ep(0, - 4), which is equivalent to the curve

y2 = x3 - 4; and G = (2, 2). One can calculate that 240G = O. A's private key is

nA = 121, so A's public key is PA = 121(2, 2) = (115, 48). B's private key

is nB = 203, so B's public key is 203(2, 3) = (130, 203). The shared secret key is

121(130, 203) = 203(115, 48) = (161, 69).

Note that the secret key is a pair of numbers. If this key is to be used as a session key for conventional encryption, then a single number must be generated. We could simply use the x coordinates or some simple function of the x coordinate.

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

Elliptic Curve Encryption/Decryption

Several approaches to encryption/decryption using elliptic curves have been analyzed in the literature. In this subsection, we look at perhaps the simplest. The first task in this system is to encode the plaintext message m to be sent as an x–y point Pm. It is the point Pm that will be encrypted as a cipher text and subsequently decrypted. Note that we cannot simply encode the message as the x or y coordinate of a point, because not all such coordinates are in Eq(a, b); for example, see Table

Again, there are several approaches to this encoding, which we will not address here, but suffice it to say that there are relatively straightforward tech- niques that can be used.

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 nA and generates a public key PA = nA * G.

To encrypt and send a message Pm to B, A chooses a random positive integer k and produces the ciphertext Cm consisting of the pair of points:

Cm = {kG, Pm + kPB}

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

Pm + kPB - nB(kG) = Pm + k(nBG) - nB(kG) = Pm

A has masked the message Pm by adding kPB to it. Nobody but A knows the value of k, so even though Pb is a public key, nobody can remove the mask kPB. However, A also includes a "clue," which is enough to remove the mask if one knows the private key nB. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed to be hard.

As an example of the encryption process (taken from , take p = 751; Ep(-1, 188) , which is equivalent to the curve y2 = x3 - x + 188; and G = (0, 376). Suppose that A wishes to send a message to B that is encoded in the elliptic point Pm = (562, 201) and that A selects the random number k = 386. B's public key is PB = (201, 5). We have 386(0, 376) = (676, 558), and (562, 201) + 386(201, 5) = (385, 328). Thus, A sends the cipher text {(676, 558), (385, 328)}.

Security of Elliptic Curve Cryptography

The security of ECC depends on how difficult it is to determine k given kP and P. This is referred to as the elliptic curve logarithm problem. The fastest known tech- nique for taking the elliptic curve logarithm is known as the Pollard rho method. Table 10.3 compares various algorithms by showing comparable key sizes in terms of computational effort for cryptanalysis. As can be seen, a considerably smaller key size can be used for ECC compared to RSA. Furthermore, for equal key lengths, the computational effort required for ECC and RSA is comparable [JURI97]. Thus, there is a computational advantage to using ECC with a shorter key length than a comparably secure RSA.

Table 10.3 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis

Table 10.3 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis

Symmetric Scheme (key size in bits)	ECC-Based Scheme (size of *n* in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Source: Certicom

PSEUDORANDOM NUMBER GENERATION BASED ON AN ASYMMETRIC CIPHER

We noted in Chapter 7 that because a symmetric block cipher produces an apparently random output, it can serve as the basis of a pseudorandom number generator (PRNG). Similarly, an asymmetric encryption algorithm produces apparently random output and can be used to build a PRNG. Because asymmetric algorithms are typically much slower than symmetric algorithms, asymmetric algorithms are not used to generate open-ended PRNG bit streams. Rather, the asymmetric approach is useful for creating a pseudorandom function (PRF) for generating a short pseudorandom bit sequence.

we examine two PRNG designs based on pseudorandom functions.

PRNG Based on RSA

For a sufficient key length, the RSA algorithm is considered secure and is a good candidate to form the basis of a PRNG. Such a PRNG, known as the Micali-Schnorr PRNG [MICA91], is recommended in the ANSI standard X9.82 (Random Number Generation) and in the ISO standard 18031 (Random Bit Generation).

The PRNG is illustrated in Figure 10.8. As can be seen, this PRNG has much the same structure as the output feedback (OFB) mode used as a PRNG (see Figure 7.3b and the portion of Figure 6.6a enclosed with a dashed box). In this case, the encryption algorithm is RSA rather than a symmetric block cipher. Also, a portion of the output is fed back to the next iteration of the encryption algorithm and the remain- der of the output is used as pseudorandom bits. The motivation for this separation of the output into two distinct parts is so that the pseudorandom bits from one stage do not provide input to the next stage. This separation should contribute to forward unpredictability.

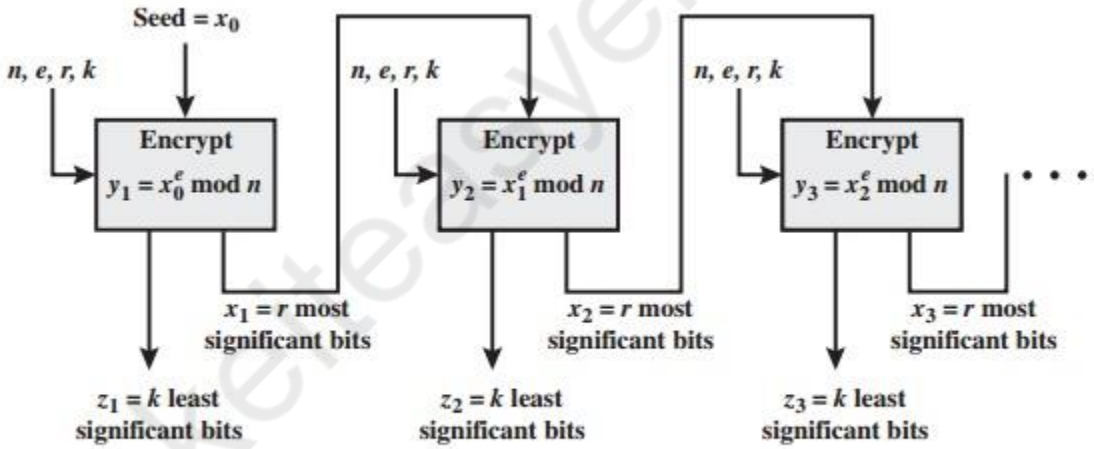


Figure 10.8 Micali-Schnorr Pseudorandom Bit Generator

Setup Select p, q primes; $n = pq$; $\phi(n) = (p - 1)(q - 1)$. Select e such that $\gcd(e, \phi(n)) = 1$. These are the standard RSA setup selections (see Figure 9.5). In addition, let $N = \lfloor \log_2 n \rfloor + 1$ (the bitlength of n). Select r, k such that $r + k = N$.

Seed Select a random seed x_0 of bitlength r.

Generate Generate a pseudorandom sequence of length $k \times m$ using the loop

for i **from** 1 to m **do**

$$y_i = x_{i-1}^e \bmod n$$
$$x_i = r \text{ most significant bits of } y_i$$
$$z_i = k \text{ least significant bits of } y_i$$

Output The output sequence is $z_1 \| z_2 \| \ldots \| z_m$.

The parameters n, r, e, and k are selected to satisfy the following six requirements.

1. $n = pq$	n is chosen as the product of two primes to have the cryptographic strength required of RSA.
2. $1 < e < \phi(n)$; $\gcd(e, \phi(n)) = 1$	Ensures that the mapping $s \rightarrow s^e \bmod n$ is 1 to 1.
3. $re \geq 2N$	Ensures that the exponentiation requires a full modular reduction.
4. $r \geq 2\,strength$	Protects against a cryptographic attacks.
5. k, r are multiples of 8	An implementation convenience.
6. $k \geq 8$; $r + k = N$	All bits are used.

The variable strength in requirement 4 is defined in NIST SP 800-90 as follows: A number associated with the amount of work (that is, the number of operations) required to break a cryptographic algorithm or system; a security strength is specified in bits and is a specific value from the set (112, 128, 192, 256) for this Recommendation. The amount of work needed is 2strength.

There is clearly a tradeoff between r and k. Because RSA is computation- ally intensive compared to a block cipher, we would like to generate as many pseudorandom bits per iteration as possible and therefore would like a large value of k. However, for cryptographic strength, we would like r to be as large as possible.

For example, if e = 3 and N = 1024, then we have the inequality 3r 7 1024, yielding a minimum required size for r of 683 bits. For r set to that size, k = 341 bits are generated for each exponentiation (each RSA encryption). In this case, each exponentiation requires only one modular squaring of a 683-bit number and one modular multiplication. That is, we need only calculate 1xi * 1xi mod n22mod n .

PRNG Based on Elliptic Curve Cryptography

This technique is recommended in NIST SP 800-90, the ANSI standard X9.82, and the ISO standard 18031. There has been some controversy regarding both the security and efficiency of this algorithm compared to other alternatives summarizes the algorithm as follows: Let P and Q be two known points on a given elliptic curve. The seed of the DEC PRNG is a random integer s0 Î {0, 1... #E (GF (p)) - 1}, where # E (GF (p)) denotes the number of points on the curve. Let x denote a function that gives the x-coordinate of a point of the curve. Let lsbi1s2 denote the i least significant bits of an integer s. The DEC PRNG trans- forms the seed into the pseudorandom sequence of length 240k, k > 0, as follows.

```
for i = 1 to k do
   Set s_i ← x(s_{i-1} P)
   Set r_i ← lsb_240 (x(s_i Q))
end for
   Return r_1, . . . , r_k
```

Given the security concerns expressed for this PRNG, the only motivation for its use would be that it is used in a system that already implements ECC but does not implement any other symmetric, asymmetric, or hash cryptographic algorithm that could be used to build a PRNG.

Key management and Distribution Symmetric Key Distribution Using Symmetric Encryption

For **symmetric** encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Therefore, the term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key. For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

Physical delivery (1 & 2) is simplest - but only applicable when there is personal contact between recipient and key issuer. This is fine for link encryption where devices & keys occur in pairs, but does not scale as number of parties who wish to communicate grows. 3 is mostly based on 1 or 2 occurring first.

A third party, whom all parties trust, can be used as a **trusted intermediary** to mediate the establishment of secure communications between them (4). Must trust intermediary not to abuse the knowledge of all session keys. As number of parties grow, some variant of 4 is only practical solution to the huge growth in number of keys potentially needed.

Key distribution center:

1 The use of a **key distribution center** is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used.
2 Communication between end systems is encrypted using a temporary key, often referred to as a **Session key**.
3 Typically, the session key is used for the duration of a logical connection and then discarded
4 **Master key** is shared by the key distribution center and an end system or user and used to encrypt the session key.

Key Distribution Scenario:

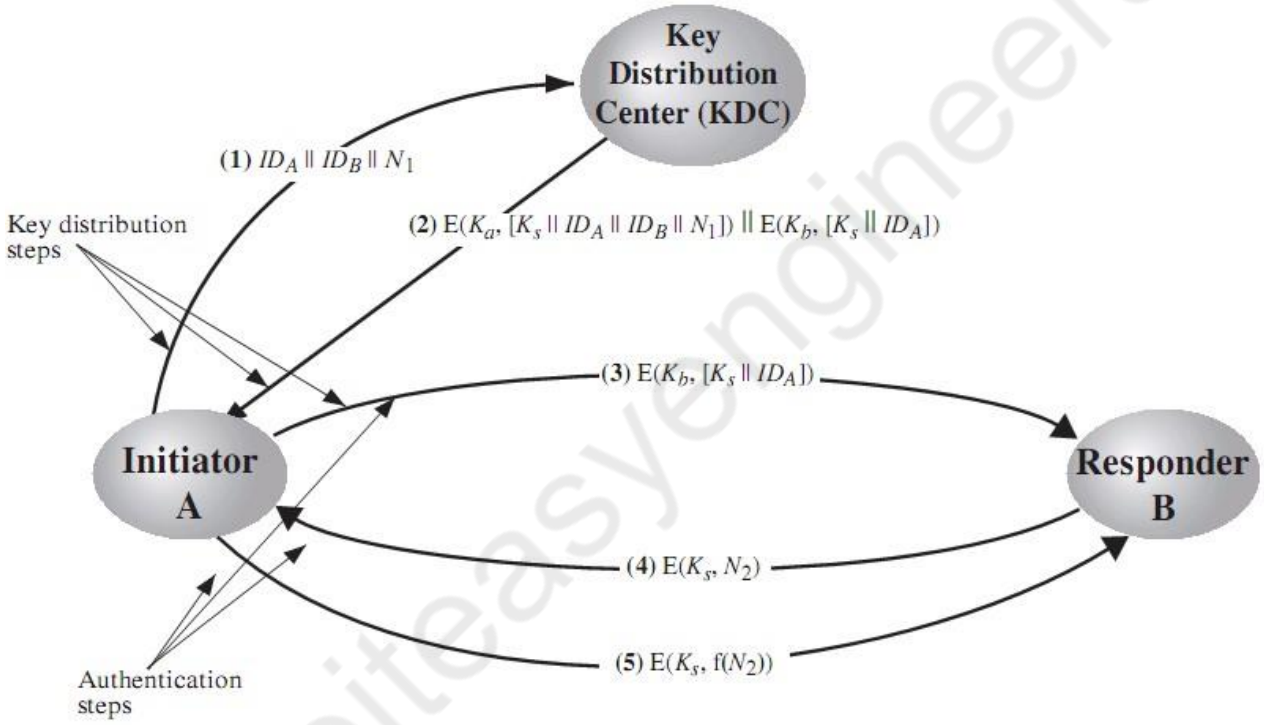


Figure 14.3 Key Distribution Scenario

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

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1, for this transaction, which we refer to as a **nonce**. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is **that it differs with each request.** Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

2. The KDC responds with a message encrypted using Ka Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:

- The **one-time session key, Ks,** to be used for the session
- The **original request message,** including the nonce, to enable A to match this response with the appropriate request

Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request. In addition, the message includes two items intended for B:

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

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

- A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, $E(K_b, [K_s \| ID_A])$. Because this information is encrypted with K_b, it is protected from eavesdropping. B now knows the session key (K_s), knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b). At this point, a session key has been securely delivered to A and B, and they may begin

their protected exchange. However, two additional steps are desirable:

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

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

These steps assure B that the original message it received (step 3) was not a replay.

Note that the actual key distribution involves only steps 1 through 3 but that steps 4 and 5, as well as 3, perform an authentication function.

Major Issues with KDC:

Hierarchical Key Control

1. It is not **necessary to limit** the key distribution function to a single KDC.Indeed, for very large networks, it may not be practical to do so.As an alternative, a hierarchy of KDCs can be established.
2. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building.
3 If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC.
4 The hierarchical concept can be extended to three or even more layers, depending on the size of the user population and the geographic scope of the internetwork.
5 A hierarchical scheme **minimizes the effort involved in master key distribution**, because most master keys are those shared by a local KDC with its local entities.

Session Key Lifetime

The distribution of session keys delays the start of any exchange and places a burden on network capacity. A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.

6 For **connection-oriented protocols**, one obvious choice is to use the same session key for the length of time that the connection is open, using a new session key for each new session.

7 If a logical connection has a very long lifetime, then it would be prudent to change the session key periodically, perhaps every time the PDU (protocol data unit) sequence number cycles.

8 For a **connectionless protocol**, such as a transaction-oriented protocol, there is no explicit connection initiation or termination.

9 Thus, it is not obvious how often one needs to change the session key. The most secure approach is to use a **new session key for each exchange**.

10 A better strategy is to use a given **session key for a certain fixed period only or for a certain number of transactions.**

A Transparent Key Control Scheme

11 The approach suggested in Figure 14.3is useful for providing end-to-end encryption at a network or transport level in a way that is transparent to the end users.

12 The approach assumes that communication makes use of a connection-oriented end-to- end protocol, such as TCP.

13 The noteworthy element of this approach is a session security module (SSM), which may consist of functionality at one protocol layer,that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

The steps involved in establishing a connection are shown in Figure 14.4.

1. When one host wishes to set up a connection to another host, it transmits a connection-request packet.
2. The SSM saves that packet and applies to the KDC for permission to establish the connection.
3. The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC.If the KDC approves the

connection request,it generates the session key and delivers it to the two appropriate SSMs,using a unique permanent key for each SSM.

4. The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems.
5. All user data exchanged between the two end systems are encrypted by their respective SSMs using the onetime session key.

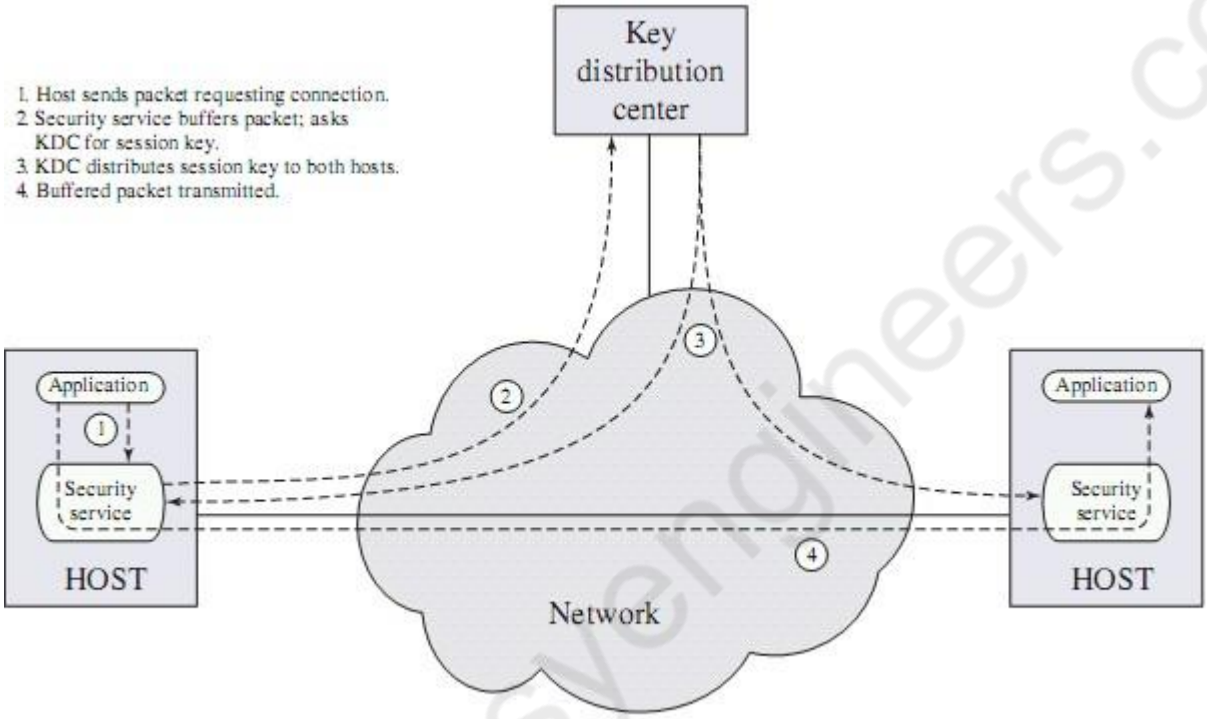


Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

a. The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other.

Decentralized Key Control

b. The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized.

c. Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.

d. A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution.

e. Thus, there may need to be as many as $n(n-1)/2$ master keys for a configuration with n end systems.

f. A session key may be established with the following sequence of steps (Figure 14.5).

1. A issues a request to B for a session key and includes a nonce, .

2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B,an identifier of B,the value $f(N_1)$, and another nonce N_2.

3. Using the new session key,A returns $f(N_2)$ to B.

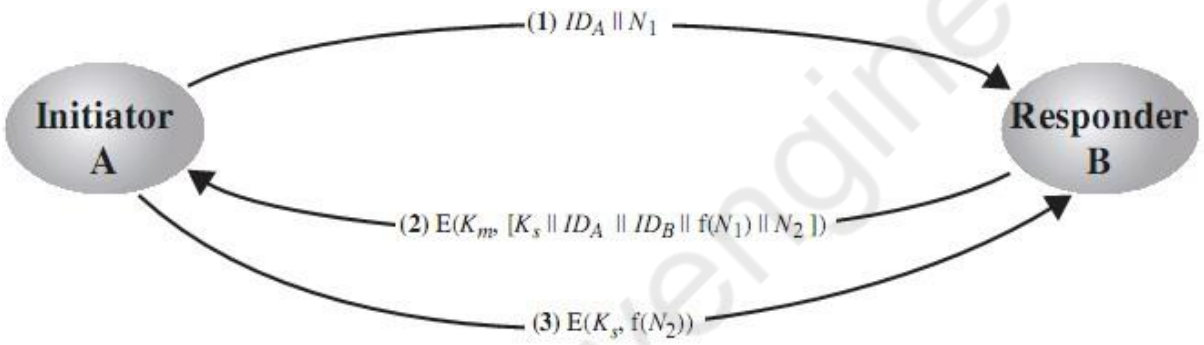


Figure 14.5 Decentralized Key Distribution

Controlling Key Usage

The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed. It also may be desirable to impose some control on the way in which automatically distributed keys are used. For example, in addition to separating master keys from session keys, we may wish to define different types of session keys on the basis of use, such as

- Data-encrypting key, for general communication across a network
- PIN-encrypting key, for personal identification numbers (PINs) used in electronic funds transfer and point-of-sale applications
- File-encrypting key, for encrypting files stored in publicly accessible locations

To illustrate the value of separating keys by type, consider the risk that a master key is imported as a data-encrypting key into a device. Normally, the master key is physically secured within the cryptographic hardware of the key distribution center and of the end systems. Session keys encrypted with this master key are available to application programs, as are the data encrypted with such session keys.

However, if a master key is treated as a session key, it may be possible for an unauthorized application to obtain plaintext of session keys encrypted with that master key.

The proposed technique is for use with DES and makes use of the extra 8 bits in each 64-bit DES key. That is, the eight non-key bits ordinarily reserved for parity checking form the key tag. The bits have the following interpretation:

- One bit indicates whether the key is a session key or a master key.
- One bit indicates whether the key can be used for encryption.
- One bit indicates whether the key can be used for decryption.
- The remaining bits are spares for future use.

Because the tag is embedded in the key, it is encrypted along with the key when that key is distributed, thus providing protection. The drawbacks of this scheme are

1. The tag length is limited to 8 bits, limiting its flexibility and functionality.
2. Because the tag is not transmitted in clear form, it can be used only at the point of decryption, limiting the ways in which key use can be controlled.

A more flexible scheme, referred to as the control vector, is described here. In this scheme, each session key has an associated control vector consisting of a number of fields

that specify the uses and restrictions for that session key. The length of the control vector may vary.

The control vector is cryptographically coupled with the key at the time of key generation at the KDC.

As a first step, the control vector is passed through a hash function that produces a value whose length is equal to the encryption key length. In essence, a hash function maps values from a larger range into a smaller range with a reasonably uniform spread. Thus, for example, if numbers in the range 1 to 100 are hashed into numbers in the range 1 to 10, approximately 10% of the source values should map into each of the target values. The hash value is then XORed with the master key to produce an output that
is used as the key input for encrypting the session key. Thus,

Hash value = H = h(CV) Key
input = Km ⊕H
Ciphertext = E([Km ⊕H], Ks)

where is the master key and is the session key. The session key is recovered in plaintext by the reverse operation:

D([Km⊕H], E([Km ⊕H], Ks))

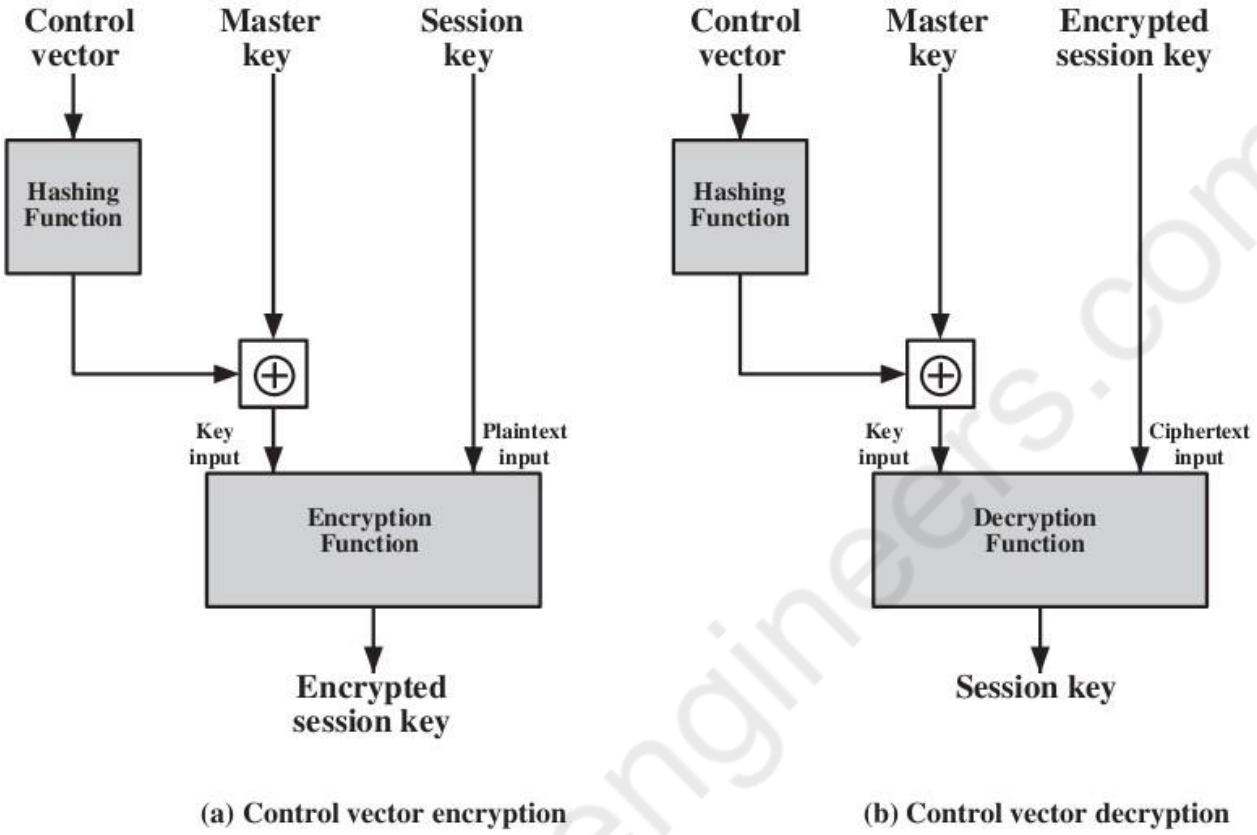


Figure 14.6 Control Vector Encryption and Decryption

When a session key is delivered to a user from the KDC, it is accompanied by the control vector in clear form. The session key can be recovered only by using both the master key that the user shares with the KDC and the control vector. Thus, the linkage between the session key and its control vector is maintained.

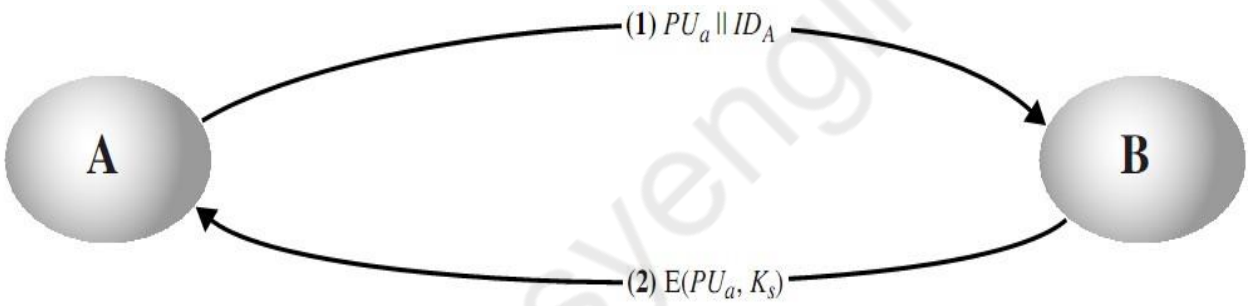
Use of the control vector has two **advantages over use of an 8-bit tag.** First, there is no restriction on length of the control vector, which enables arbitrarily complex controls to be imposed on key use. Second, the control vector is available in clear form at all stages of operation. Thus, control of key use can be exercised in multiple locations.

14.2 SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION

14 Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering, or both, is possible.
15 Public-key encryption provides for the distribution of secret keys to be used for conventional encryption.

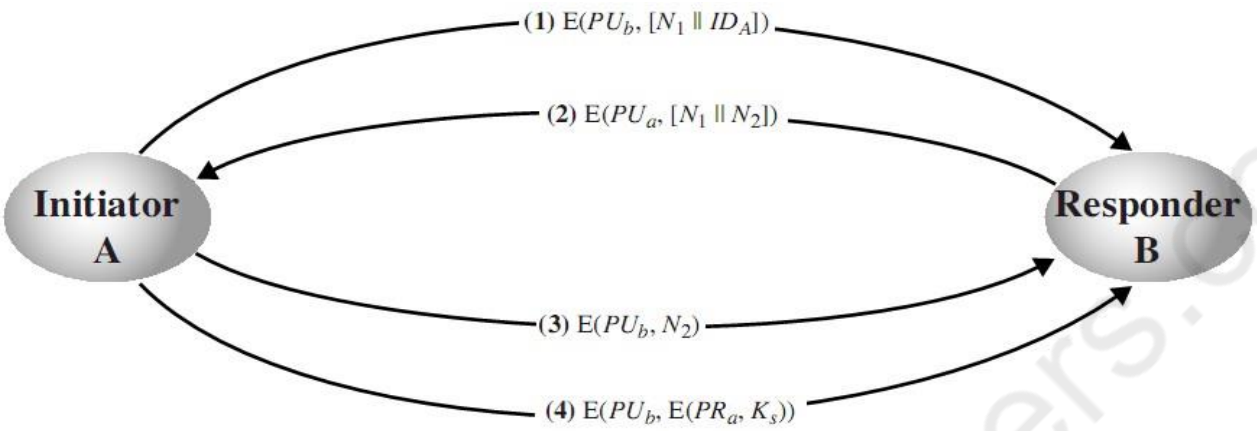
Simple Secret Key Distribution

- A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PUa and an identifier of A, IDA
- B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.
- A computes D(PRa, E(PUa, Ks)) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks.
- A discards PUa and PRa and B discards PUa.



Here third party can intercept messages and then either relay the intercepted message or substitute another message Such an attack is known as a **man-in-the-middle attack.**

Secret Key Distribution with Confidentiality and Authentication:



16 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

17 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

18 A returns N_2 encrypted using B's public key, to assure B that its correspondent is A.

19 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.

20 B computes D (PU_a, D(PR_b, M)) to recover the secret key.

A Hybrid Scheme:

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach.

- 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 addition of a public-key layer provides a secure, efficient means of distributing master keys.

Distribution of Public Keys:

Several techniques have been proposed for the distribution of public keys, which can mostly be grouped into the categories shown.

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

Public Announcement of Public Keys

The point of public-key encryption is that the public key is public, hence any participant can send his or her public key to any other participant, or broadcast the key to the community at large. eg. append PGP keys to email messages or post to news groups or email list

Figure 10.1 Uncontrolled Public Key Distribution

Its major weakness is forgery; anyone could pretend to be user A and send a public key to another participant or broadcast such a public key. Until the forgery is discovered they can masquerade as the claimed user.

Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
  - The authority maintains a directory with a {name, public key} entry for each participant.
  - Each participant registers a public key with the directory authority.
  - A participant may replace the existing key with a new one at any time 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.

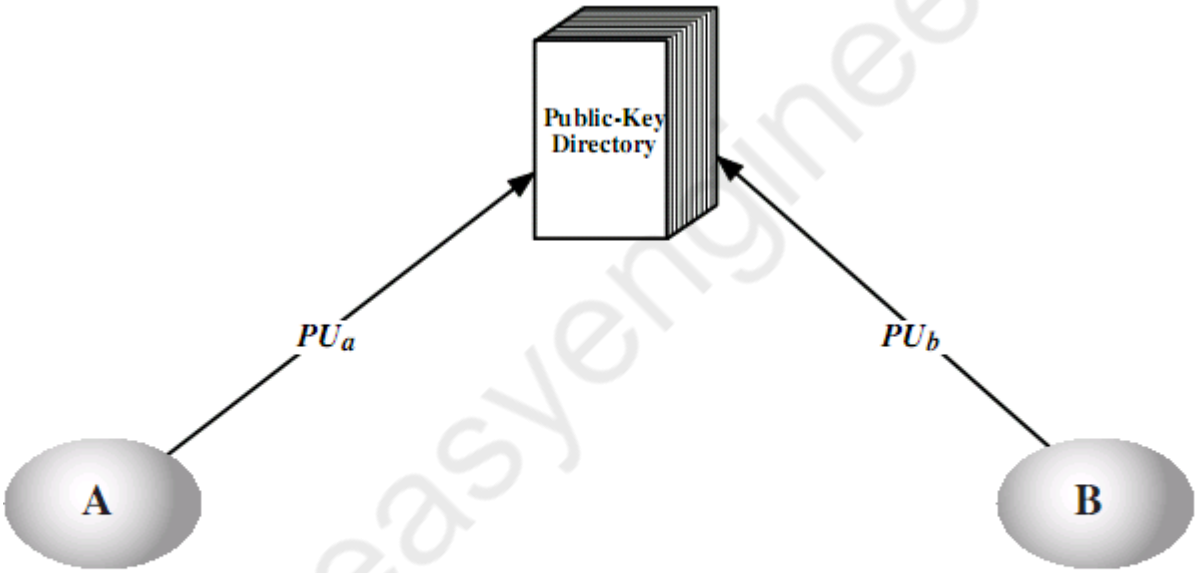


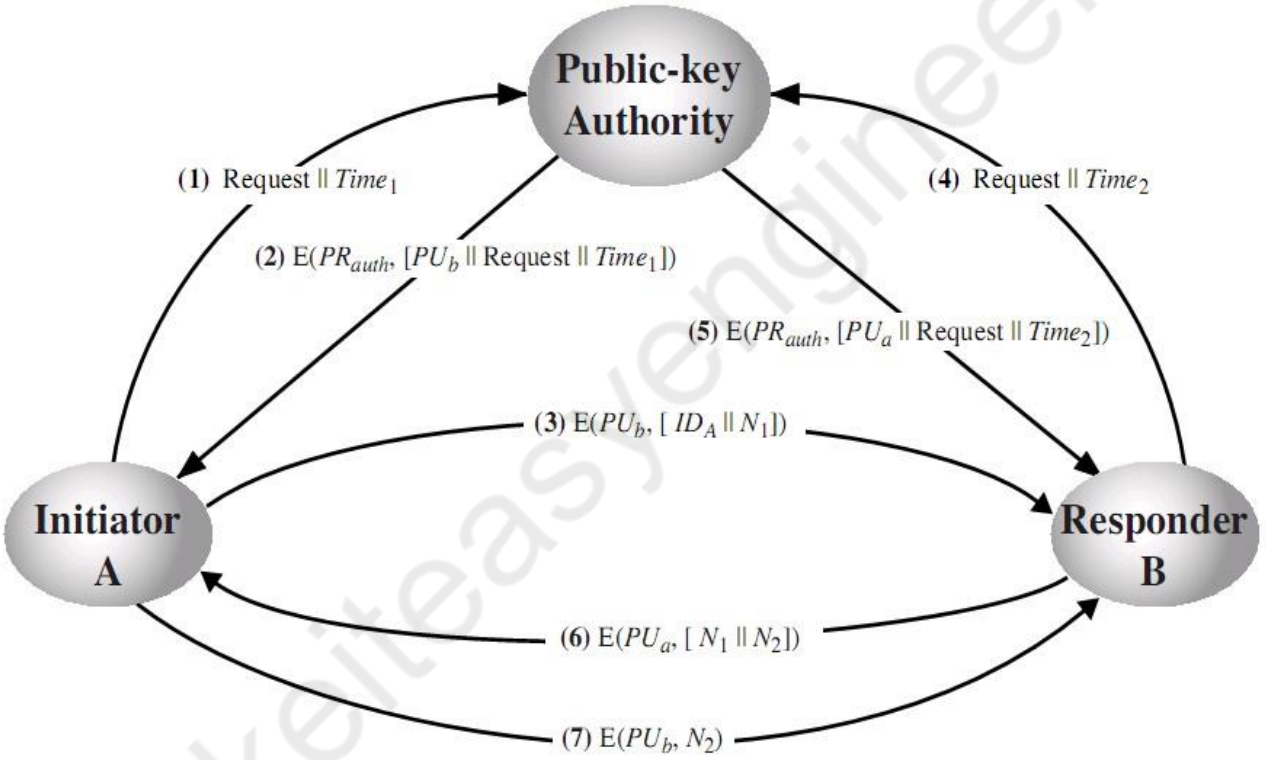
Figure 10.2 Public Key Publication

This scheme is clearly more secure than individual public announcements but still has vulnerabilities.

If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

Public-Key Authority:

21 Stronger security for public-key distribution can be achieved by providing tightercontrol over the distribution of public keys from the directory.
22 It requires users to know the public key for the directory, and that they interact with directory in real-time to obtain any desired public key securely.
23 Totally seven messages are required.



1. A sends a time stamped 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

24 A user must appeal to the authority for a public key for every other user that it wishes to contact and it is vulnerable to tampering too.
25 Public key certificates can be used to exchange keys without contacting a public-key authority.
26 A certificate binds an **identity** to **public key**, with all contents **signed** by a trusted Public- Key or Certificate Authority (CA).
27 This can be verified by anyone who knows the public-key authorities public-key.

A participant can also convey its key information to another by transmitting its certificate.

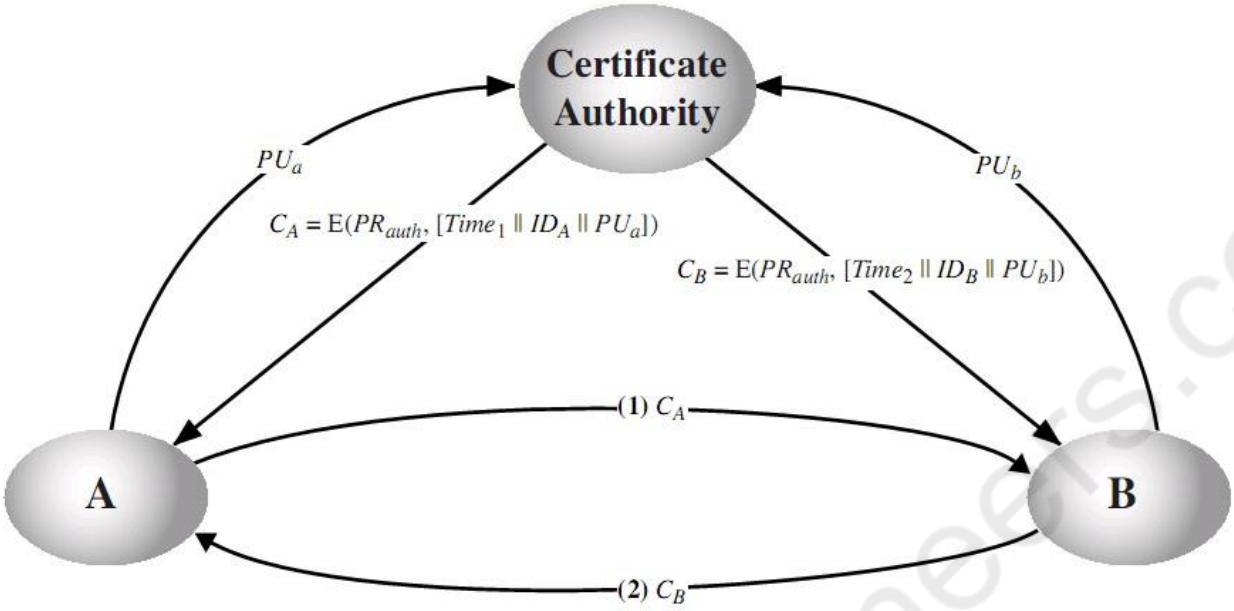
Other participants can verify that the certificate was created by the authority. We can place the following requirements on this scheme:

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.

One scheme has become universally accepted for formatting public-key certificates: the X.509 standard.

X.509 certificates are used in most network security applications, including IP security, secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME.



****