Chapter 10

Symmetric-Key Cryptography

Objectives

- To distinguish between two cryptosystems: symmetric-key and asymmetric-key
- To introduce trapdoor one-way functions and their use in asymmetric-key cryptosystems
- To introduce the knapsack cryptosystem as one of the first ideas in asymmetric-key cryptography
- To discuss the RSA cryptosystem
- To discuss the Rabin cryptosystem
- To discuss the ElGamal cryptosystem
- To discuss the elliptic curve cryptosystem

10-1 INTRODUCTION

Symmetric and asymmetric-key cryptography will exist in parallel and continue to serve the community. We actually believe that they are complements of each other; the advantages of one can compensate for the disadvantages of the other.

Topics discussed in this section:

10.1.1 Keys
10.1.2 General Idea
10.1.3 Need for Both
10.1.4 Trapdoor One-Way Function
10.1.5 Knapsack Cryptosystem

10-1 INTRODUCTION

Symmetric and asymmetric-key cryptography will exist in parallel and continue to serve the community. We actually believe that they are complements of each other; the advantages of one can compensate for the disadvantages of the other.

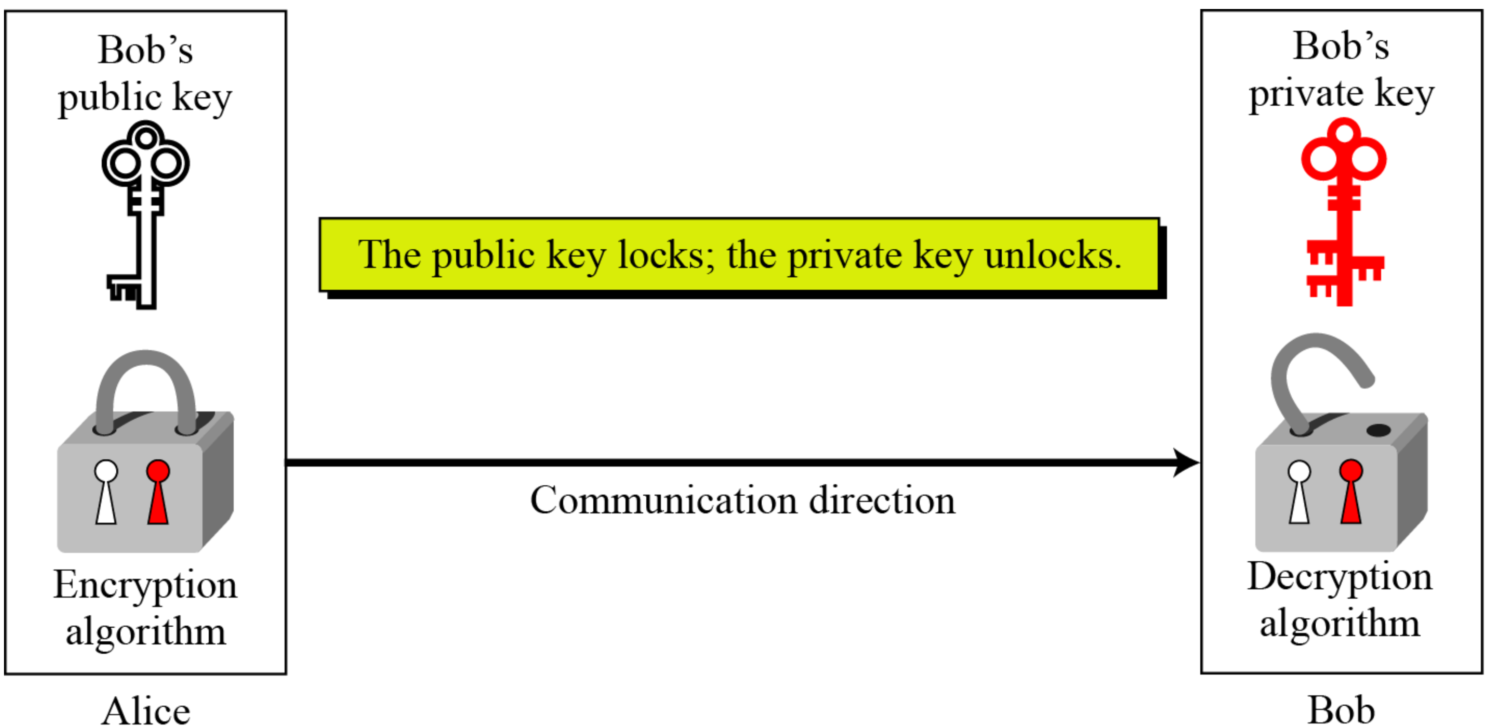
Note

Symmetric-key cryptography is based on sharing secrecy; asymmetric-key cryptography is based on personal secrecy.

10.1.1 Keys

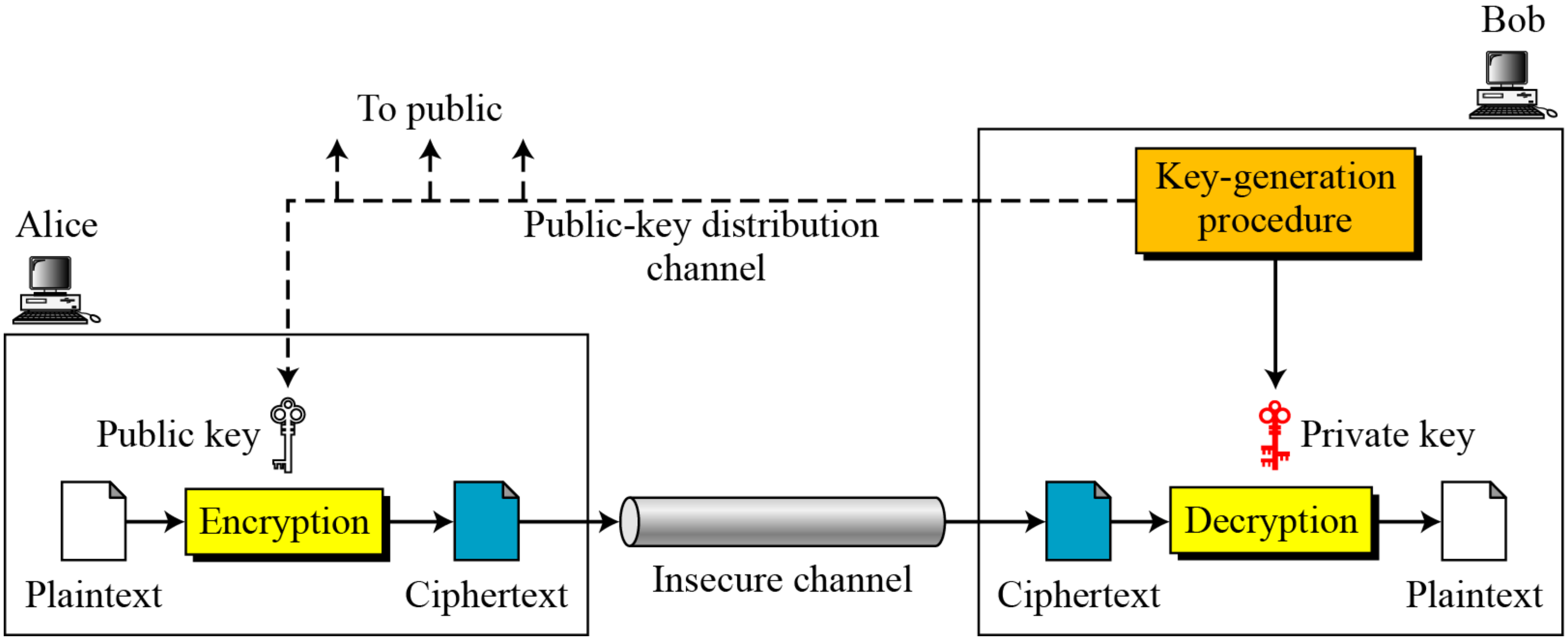
Asymmetric key cryptography uses two separate keys: one private and one public.

Figure 10.1 ***Locking and unlocking in asymmetric-key cryptosystem***



10.1.2 General Idea

Figure 10.2 *General idea of asymmetric-key cryptosystem*



10.1.2 Continued

Plaintext/Ciphertext

Unlike in symmetric-key cryptography, plaintext and ciphertext are treated as integers in asymmetric-key cryptography.

Encryption/Decryption

$$\boxed{C = f\,(K_{public}\,,\, P) \qquad P = g(K_{private}\,,\, C)}$$

10.1.3 Need for Both

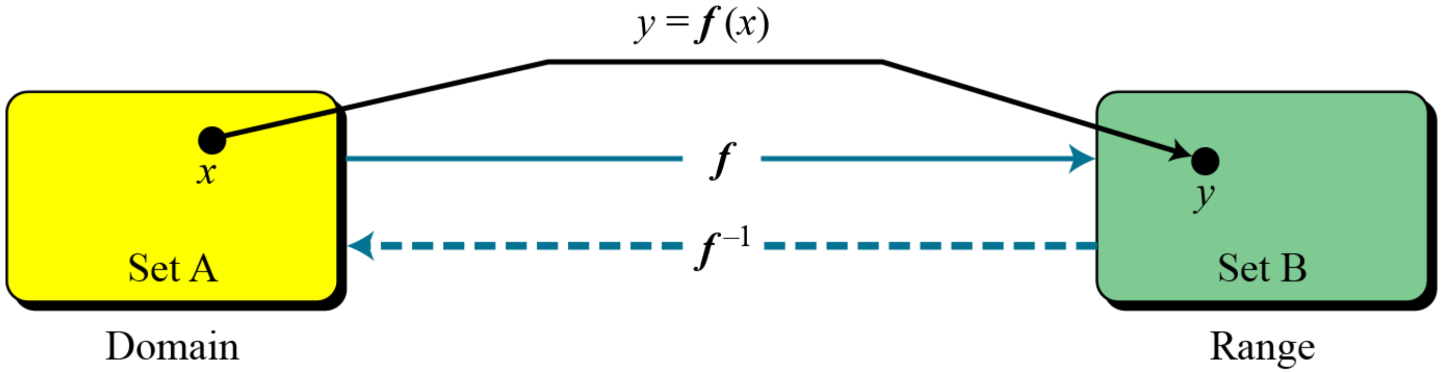
There is a very important fact that is sometimes misunderstood: The advent of asymmetric-key cryptography does not eliminate the need for symmetric-key cryptography.

10.1.4 Trapdoor One-Way Function

The main idea behind asymmetric-key cryptography is the concept of the trapdoor one-way function.

Functions

Figure 10.3 ***A function as rule mapping a domain to a range***



10.1.4 Continued

One-Way Function (OWF)

1. f is easy to compute.

2. f^{-1} is difficult to compute.

Trapdoor One-Way Function (TOWF)

3. Given y and a trapdoor, x can be computed easily.

10.1.4 Continued

Example 10. 1

When n is large, $n = p \times q$ is a one-way function. Given p and q , it is always easy to calculate n ; given n, it is very difficult to compute p and q. This is the factorization problem.

Example 10. 2

When n is large, the function $y = x^k \bmod n$ is a trapdoor one-way function. Given x, k, and n, it is easy to calculate y. Given y, k, and n, it is very difficult to calculate x. This is the discrete logarithm problem. However, if we know the trapdoor, k′ such that $k \times k' = 1 \bmod \phi(n)$, we can use $x = y^{k'} \bmod n$ to find x.

10-2 RSA CRYPTOSYSTEM

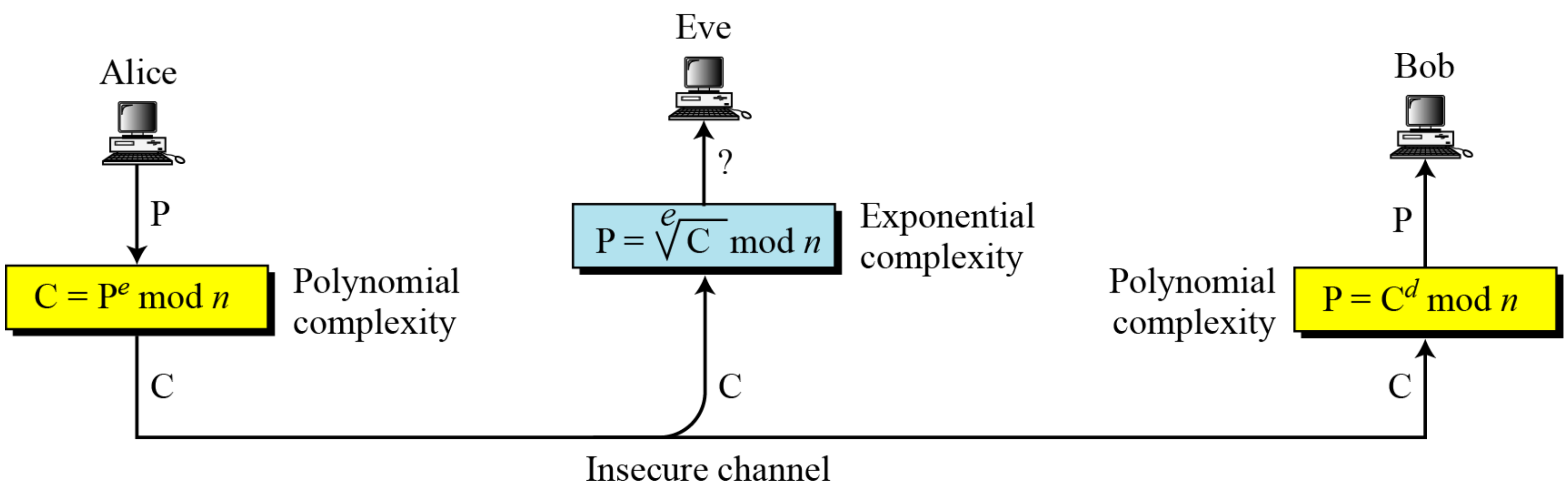
The most common public-key algorithm is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman).

Topics discussed in this section:

10.2.1 Introduction
10.2.2 Procedure
10.2.3 Some Trivial Examples
10.2.4 Attacks on RSA
10.2.5 Recommendations
10.2.6 Optimal Asymmetric Encryption Padding (OAEP)
10.2.7 Applications

10.2.1 Introduction

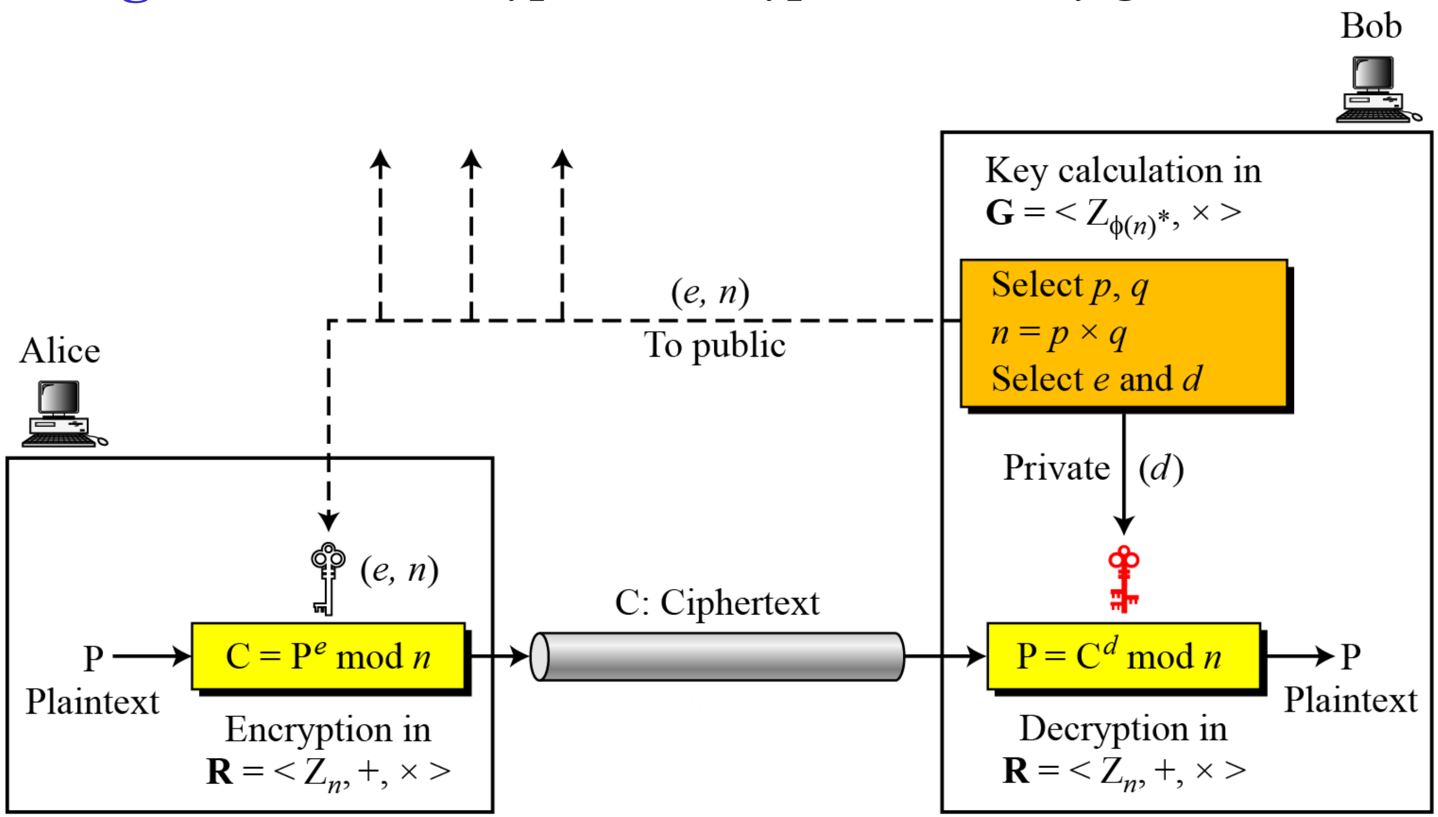
Figure 10.5 *Complexity of operations in RSA*



RSA uses modular exponentiation for encryption/decryption; To attack it, Eve needs to calculate $\sqrt[e]{C} \bmod n$.

10.2.2 Procedure

Figure 10.6 *Encryption, decryption, and key generation in RSA*



10.2.2 Continued

Two Algebraic Structures

Encryption/Decryption Ring: $R = \langle Z_n, +, \times \rangle$

Key-Generation Group: $G = \langle Z_{\phi(n)}^*, \times \rangle$

RSA uses two algebraic structures:
a public ring $R = \langle Z_n, +, \times \rangle$ and a private group $G = \langle Z_{\phi(n)}^*, \times \rangle$.

In RSA, the tuple (e, n) is the public key; the integer d is the private key.

Algorithm 10.2 *RSA Key Generation*

RSA_Key_Generation

{

Select two large primes p and q such that $p \neq q$.

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p - 1) \times (q - 1)$

Select e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$ // d is inverse of e modulo $\phi(n)$

Public_key $\leftarrow$ (e, n) // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

10.2.2 Continued

Encryption

Algorithm 10.3 *RSA encryption*

```
RSA_Encryption (P, e, n)                    // P is the plaintext in Z_n and P < n
{
    C ← Fast_Exponentiation (P, e, n)       // Calculation of (P^e mod n)
    return C
}
```

In RSA, p and q must be at least 512 bits; n must be at least 1024 bits.

10.2.2 Continued

Decryption

Algorithm 10.4 *RSA decryption*

```
RSA_Decryption (C, d, n)                      //C is the ciphertext in Zn
{
    P ← Fast_Exponentiation (C, d, n)         // Calculation of (C^d mod n)
    return P
}
```

10.2.2 Continued

Proof of RSA

If $n = p \times q$, $a < n$, and k is an integer, then $a^{k \times \phi(n)+1} \equiv a \pmod{n}$.

$P_1 = C^d \bmod n = (P^e \bmod n)^d \bmod n = P^{ed} \bmod n$

$ed = k\phi(n) + 1$ // d and e are inverses modulo $\phi(n)$

$P_1 = P^{ed} \bmod n \rightarrow P_1 = P^{k\phi(n)+1} \bmod n$

$P_1 = P^{k\phi(n)+1} \bmod n = P \bmod n$ // Euler's theorem (second version)

10.2.3 Some Trivial Examples

Example 10. 5

Bob chooses 7 and 11 as p and q and calculates $n = 77$. The value of $\phi(n) = (7 - 1)(11 - 1)$ or 60. Now he chooses two exponents, e and d, from $Z_{60}*$. If he chooses e to be 13, then d is 37. Note that $e \times d$ mod 60 = 1 (they are inverses of each Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plaintext: 5	$C = 5^{13} = 26 \bmod 77$	Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26	$P = 26^{37} = 5 \bmod 77$	Plaintext: 5

10.2.3 Some Trivial Examples

Example 10. 6

Now assume that another person, John, wants to send a message to Bob. John can use the same public key announced by Bob (probably on his website), 13; John's plaintext is 63. John calculates the following:

Plaintext: 63	$C = 63^{13} = 28 \bmod 77$	Ciphertext: 28

Bob receives the ciphertext 28 and uses his private key 37 to decipher the ciphertext:

Ciphertext: 28	$P = 28^{37} = 63 \bmod 77$	Plaintext: 63

10.2.3 Some Trivial Examples

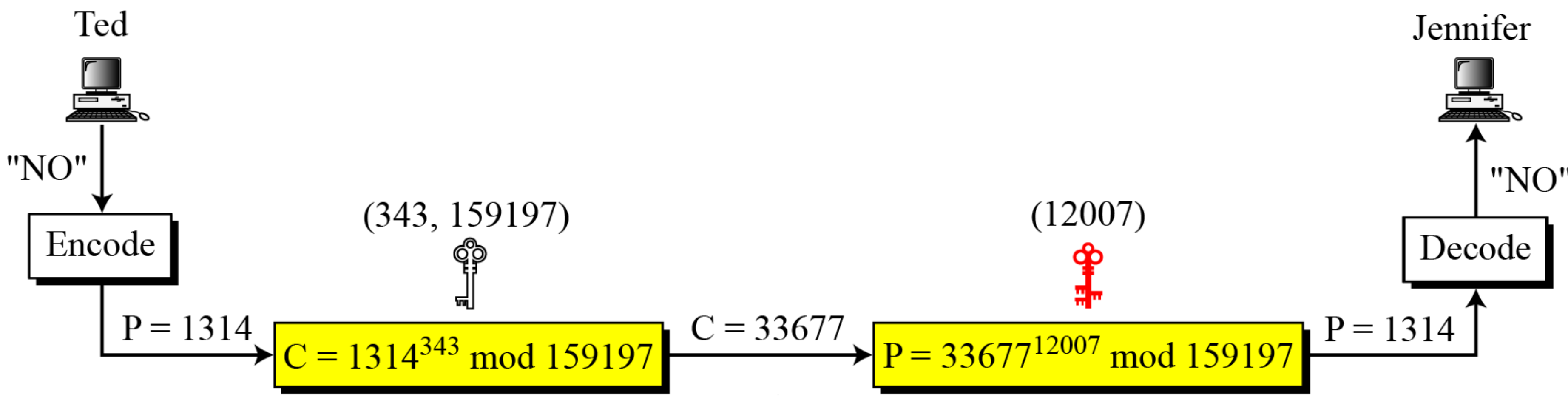
Example 10. 7

Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 159197$. She then calculates $\phi(n) = 158400$. She then chooses $e = 343$ and $d = 12007$. Show how Ted can send a message to Jennifer if he knows e and n.

Suppose Ted wants to send the message "NO" to Jennifer. He changes each character to a number (from 00 to 25), with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314. Figure 10.7 shows the process.

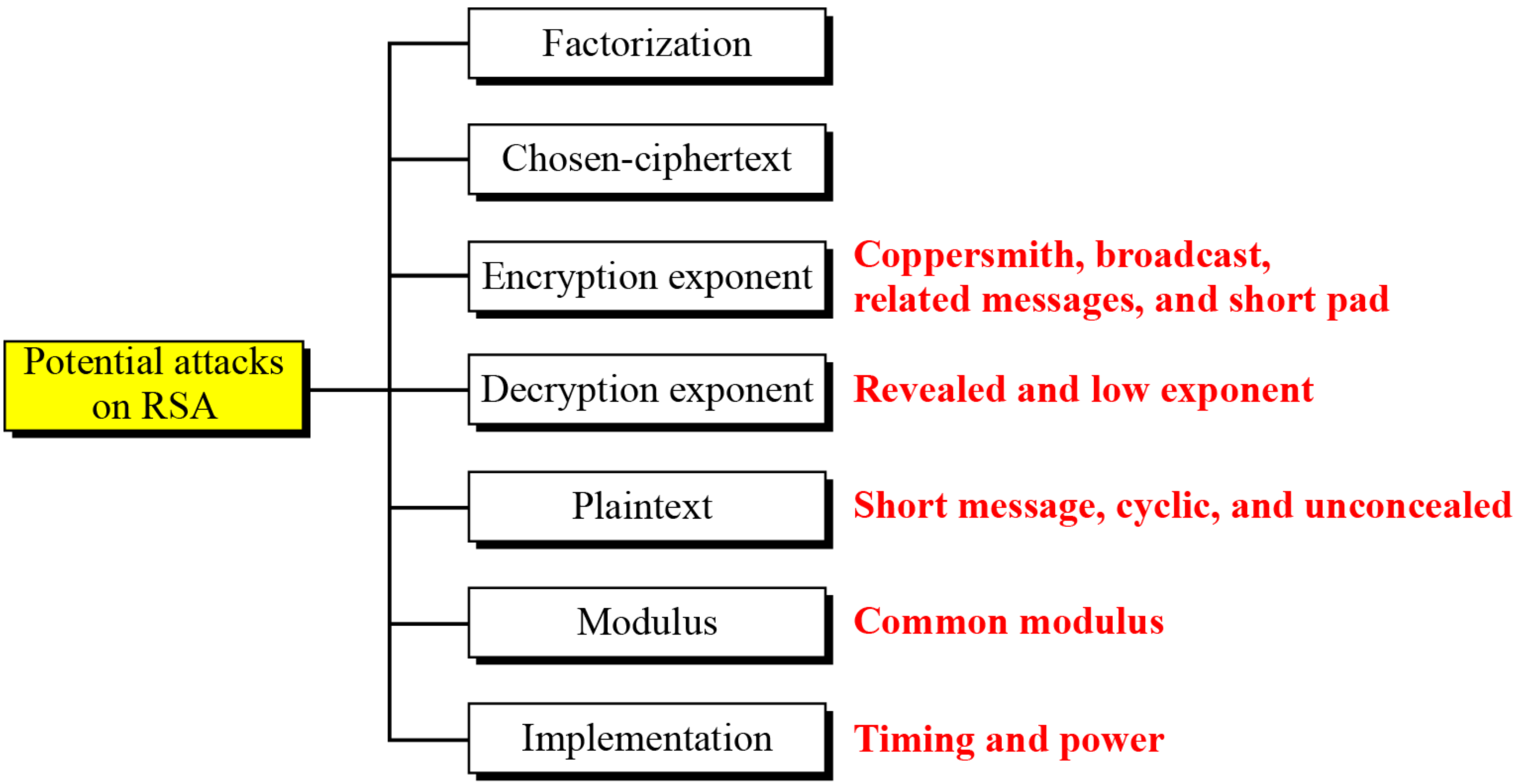
10.2.3 Continued

Figure 10.7 *Encryption and decryption in Example 10.7*



10.2.4 Attacks on RSA

Figure 10.8 *Taxonomy of potential attacks on RSA*



10.2.6 Continued

Example 10. 8

Here is a more realistic example. We choose a 512-bit p and q, calculate n and $\phi(n)$, then choose e and test for relative primeness with $\phi(n)$. We then calculate d. Finally, we show the results of encryption and decryption. The integer p is a 159-digit number.

p =	96130345313583504574191581280615427909309845594996215822583150879647940455056470638491257160180347503120986660649242019180878066742 1096063354219926661209

q =	12060191957231446918276794204450896001555925054637033936061798321731482148483764659215389453209175225273226830107120695604602513887145524969000359660045617

10.2.6 Continued

Example 10. 8 *Continued*

The modulus $n = p \times q$. It has 309 digits.

$n =$	115935041739676149688925098646158875237714573754541447754855261376 147885408326350817276878815968325168468849300625485764111250162414 552339182927162507656772727460097082714127730434960500556347274566 628060099924037102991424472292215772798531727033839381334692684137 327622000966676671831831088373420823444370953

$\phi(n) = (p - 1)(q - 1)$ has 309 digits.

$\phi(n) =$	115935041739676149688925098646158875237714573754541447754855261376 147885408326350817276878815968325168468849300625485764111250162414 552339182927162507656751054233608492916752034482627988117554787657 013923444405716989581728196098226361075467211864612171359107358640 614008885170265377277264467341066243857664128

10.2.6 Continued

Example 10. 8 *Continued*

Bob chooses e = 35535 (the ideal is 65537) and tests it to make sure it is relatively prime with ϕ(n). He then finds the inverse of *e* modulo ϕ(n) and calls it *d*.

e =	35535
d =	58008302860037763936093661289677917594669062089650962180422866111380593852822358731706286910030021710859044338402170729869087600611530620252495988444804756824096624708148581713046324064407770483313401085094738529564507193677406119732655742423721761767462077637164207600337085333288532144708859551366702948 31

10.2.6 Continued

Example 10. 8 *Continued*

Alice wants to send the message "THIS IS A TEST", which can be changed to a numeric value using the 00−26 encoding scheme (26 is the space character).

P =	1907081826081826002619041819

The ciphertext calculated by Alice is C = P*ᵉ*, which is

C =	47530912364622682720636555061054518094237179607049171652323924305445296061319932856661784341835911415119741125200568297979457173603610127821884789274156609048002350719071527718591497518846588863210114835410336165789846796838676373376577746562507928052114814184404814184430812773059004692874248559166462108656

Example 10. 8 *Continued*

Bob can recover the plaintext from the ciphertext using $P = C^d$, which is

P =	1907081826081826002619041819

The recovered plaintext is "THIS IS A TEST" after decoding.

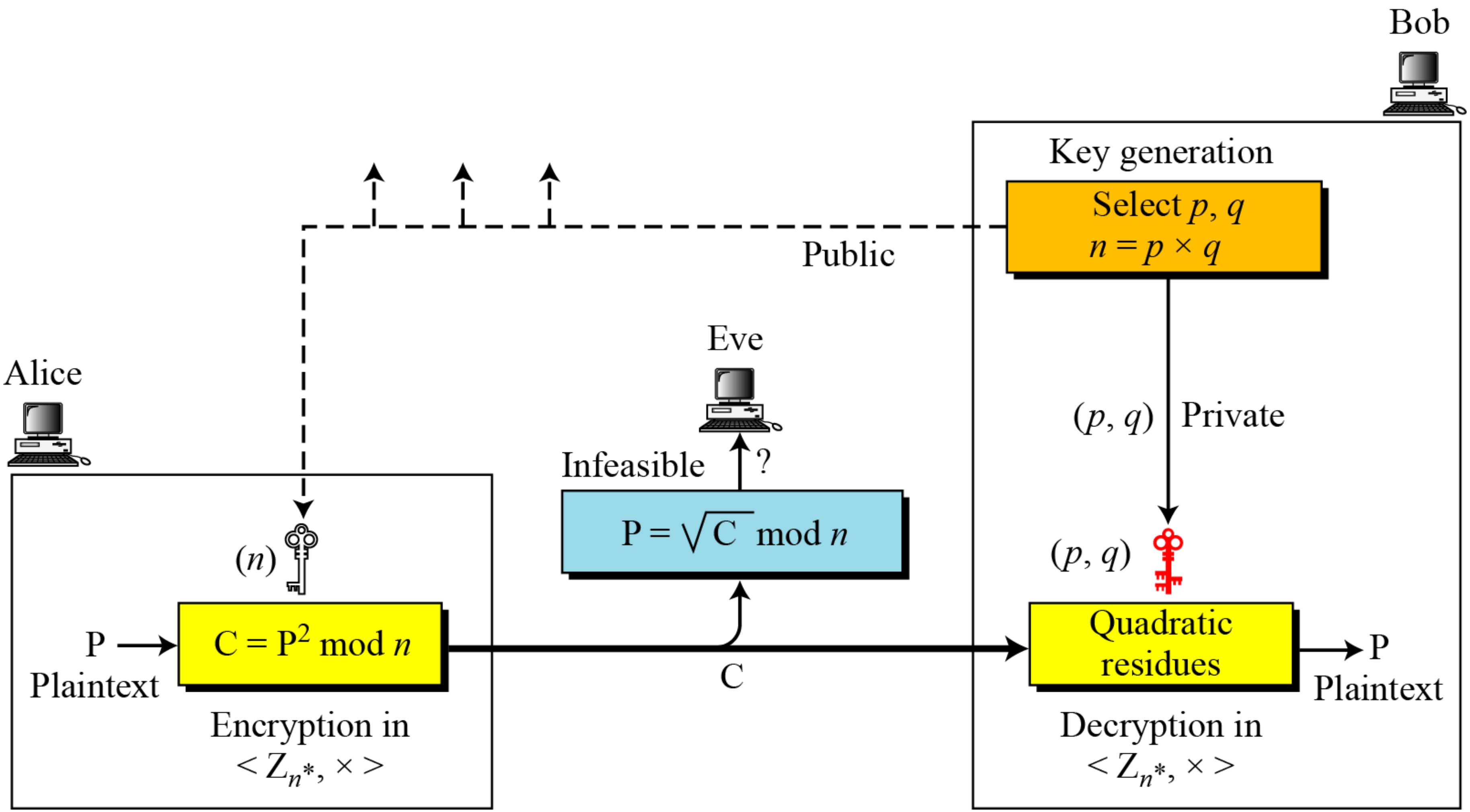
10-3 RABIN CRYPTOSYSTEM

The Rabin cryptosystem can be thought of as an RSA cryptosystem in which the value of e and d are fixed. The encryption is $C \equiv P^2 \pmod{n}$ and the decryption is $P \equiv C^{1/2} \pmod{n}$.

Topics discussed in this section:

10.3.1 Procedure
10.3.2 Security of the Rabin System

Figure 10.10 ***Rabin cryptosystem***



10.3.1 Procedure

Key Generation

Algorithm 10.6 *Key generation for Rabin cryptosystem*

Rabin_Key_Generation

{

Choose two large primes p and q in the form $4k + 3$ and $p \neq q$.

$n \leftarrow p \times q$

Public_key $\leftarrow$ n // To be announced publicly

Private_key $\leftarrow (q, n)$ // To be kept secret

return Public_key and Private_key

}

Encryption

Algorithm 10.7 *Encryption in Rabin cryptosystem*

```
Rabin_Encryption (n, P)          // n is the public key; P is the ciphertext from Zn*
{
    C ← P^2 mod n                // C is the ciphertext
    return C
}
```

10.3.1 Continued

Decryption

Algorithm 10.8 *Decryption in Rabin cryptosystem*

```
Rabin_Decryption (p, q, C)              // C is the ciphertext; p and q are private keys
{
    a1 ← +(C^((p+1)/4)) mod p
    a2 ← −(C^((p+1)/4)) mod p
    b1 ← +(C^((q+1)/4)) mod q
    b2 ← −(C^((q+1)/4)) mod q
    // The algorithm for the Chinese remainder algorithm is called four times.
    P1 ← Chinese_Remainder (a1, b1, p, q)
    P2 ← Chinese_Remainder (a1, b2, p, q)
    P3 ← Chinese_Remainder (a2, b1, p, q)
    P4 ← Chinese_Remainder (a2, b2, p, q)
    return P1, P2, P3, and P4
}
```

Note

The Rabin cryptosystem is not deterministic: Decryption creates four plaintexts.

Example 10. 9

Here is a very trivial example to show the idea.

1. **Bob selects $p = 23$ and $q = 7$. Note that both are congruent to 3 mod 4.**
2. **Bob calculates $n = p \times q = 161$.**
3. **Bob announces n publicly; he keeps p and q private.**
4. **Alice wants to send the plaintext $P = 24$. Note that 161 and 24 are relatively prime; 24 is in Z_{161}*. She calculates $C = 24^2 = 93$ mod 161, and sends the ciphertext 93 to Bob.**

10.3.1 Continued

Example 10. 9

5. **Bob receives 93 and calculates four values:**

   $a_1 = +(93^{(23+1)/4}) \bmod 23 = 1 \bmod 23$

   $a_2 = -(93^{(23+1)/4}) \bmod 23 = 22 \bmod 23$

   $b_1 = +(93^{(7+1)/4}) \bmod 7 = 4 \bmod 7$

   $b_2 = -(93^{(7+1)/4}) \bmod 7 = 3 \bmod 7$

6. **Bob takes four possible answers, (a_1, b_1), (a_1, b_2), (a_2, b_1), and (a_2, b_2), and uses the Chinese remainder theorem to find four possible plaintexts: 116, 24, 137, and 45. Note that only the second answer is Alice's plaintext.**

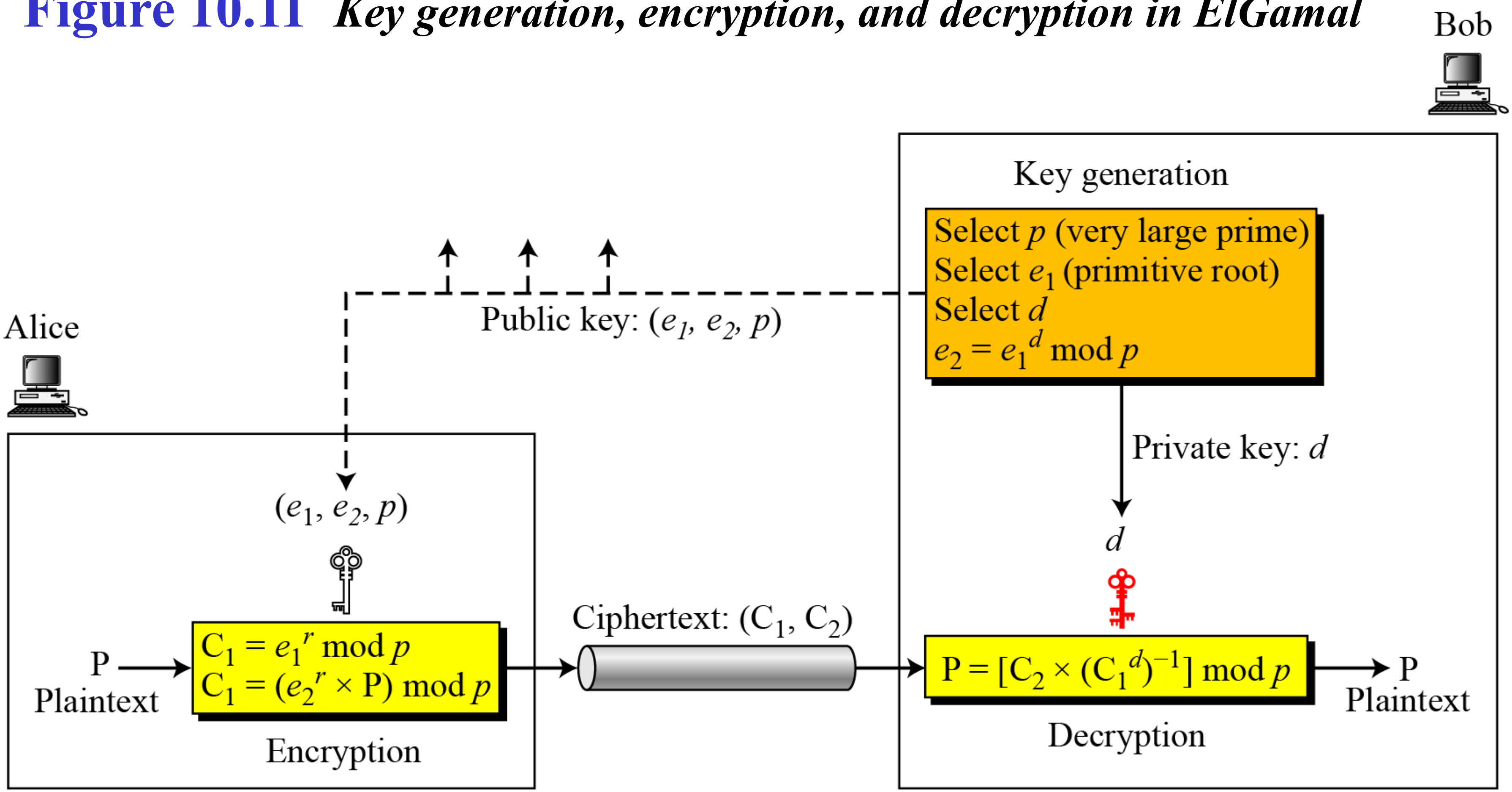
10-4 ELGAMAL CRYPTOSYSTEM

Besides RSA and Rabin, another public-key cryptosystem is ElGamal. ElGamal is based on the discrete logarithm problem discussed in Chapter 9.

Topics discussed in this section:

10.4.1 ElGamal Cryptosystem
10.4.2 Procedure
10.4.3 Proof
10.4.4 Analysis
10.4.5 Security of ElGamal
10.4.6 Application

Figure 10.11 *Key generation, encryption, and decryption in ElGamal*



10.4.2 Continued

Key Generation

Algorithm 10.9 *ElGamal key generation*

```
ElGamal_Key_Generation
{
    Select a large prime p
    Select d to be a member of the group G = <Zp*, ×> such that 1 ≤ d ≤ p − 2
    Select e1 to be a primitive root in the group G = <Zp*, ×>
    e2 ← e1^d mod p
    Public_key ← (e1, e2, p)                  // To be announced publicly
    Private_key ← d                           // To be kept secret
    return Public_key and Private_key
}
```

Table 8.3 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Algorithm 10.10 *ElGamal encryption*

```
ElGamal_Encryption (e1, e2, p, P)                  // P is the plaintext
{
    Select a random integer r in the group G = < Zp*, × >
    C1 ← e1^r mod p
    C2 ← (P × e2^r) mod p                          // C1 and C2 are the ciphertexts
    return C1 and C2
}
```

10.4.2 Continued

Algorithm 10.11 *ElGamal decryption*

```
ElGamal_Decryption (d, p, C1, C2)                // C1 and C2 are the ciphertexts
{
    P ← [C2 (C1^d)^-1] mod p                      // P is the plaintext
    return P
}
```

$P \leftarrow [C_2 (C_1^{d})^{-1}] \bmod p$

Note

The bit-operation complexity of encryption or decryption in ElGamal cryptosystem is polynomial.

Example 10. 10

Here is a trivial example. Bob chooses $p = 11$ and $e_1 = 2$. and $d = 3$ $e_2 = e_1^d = 8$. So the public keys are (2, 8, 11) and the private key is 3. Alice chooses $r = 4$ and calculates C1 and C2 for the plaintext 7.

Plaintext: 7

$C_1 = e_1^r \bmod 11 = 16 \bmod 11 = 5 \bmod 11$

$C_2 = (P \times e_2^r) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$

Ciphertext: (5, 6)

Bob receives the ciphertexts (5 and 6) and calculates the plaintext.

$[C_2 \times (C_1^d)^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$

Plaintext: 7

Example 10. 11

Instead of using $P = [C_2 \times (C_1^{d})^{-1}] \bmod p$ for decryption, we can avoid the calculation of multiplicative inverse and use $P = [C_2 \times C_1^{p-1-d}] \bmod p$ (see Fermat's little theorem in Chapter 9). In Example 10.10, we can calculate $P = [6 \times 5^{11-1-3}] \bmod 11 = 7 \bmod 11$.

Note

For the ElGamal cryptosystem, *p* must be at least 300 digits and *r* must be new for each encipherment.

10.4.3 Continued

Example 10. 12

Bob uses a random integer of 512 bits. The integer p is a 155-digit number (the ideal is 300 digits). Bob then chooses e_1, d, and calculates e_2, as shown below:

$p =$	115348992725616762449253137170143317404900945326098349598143469219 056898698622645932129754737871895144368891765264730936159299937280 61165964347353440008577
$e_1 =$	2

$d =$	1007
$e_2 =$	978864130430091895087668569380977390438800628873376876100220622332 554507074156189212318317704610141673360150884132940857248537703158 2066010072558707455

10.4.3 Continued

Example 10. 10

Alice has the plaintext P = 3200 to send to Bob. She chooses r = 545131, calculates C1 and C2, and sends them to Bob.

P =	3200
r =	545131
C_1 =	8872970693835284710225704714922756631202600672565621250181883514294172235997126811141053636617051730515815331891654009737363550802957367885690606191528 81
C_2 =	7084543330489299445770160123807949995674360218361924469617745069212446961551658007794555930803458896144024085995259195792097216288796813505827795664302950

Bob calculates the plaintext $P = C_2 \times ((C_1)^d)^{-1} \bmod p = 3200 \bmod p$.

P =	3200

10-5 ELLIPTIC CURVE CRYPTOSYSTEMS

Although RSA and ElGamal are secure asymmetric-key cryptosystems, their security comes with a price, their large keys. Researchers have looked for alternatives that give the same level of security with smaller key sizes. One of these promising alternatives is the elliptic curve cryptosystem (ECC).

Topics discussed in this section:

10.5.1 Elliptic Curves over Real Numbers
10.5.2 Elliptic Curves over GF(*p*)
10.5.3 Elliptic Curves over GF(2^n)
10.5.4 Elliptic Curve Cryptography Simulating ElGamal

10.5.1 Elliptic Curves over Real Numbers

The general equation for an elliptic curve is

$$y^2 + b_1xy + b_2y = x^3 + a_1x^2 + a_2x + a_3$$

Elliptic curves over real numbers use a special class of elliptic curves of the form

$$y^2 = x^3 + ax + b$$

Example 10. 13

Figure 10.12 shows two elliptic curves with equations $y^2 = x^3 - 4x$ and $y^2 = x^3 - 1$. Both are nonsingular. However, the first has three real roots ($x = -2$, $x = 0$, and $x = 2$), but the second has only one real root ($x = 1$) and two imaginary ones.

Figure 10.12 *Two elliptic curves over a real field*

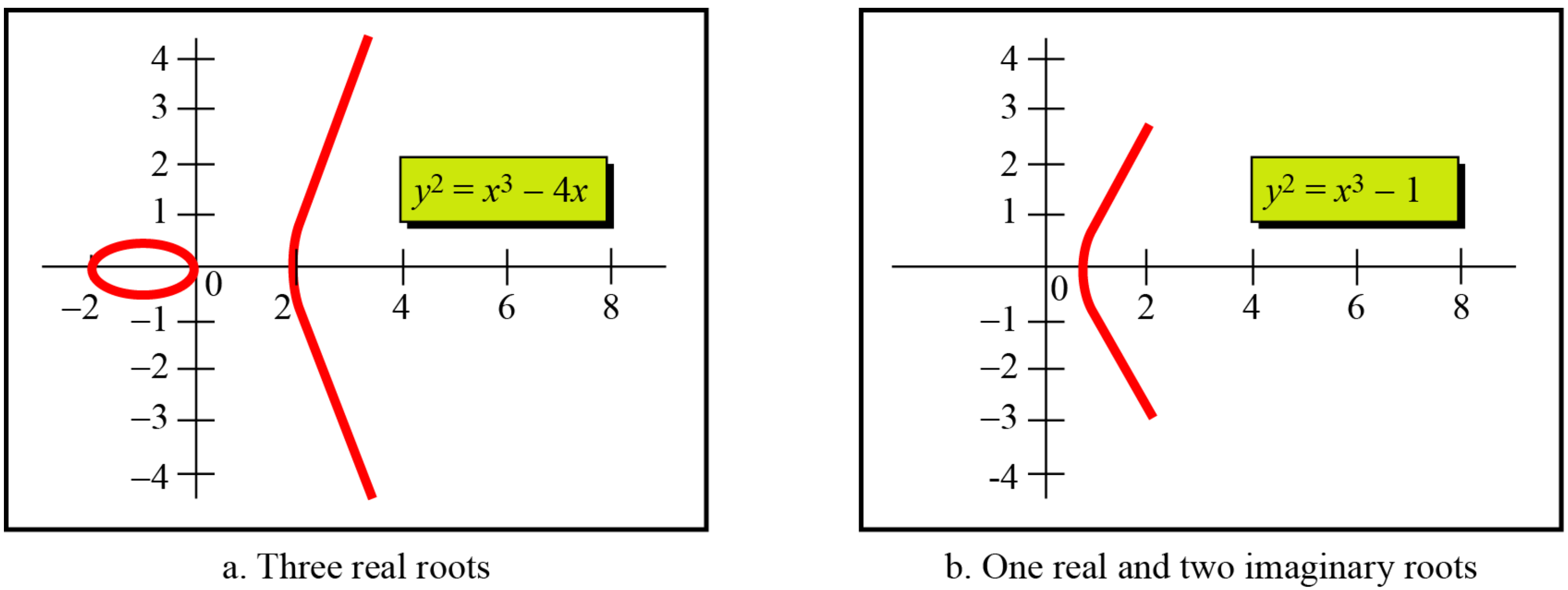
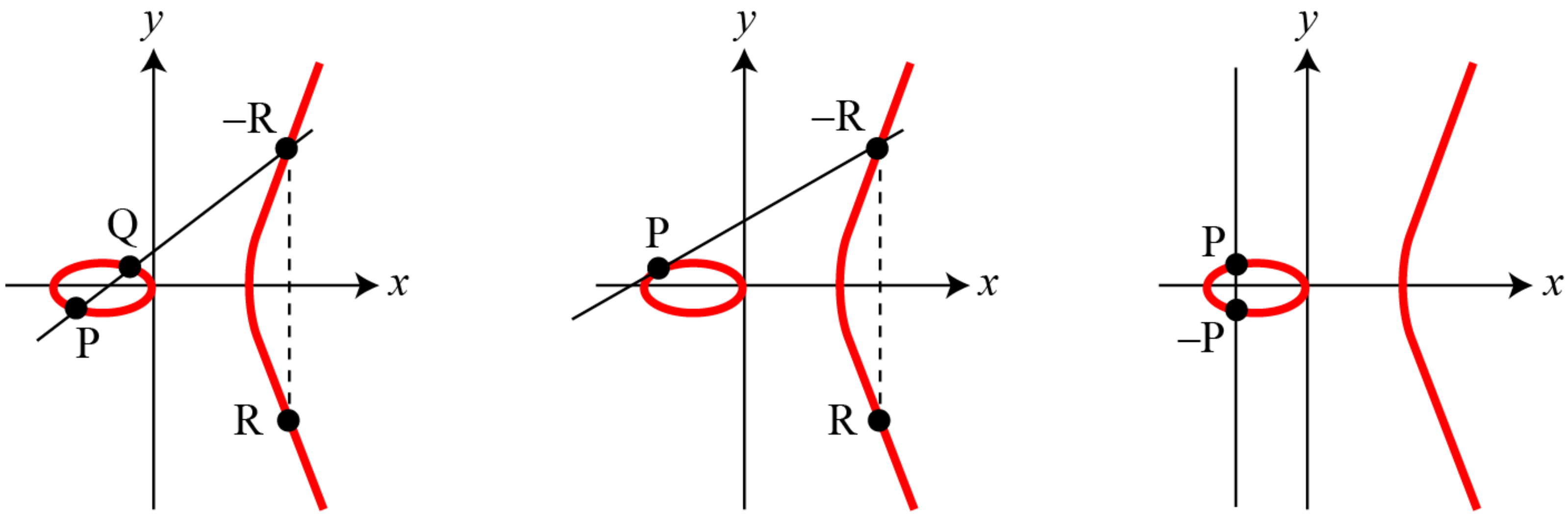


Figure 10.13 ***Three adding cases in an elliptic curve***



a. (R = P + Q) b. (R = P + P) c. (O = P + (– P))

10.5.1 Continued

1.

$$\lambda = (y_2 - y_1) / (x_2 - x_1)$$

$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = \lambda\,(x_1 - x_3) - y_1$$

2.

$$\lambda = (3x_1{}^2 + a)/(2y_1)$$

$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = \lambda\,(x_1 - x_3) - y_1$$

3. The intercepting point is at infinity; a point O as the point at infinity or zero point, which is the additive identity of the group.

10.5.2 Elliptic Curves over GF(p)

Finding an Inverse

The inverse of a point (x, y) is (x, −y), where −y is the additive inverse of y. For example, if p = 13, the inverse of (4, 2) is (4, 11).

Finding Points on the Curve

Algorithm 10.12 shows the pseudocode for finding the points on the curve Ep(a, b).

10.5.2 Continued

Algorithm 10.12 *Pseudocode for finding points on an elliptic curve*

```
ellipticCurve_points (p, a, b)                              // p is the modulus
{
    x ← 0
    while (x < p)
    {
        w ← (x^3 + ax + b) mod p                                // w is y^2
        if (w is a perfect square in Z_p) output (x, √w) (x, − √w)
        x ← x + 1
    {
}
```

Example 10. 14

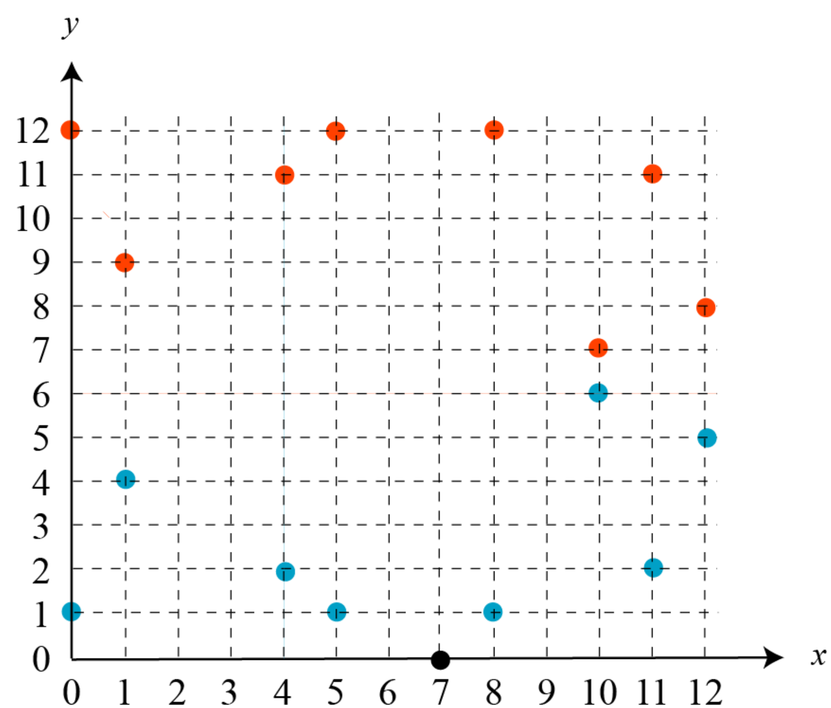
The equation is $y^2 = x^3 + x + 1$ and the calculation is done modulo 13.

Figure 10.14 ***Points on an elliptic curve over GF(p)***

P	–P
(0, 1)	(0, 12)
(1, 4)	(1, 9)
(4, 2)	(4, 11)
(5, 1)	(5, 12)
(7, 0)	(7, 0)
(8, 1)	(8, 12)
(10, 6)	(10, 7)
(11, 2)	(11, 11)
(12, 5)	(12, 18)

P = –P → (7, 0)

Points



Graph

10.5.2 Continued

Example 10. 15

Let us add two points in Example 10.14, $R = P + Q$, where $P = (4, 2)$ and $Q = (10, 6)$.

a. $\lambda = (6 - 2) \times (10 - 4)^{-1} \bmod 13 = 4 \times 6^{-1} \bmod 13 = 5 \bmod 13.$

b. $x = (5^2 - 4 - 10) \bmod 13 = 11 \bmod 13.$

c. $y = [5\ (4 - 11) - 2] \bmod 13 = 2 \bmod 13.$

d. ***$R = (11, 2)$, which is a point on the curve in Example 10.14.***

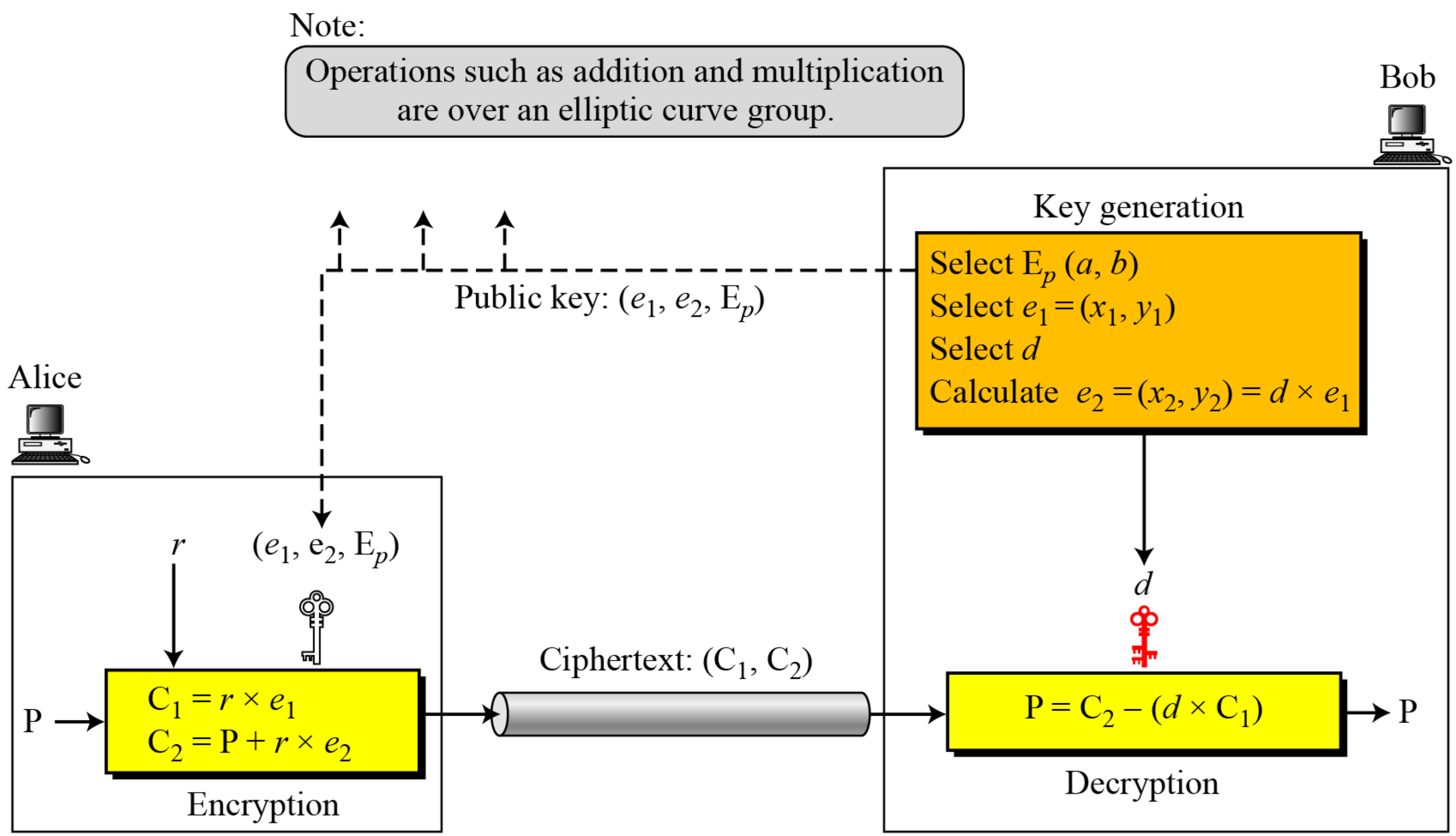
How about $E_{23}(1,1)$, let $P=(3, 10)$ and $Q=(9,7)$

$P + Q$?

$2P$?

10.5.4 ECC Simulating ElGamal

Figure 10.16 ***ElGamal cryptosystem using the elliptic curve***



10.5.4 Continued

Generating Public and Private Keys

$E(a, b)$ $e_1(x_1, y_1)$ d $e_2(x_2, y_2) = d \times e_1(x_1, y_1)$

Encryption

$$C_1 = r \times e_1 \qquad C_2 = P + r \times e_2$$

Decryption

$$P = C_2 - (d \times C_1)$$

The minus sign here means adding with the inverse.

Note

The security of ECC depends on the difficulty of solving the elliptic curve logarithm problem.

Example 10. 19

1. ***Bob selects $E_{67}(2, 3)$ as the elliptic curve over GF(p).***
2. ***Bob selects $e_1 = (2, 22)$ and $d = 4$.***
3. ***Bob calculates $e_2 = (13, 45)$, where $e_2 = d \times e_1$.***
4. ***Bob publicly announces the tuple (E, e_1, e_2).***
5. ***Alice sends the plaintext $P = (24, 26)$ to Bob. She selects $r = 2$.***
6. ***Alice finds the point $C_1=(35, 1)$, $C_2=(21, 44)$.***
7. ***Bob receives C_1, C_2. He uses $4xC_1(35,1)$ to get (23, 25), inverts the points (23, 25) to get the points (23, 42).***
8. ***Bob adds (23, 42) with $C_2=(21, 44)$ to get the original one P=(24, 26).***

10.5.4 Comparable Key Sizes for Equivalent Security

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