UNIT - III

Mathematics of Cryptography

Part III: Primes and Related Congruence Equations

Objectives

- To introduce prime numbers and their applications in cryptography.
- To discuss some primality test algorithms and their efficiencies.
- To discuss factorization algorithms and their applications in cryptography.
- To describe the Chinese remainder theorem and its application.
- To introduce quadratic congruence.
- To introduce modular exponentiation and logarithm.

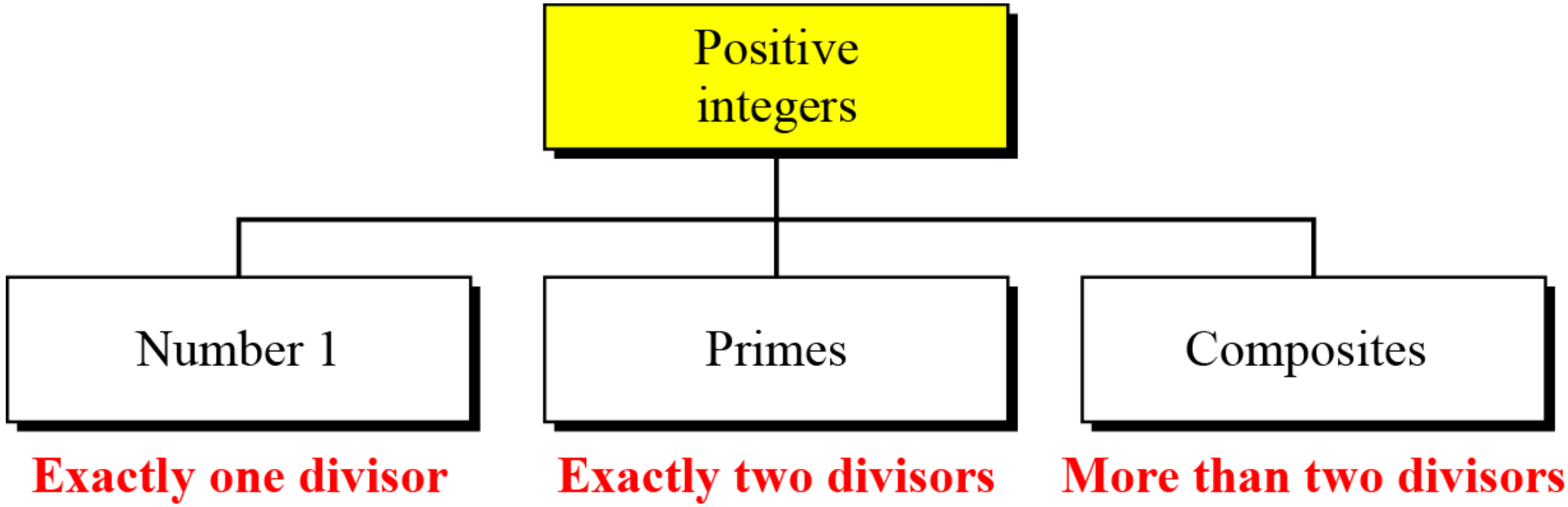
6-1 PRIMES

Asymmetric-key cryptography uses primes extensively. The topic of primes is a large part of any book on number theory.

Topics discussed in this section:

6.1.1 Definition

Figure 6.1 *Three groups of positive integers*



Note

A prime is divisible only by itself and 1.

Given a number n, how can we determine if n is a prime? The answer is that we need to see if the number is divisible by all primes less than

$$\sqrt{n}$$

We know that this method is inefficient, but it is a good start.

Theorem

If n is composite, then n has a prime divisor less than or equal to $\sqrt{n}$.

Proof.

- Let $n = ab,\ 1 < a < n, 1 < b < n$.
- We can't have both $a > \sqrt{n}$ and $b > \sqrt{n}$ since this would lead to $ab > n$.
- Therefore, n must have a prime divisor less than or equal to $\sqrt{n}$.

$\square$

Example 6.5

Is 97 a prime?

Solution

The floor of $\sqrt{97} = 9$. The primes less than 9 are 2, 3, 5, and 7. We need to see if 97 is divisible by any of these numbers. It is not, so 97 is a prime.

Example 6.6

Is 301 a prime?

Solution

The floor of $\sqrt{301} = 17$. We need to check 2, 3, 5, 7, 11, 13, and 17. The numbers 2, 3, and 5 do not divide 301, but 7 does. Therefore 301 is not a prime.

Euler's phi-function, ϕ (n), which is sometimes called the *Euler's totient function plays a very important role in cryptography.*

1. $\phi(1) = 0$.
2. $\phi(p) = p - 1$ if p is a prime.
3. $\phi(m \times n) = \phi(m) \times \phi(n)$ if m and n are relatively prime.
4. $\phi(p^e) = p^e - p^{e-1}$ if p is a prime.

We can combine the above four rules to find the value of $\phi(n)$. For example, if n can be factored as

$$n = p_1^{e_1} \times p_2^{e_2} \times \ldots \times p_k^{e_k}$$

then we combine the third and the fourth rule to find

$$\boxed{\phi(n) = (p_1^{e_1} - p_1^{e_1-1}) \times (p_2^{e_2} - p_2^{e_2-1}) \times \cdots \times (p_k^{e_k} - p_k^{e_k-1})}$$

Note

The difficulty of finding $\phi(n)$ depends on the difficulty of finding the factorization of *n*.

Example 6.7

What is the value of $\phi(13)$?

Solution

Because 13 is a prime, $\phi(13) = (13 - 1) = 12$.

Example 6.8

What is the value of $\phi(10)$?

Solution

We can use the third rule: $\phi(10) = \phi(2) \times \phi(5) = 1 \times 4 = 4$, because 2 and 5 are primes.

Example 6.9

What is the value of $\phi(240)$?

Solution

We can write $240 = 2^4 \times 3^1 \times 5^1$. Then

$$\phi(240) = (2^4 - 2^3) \times (3^1 - 3^0) \times (5^1 - 5^0) = 64$$

Example 6.10

Can we say that $\phi(49) = \phi(7) \times \phi(7) = 6 \times 6 = 36$?

Solution

No. The third rule applies when *m* and *n* are relatively prime. Here $49 = 7^2$. We need to use the fourth rule: $\phi(49) = 7^2 - 7^1 = 42$.

Example 6.11

What is the number of elements in Z_{14}*?

Solution

The answer is $\phi(14) = \phi(7) \times \phi(2) = 6 \times 1 = 6$. The members are 1, 3, 5, 9, 11, and 13.

Note

Interesting point: If $n > 2$, the value of $\phi(n)$ is even.

First Version

$$\boxed{a^{p-1} \equiv 1 \bmod p}$$

If p is prime and a is an integer such that p does not divide a

Second Version

$$\boxed{a^{p} \equiv a \bmod p}$$

If p is prime and a is an integer

Example 6.12

Find the result of 6^{10} mod 11.

Solution

We have 6^{10} mod 11 = 1. This is the first version of Fermat's little theorem where $p = 11$.

Example 6.13

Find the result of 3^{12} mod 11.

Solution

Here the exponent (12) and the modulus (11) are not the same. With substitution this can be solved using Fermat's little theorem.

$$3^{12} \bmod 11 = (3^{11} \times 3) \bmod 11 = (3^{11} \bmod 11)\,(3 \bmod 11) = (3 \times 3) \bmod 11 = 9$$

Multiplicative Inverses

$$\boxed{a^{-1} \bmod p = a^{p-2} \bmod p}$$

Example 6.14

The answers to multiplicative inverses modulo a prime can be found without using the extended Euclidean algorithm:

a. $8^{-1} \bmod 17 = 8^{17-2} \bmod 17 = 8^{15} \bmod 17 = 15 \bmod 17$

b. $5^{-1} \bmod 23 = 5^{23-2} \bmod 23 = 5^{21} \bmod 23 = 14 \bmod 23$

c. $60^{-1} \bmod 101 = 60^{101-2} \bmod 101 = 60^{99} \bmod 101 = 32 \bmod 101$

d. $22^{-1} \bmod 211 = 22^{211-2} \bmod 211 = 22^{209} \bmod 211 = 48 \bmod 211$

If a and n are coprime then

First Version

$$\boxed{a^{\phi(n)} \equiv 1 \pmod{n}}$$

Second Version

$$\boxed{a^{k \times \phi(n) + 1} \equiv a \pmod{n}}$$

It can be thought of as generalization of fermat theorm. Here modulus is an integer and not a prime

Example 6.15

Find the result of 6^{24} mod 35.

Solution

We have $6^{24} \bmod 35 = 6^{\phi(35)} \bmod 35 = 1$.

Example 6.16

Find the result of 20^{62} mod 77.

Solution

If we let $k = 1$ on the second version, we have

$$20^{62} \bmod 77 = (20 \bmod 77)\,(20^{\phi(77)+1} \bmod 77) \bmod 77$$
$$= (20)(20) \bmod 77 = 15.$$

Multiplicative Inverses

Euler's theorem can be used to find multiplicative inverses modulo a composite.

$$\boxed{a^{-1} \bmod n = a^{\phi(n)-1} \bmod n}$$

Example 6.17

The answers to multiplicative inverses modulo a composite can be found without using the extended Euclidean algorithm if we know the factorization of the composite:

a. $8^{-1} \bmod 77 = 8^{\phi(77)-1} \bmod 77 = 8^{59} \bmod 77 = 29 \bmod 77$

b. $7^{-1} \bmod 15 = 7^{\phi(15)-1} \bmod 15 = 7^{7} \bmod 15 = 13 \bmod 15$

c. $60^{-1} \bmod 187 = 60^{\phi(187)-1} \bmod 187 = 60^{159} \bmod 187 = 53 \bmod 187$

d. $71^{-1} \bmod 100 = 71^{\phi(100)-1} \bmod 100 = 71^{39} \bmod 100 = 31 \bmod 100$

Mersenne Primes

$$\boxed{M_p = 2^p - 1}$$

$M_2 = 2^2 - 1 = 3$

$M_3 = 2^3 - 1 = 7$

$M_5 = 2^5 - 1 = 31$

$M_7 = 2^7 - 1 = 127$

$M_{11} = 2^{11} - 1 = 2047$ **Not a prime ($2047 = 23 \times 89$)**

$M_{13} = 2^{13} - 1 = 8191$

$M_{17} = 2^{17} - 1 = 131071$

Note

A number in the form $M_p = 2^p - 1$ is called a Mersenne number and may or may not be a prime.

Fermat Primes

$$\mathbf{F}_n = 2^{2^n} + 1$$

$F_0 = 3 \quad F_1 = 5 \quad F_2 = 17 \quad F_3 = 257 \quad F_4 = 65537$

$F_5 = 4294967297 = 641 \times 6700417$ *Not a prime*

6-4 CHINESE REMAINDER THEOREM

The Chinese remainder theorem (CRT) is used to solve a set of congruent equations with one variable but different moduli, which are relatively prime, as shown below:

$$
\begin{gathered}
x \equiv a_1 \pmod{m_1} \\
x \equiv a_2 \pmod{m_2} \\
\cdots \\
x \equiv a_k \pmod{m_k}
\end{gathered}
$$

Example 6.35

The following is an example of a set of equations with different moduli:

$$x \equiv 2 \pmod{3}$$
$$x \equiv 3 \pmod{5}$$
$$x \equiv 2 \pmod{7}$$

The solution to this set of equations is given in the next section; for the moment, note that the answer to this set of equations is $x = 23$. This value satisfies all equations: $23 \equiv 2 \pmod{3}$, $23 \equiv 3 \pmod{5}$, and $23 \equiv 2 \pmod{7}$.

Solution To Chinese Remainder Theorem

1. Find $M = m_1 \times m_2 \times \ldots \times m_k$. This is the common modulus.
2. Find $M_1 = M/m_1$, $M_2 = M/m_2$, …, $M_k = M/m_k$.
3. Find the multiplicative inverse of M_1, M_2, …, M_k using the corresponding moduli (m_1, m_2, …, m_k). Call the inverses M_1^{-1}, M_2^{-1}, …, M_k^{-1}.
4. The solution to the simultaneous equations is

$$\boxed{x = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + \cdots + a_k \times M_k \times M_k^{-1}) \bmod M}$$

Example 9.36

Find the solution to the simultaneous equations:

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

Solution

We follow the four steps.

1. $M = 3 \times 5 \times 7 = 105$

2. $M_1 = 105 / 3 = 35$, $M_2 = 105 / 5 = 21$, $M_3 = 105 / 7 = 15$

3. The inverses are $M_1^{-1} = 2$, $M_2^{-1} = 1$, $M_3^{-1} = 1$

4. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 = 23 \bmod 105$

Example 9.37

Find an integer that has a remainder of 3 when divided by 7 and 13, but is divisible by 12.

Solution

This is a CRT problem. We can form three equations and solve them to find the value of x.

$$x = 3 \bmod 7$$
$$x = 3 \bmod 13$$
$$x = 0 \bmod 12$$

If we follow the four steps, we find x = 276. We can check that 276 = 3 mod 7, 276 = 3 mod 13 and 276 is divisible by 12 (the quotient is 23 and the remainder is zero).

9-6 EXPONENTIATION AND LOGARITHM

Exponentiation: $y = a^x$ $\rightarrow$ **Logarithm:** $x = \log_a y$

Topics discussed in this section:

9.6.1 Exponentiation
9.6.2 Logarithm

Fast Exponentiation

Figure 9.6 *The idea behind the square-and-multiply method*

$$y = a^{\boxed{x_{n_b-1} \times 2^{n_b-1} + x_{n_b-2} \times 2^{n_b-2} + \cdots + x_1 \times 2^1 + x_0 \times 2^0}}$$

in which x_i is 0 or 1

$$y = \boxed{\boxed{a^{2^{n_b-1}}} \text{ or } \boxed{1}} \times \boxed{\boxed{a^{2^{n_b-2}}} \text{ or } \boxed{1}} \times \cdots \times \boxed{\boxed{a^2} \text{ or } \boxed{1}} \times \boxed{\boxed{a} \text{ or } \boxed{1}}$$

Example:

$$\boxed{y = a^9 = a^{1001_2} = a^8 \times 1 \times 1 \times a}$$

Algorithm 9.7 *Pseudocode for square-and-multiply algorithm*

```
Square_and_Multiply (a, x, n)
{
    y ← 1
    for (i ← 0 to n_b − 1)                     // n_b is the number of bits in x
    {
        if (x_i = 1)   y ← a × y  mod n        // multiply only if the bit is 1

        a ← a^2 mod n                          // squaring is not needed in the last iteration
    }
    return y
}
```

Example 9.45

Figure 9.7 shows the process for calculating $y = a^x$ using the Algorithm 9.7 (for simplicity, the modulus is not shown). In this case, $x = 22 = (10110)2$ in binary. The exponent has five bits.

Figure 9.7 *Demonstration of calculation of a^{22} using square-and-multiply method*

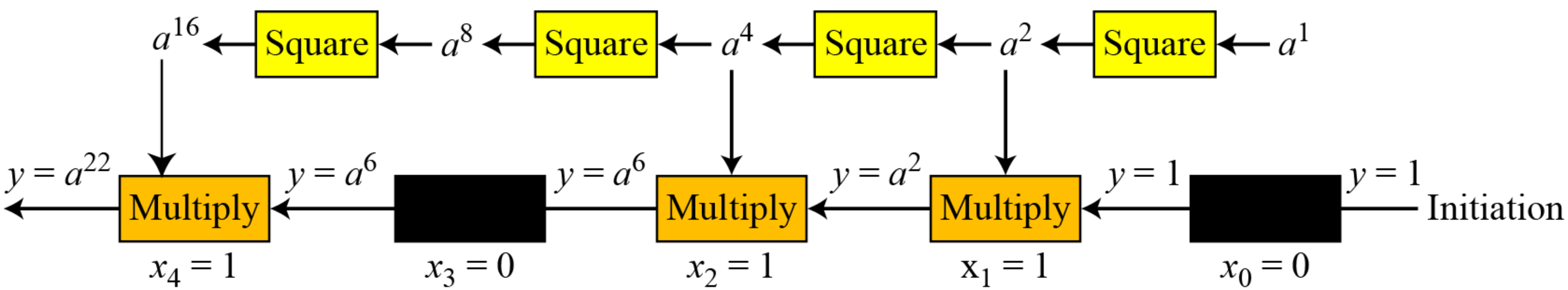


Table 9.3 *Calculation of 17^{22} mod 21*

i	x_i	Multiplication (Initialization: $y = 1$)		Squaring (Initialization: $a = 17$)
0	0		$\rightarrow$	$a = 17^2 \bmod 21 = 16$
1	1	$y = 1 \times 16 \bmod 21 = 16$	$\rightarrow$	$a = 16^2 \bmod 21 = 4$
2	1	$y = 16 \times 4 \bmod 21 = 1$	$\rightarrow$	$a = 4^2 \bmod 21 = 16$
3	0		$\rightarrow$	$a = 16^2 \bmod 21 = 4$
4	1	$y = 1 \times 4 \bmod 21 = 4$	$\rightarrow$	

How about 21^{24} mod 8?

In cryptography, we also need to discuss modular logarithm.

Exhaustive Search

Algorithm 9.8 *Exhaustive search for modular logarithm*

```
Modular_Logarithm (a, y, n)
{
    for (x = 1 to n −1)                          // k is the number of bits in x
    {
        if (y ≡ a^x mod n) return x
    }
    return failure
}
```

Order of an Element

Example 9.47

Find the order of all elements in $G = <Z_{10}*, \times>$.

Solution

This group has only $\phi(10) = 4$ elements: 1, 3, 7, 9. We can find the order of each element by trial and error.

a. $1^1 \equiv 1 \bmod (10) \rightarrow \text{ord}(1) = 1$.
b. $3^4 \equiv 1 \bmod (10) \rightarrow \text{ord}(3) = 4$.
c. $7^4 \equiv 1 \bmod (10) \rightarrow \text{ord}(7) = 4$.
d. $9^2 \equiv 1 \bmod (10) \rightarrow \text{ord}(9) = 2$.

The idea of Discrete Logarithm

Properties of $G = \langle Z_p{}^*, \times\rangle$:

1. *Its elements include all integers from 1 to $p - 1$.*

2. *It always has primitive roots.*

3. *It is cyclic. The elements can be created using g^x where x is an integer from 1 to $\phi(n) = p - 1$.*

4. *The primitive roots can be thought as the base of logarithm.*

Solution to Modular Logarithm Using Discrete Logs

Tabulation of Discrete Logarithms

Table 9.6 *Discrete logarithm for* $\mathbf{G} = \langle \mathbf{Z}_7{}^*, \times \rangle$

y	*1*	*2*	*3*	*4*	*5*	*6*
$x = L_3\, y$	6	2	1	4	5	3
$x = L_5\, y$	6	4	5	2	1	3

Using Properties of Discrete Logarithms

Table 9.7 *Comparison of traditional and discrete logarithms*

Traditional Logarithm	*Discrete Logarithms*
$\log_a 1 = 0$	$L_g 1 \equiv 0 \pmod{\phi(n)}$
$\log_a (x \times y) = \log_a x + \log_a y$	$L_g(x \times y) \equiv (L_g x + L_g y) \pmod{\phi(n)}$
$\log_a x^k = k \times \log_a x$	$L_g x^k \equiv k \times L_g x \pmod{\phi(n)}$

Using Algorithms Based on Discrete

Note

The discrete logarithm problem has the same complexity as the factorization problem.

Asymmetric-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

7-1 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:

7.1.1 Keys
7.1.2 General Idea
7.1.3 Need for Both
7.1.4 Trapdoor One-Way Function
7.1.5 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.

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

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

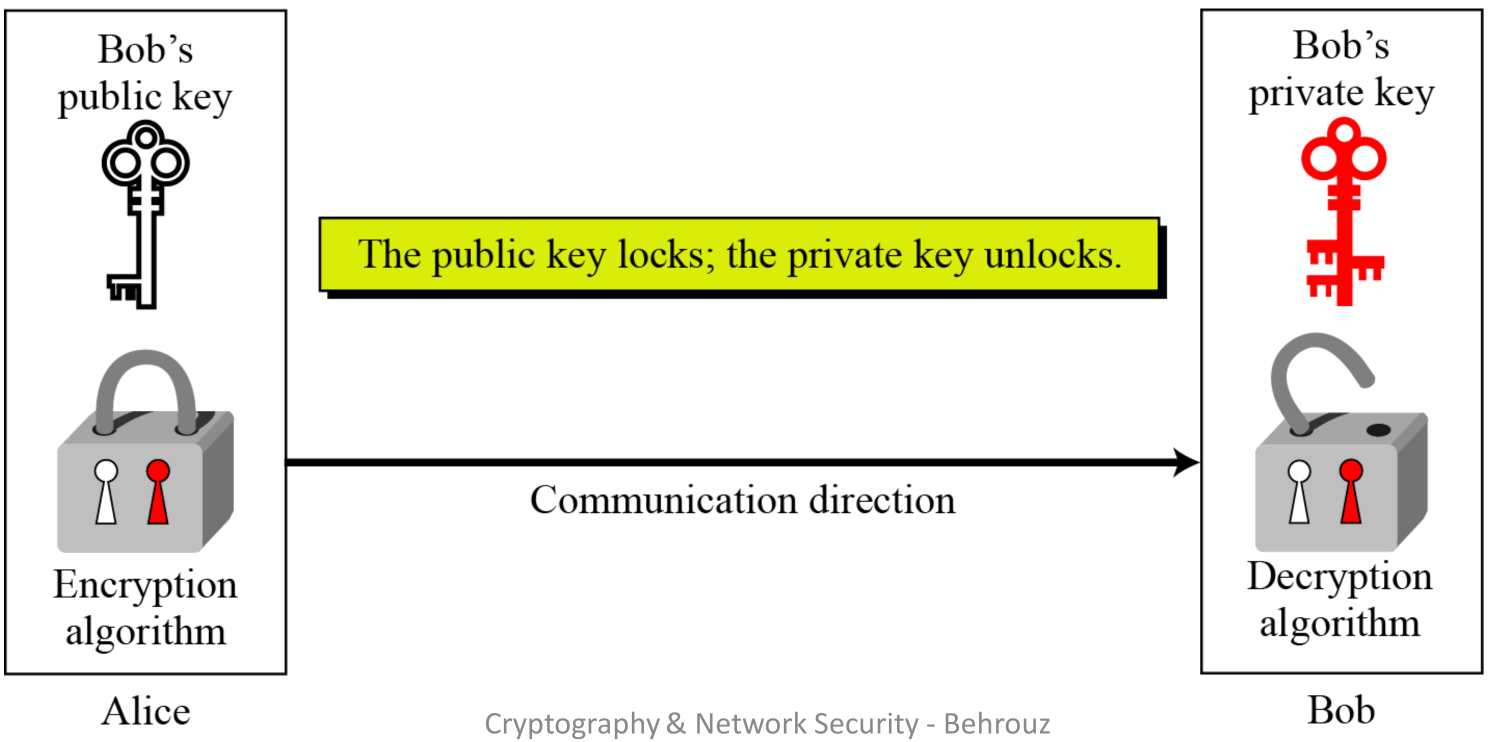
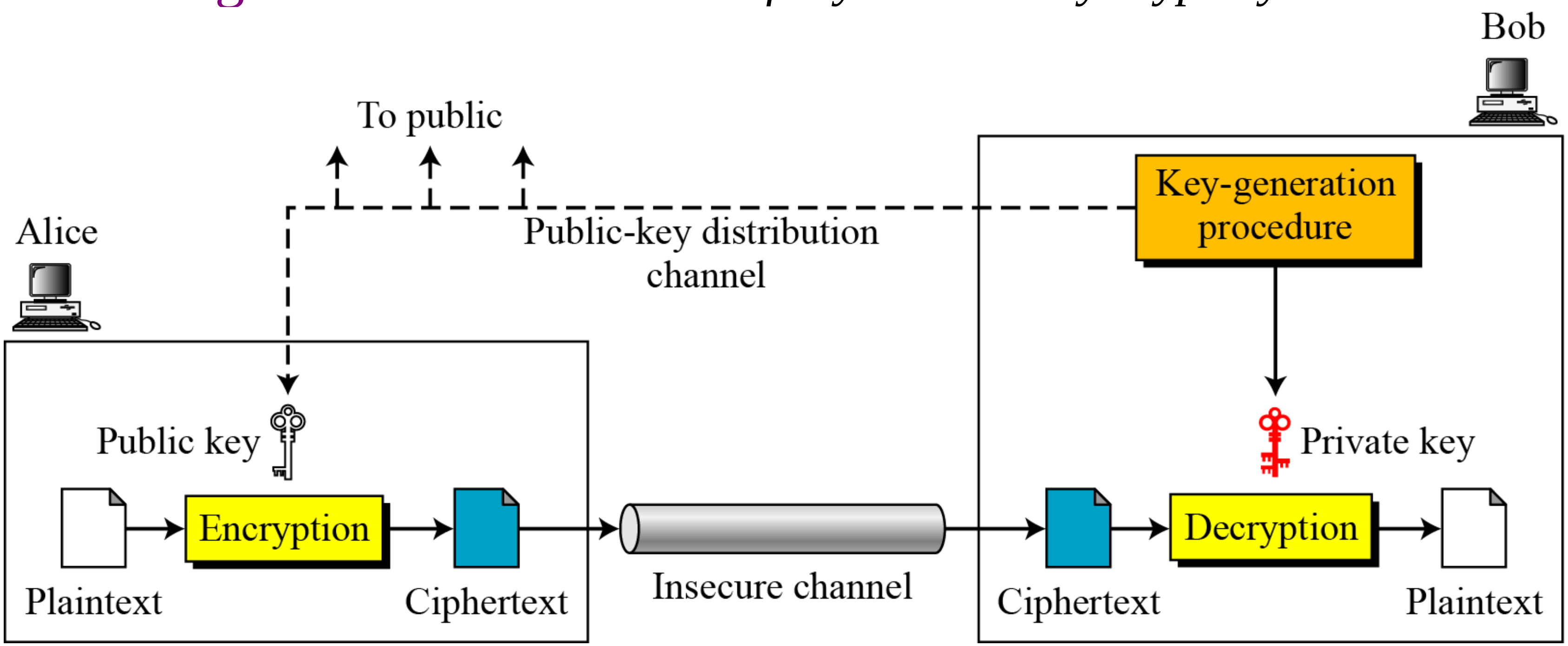


Figure 10.2 *General idea of asymmetric-key cryptosystem*



Plaintext/Ciphertext

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

Encryption/Decryption

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

The decryption function f is used only for encryption
The decryption function g is used only for decryption

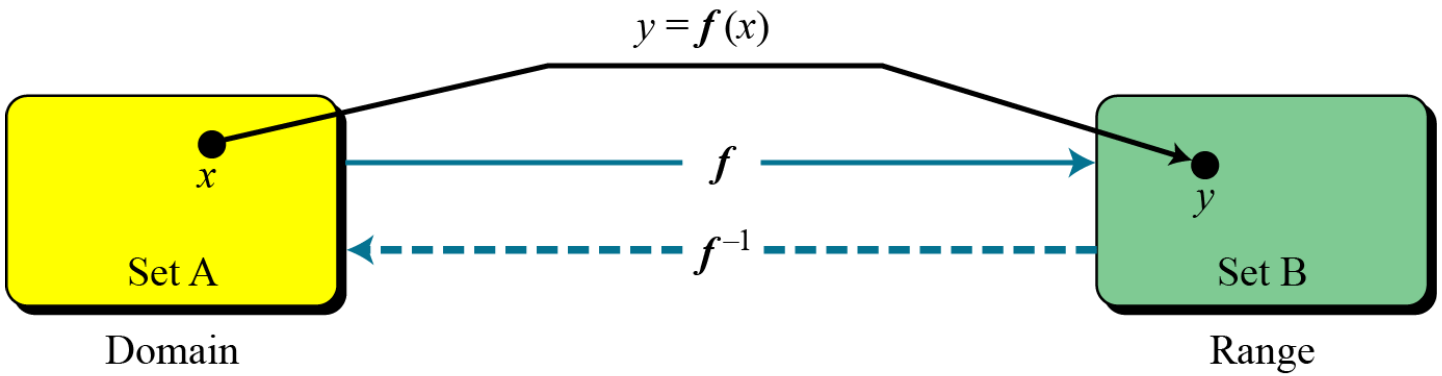
10.1.3 Need for Both

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.

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*



Invertible function

An invertible function is a function that associate each element in the range with exactly one element

In the domain

One-Way Function (OWF)

1. f is easy to compute. ie x,y =f(x)
2. f^{-1} is difficult to compute. Y to calcuate $x=f^{-1}(y)$

Trapdoor One-Way Function (TOWF)

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

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

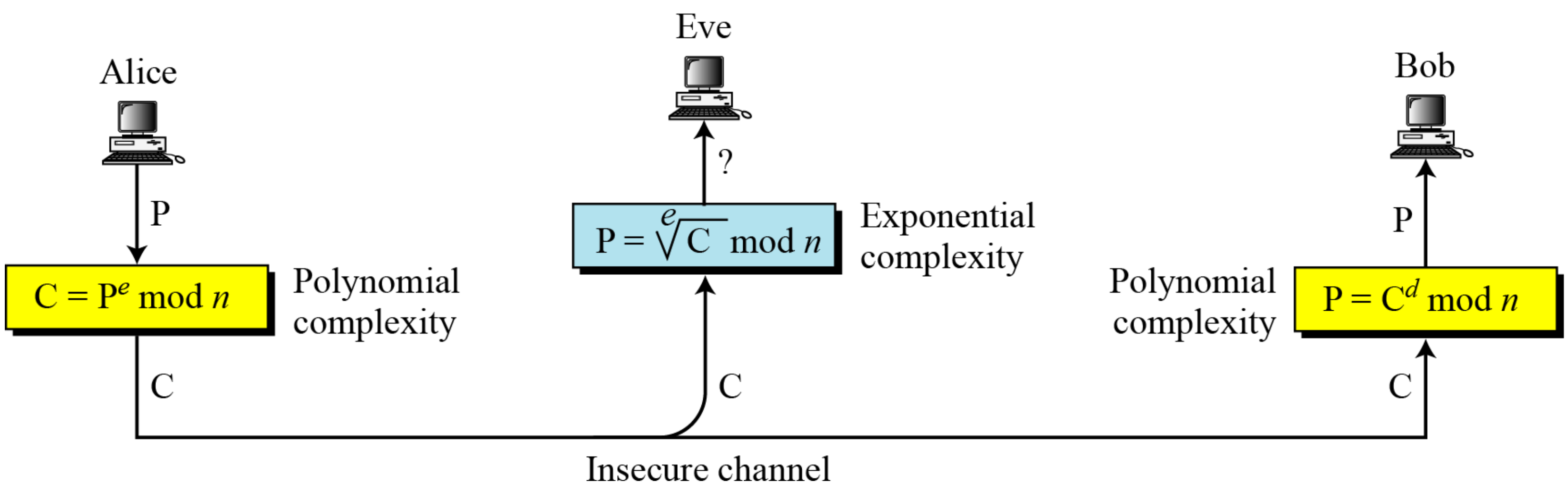
When n is large, the function $y = x^k \bmod n$ is a trapdoor one-way function. Given x, k, and n, it is easy to calculate y. Given y, k, and n, it is very difficult to calculate x. This is the discrete logarithm problem. However, if we know the trapdoor, k′ such that $k \times k' = 1 \bmod \phi(n)$,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

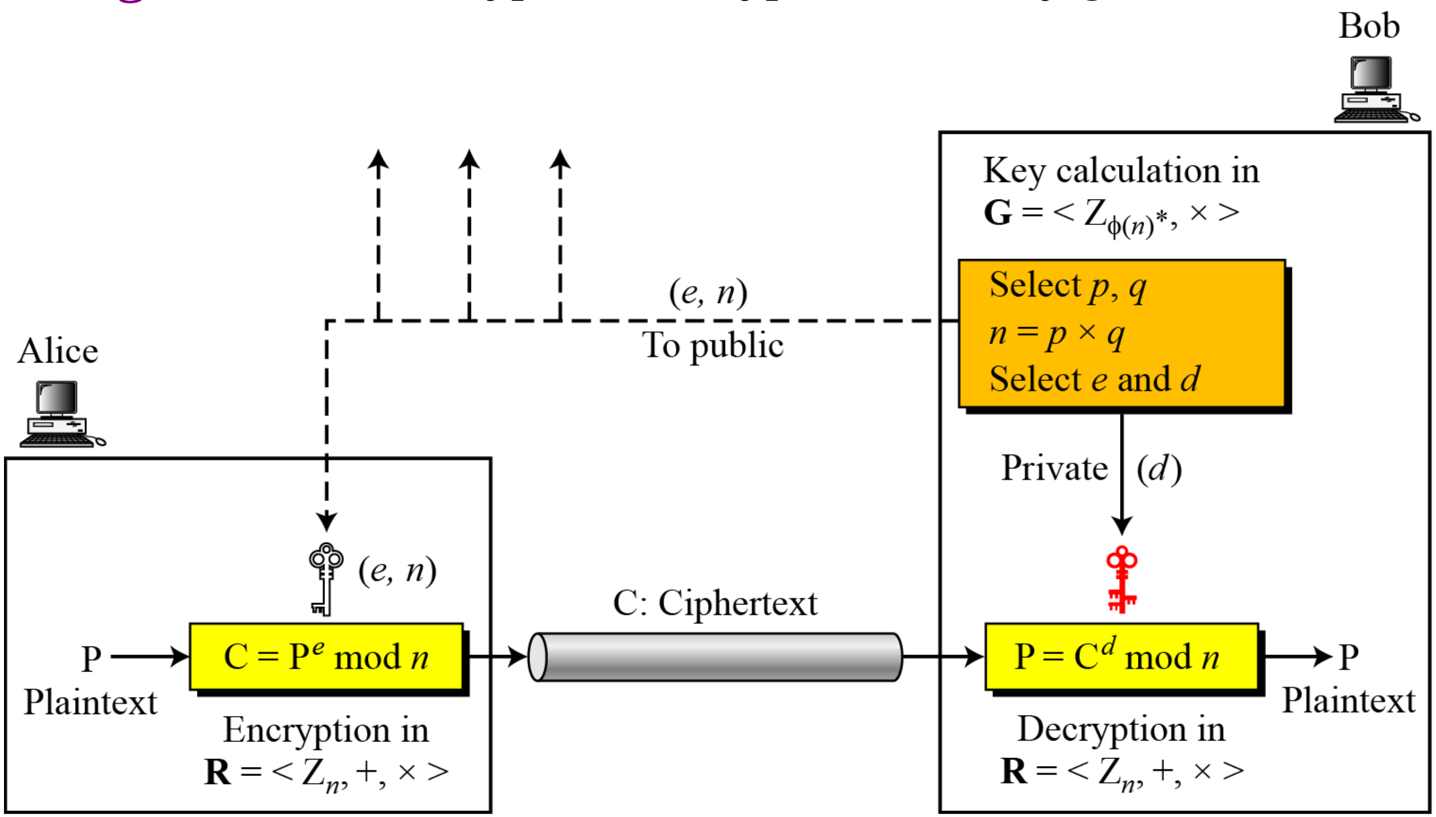
Topics discussed in this section:

Figure 10.5 *Complexity of operations in RSA*



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

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



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 ≠ q.
    n ← p × q
    φ(n) ← (p − 1) × (q − 1)
    Select e such that 1 < e < φ(n) and e is coprime to φ(n)
    d ← e^−1 mod φ(n)                    // d is inverse of e modulo φ(n)
    Public_key ← (e, n)                   // To be announced publicly
    Private_key ← d                       // To be kept secret
    return Public_key and Private_key
}
```

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.

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
}
```

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)

Figure 10.7 *Encryption and decryption in Example 10.7*

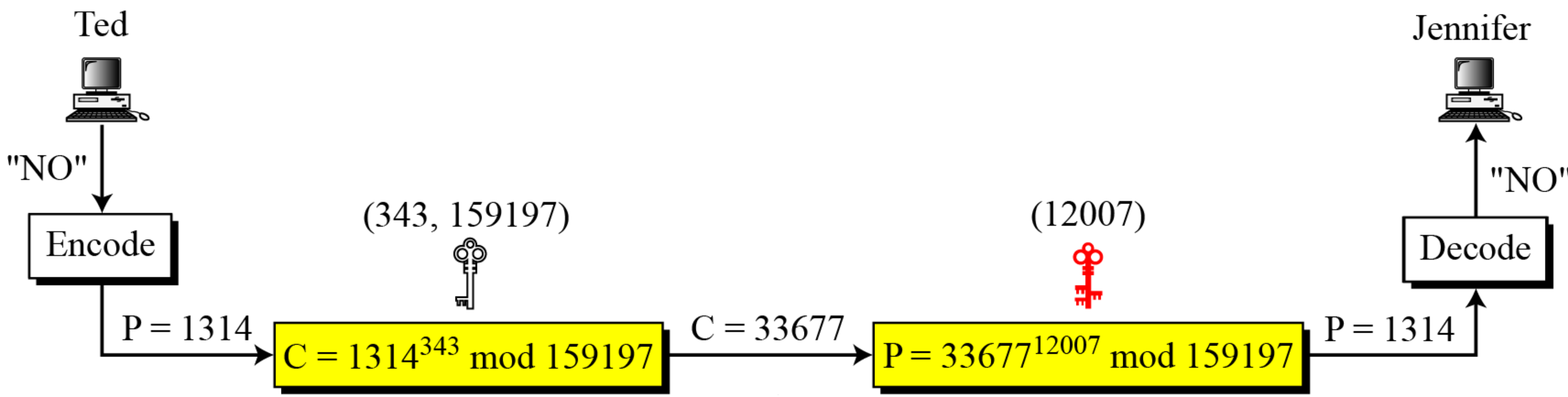
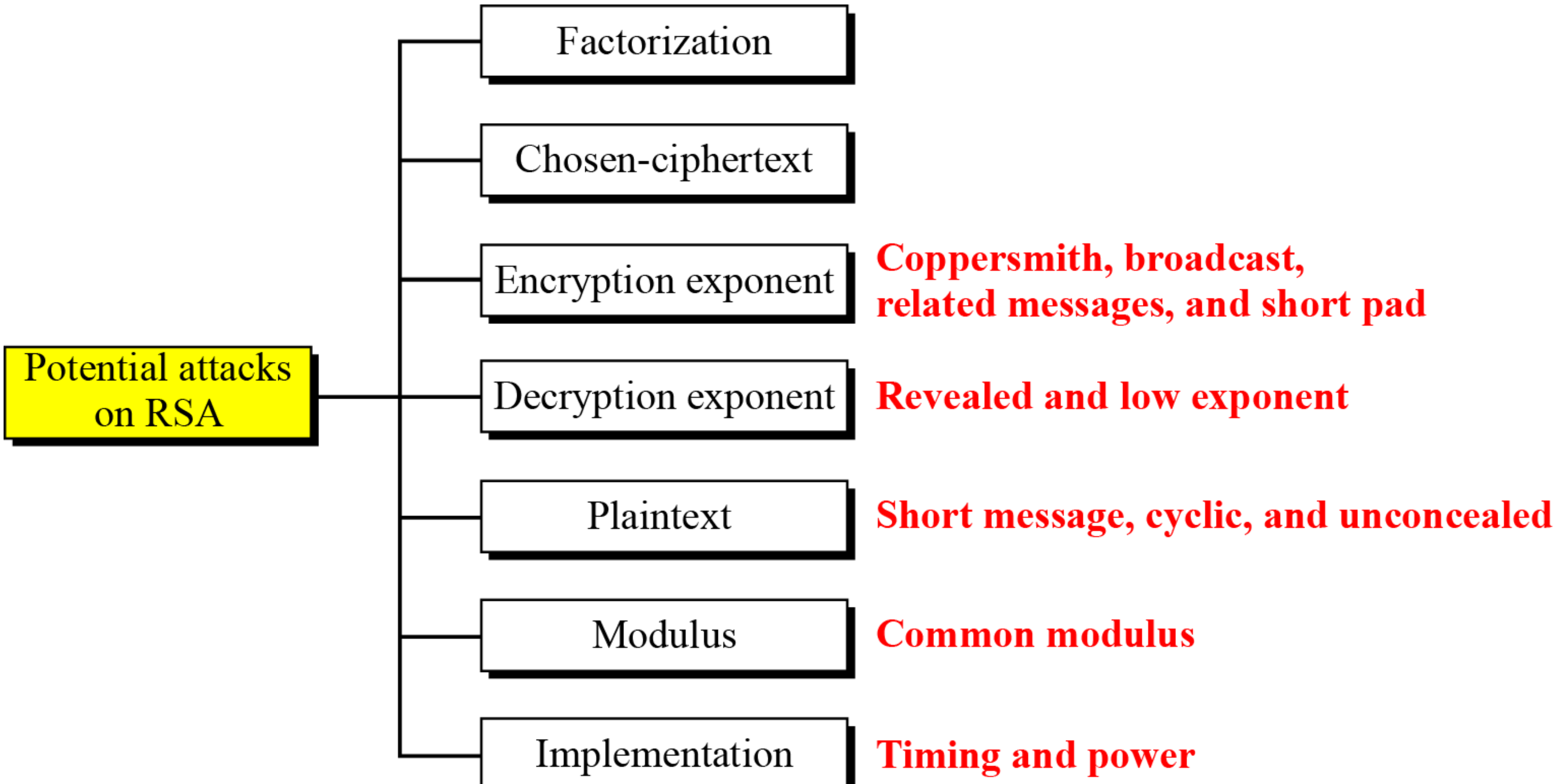


Figure 10.8 *Taxonomy of potential attacks on RSA*

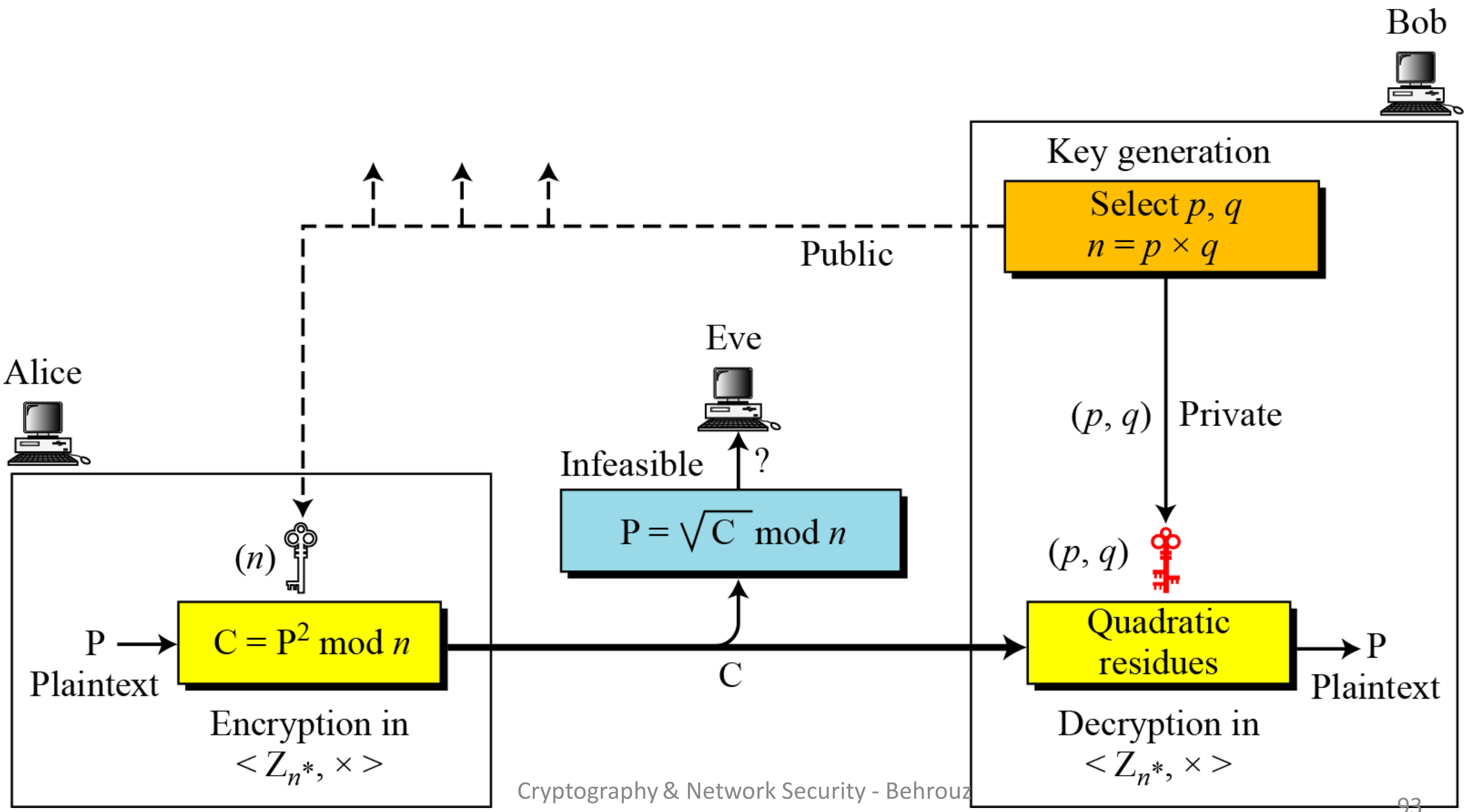


10-3 RABIN CRYPTOSYSTEM

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:

Figure 10.10 *Rabin cryptosystem*



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 ≠ q.
    n ← p × q
    Public_key ← n                        // To be announced publicly
    Private_key ← (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
}
```

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 $C = 24^2 = 93 \bmod 161$, and sends the ciphertext 93 to Bob.

Example 10. 9

5. Bob receives 93 and calculates four values:

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

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

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

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

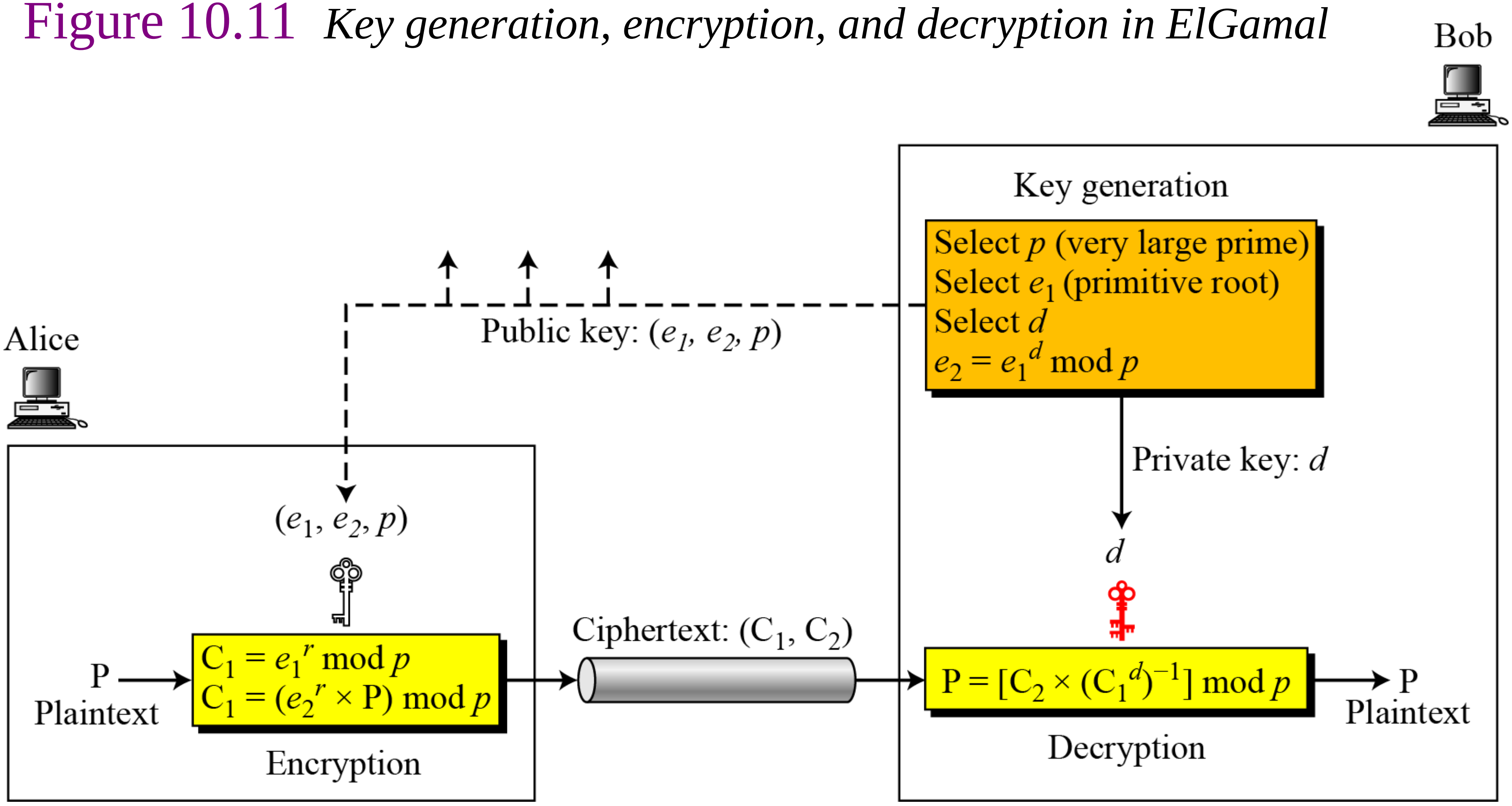
6. Bob takes four possible answers, (a_1, b_1), (a_1, b_2), (a_2, b_1), and (a_2, b_2), and uses the Chinese remainder theorem to find four possible plaintexts: 116, 24, 137, and 45. Note that only the second answer is Alice's plaintext.

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:

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



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 $\mathbf{G} = < \mathbf{Z}_p{}^*, \times >$ such that $1 \le d \le p - 2$

Select e_1 to be a primitive root in the group $\mathbf{G} = < \mathbf{Z}_p{}^*, \times >$

$e_2 \leftarrow e_1{}^d \bmod p$

Public_key $\leftarrow (e_1, e_2, p)$ // To be announced publicly

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

return Public_key and Private_key

}

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
}
```

Algorithm 10.11 *ElGamal decryption*

ElGamal_Decryption (d, p, C_1, C_2) // C_1 and C_2 are the ciphertexts

{

$P \leftarrow [C_2\,(C_1{}^{d})^{-1}] \bmod p$ // P is the plaintext

return 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

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

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

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

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

{

  $x \leftarrow 0$

  while $(x < p)$

  {

    $w \leftarrow (x^3 + ax + b) \bmod p$ // w is y^2

    if (w is a perfect square in $\mathbf{Z}_p$) output $(x, \sqrt{w})$ $(x, -\sqrt{w})$

    $x \leftarrow x + 1$

  {

}

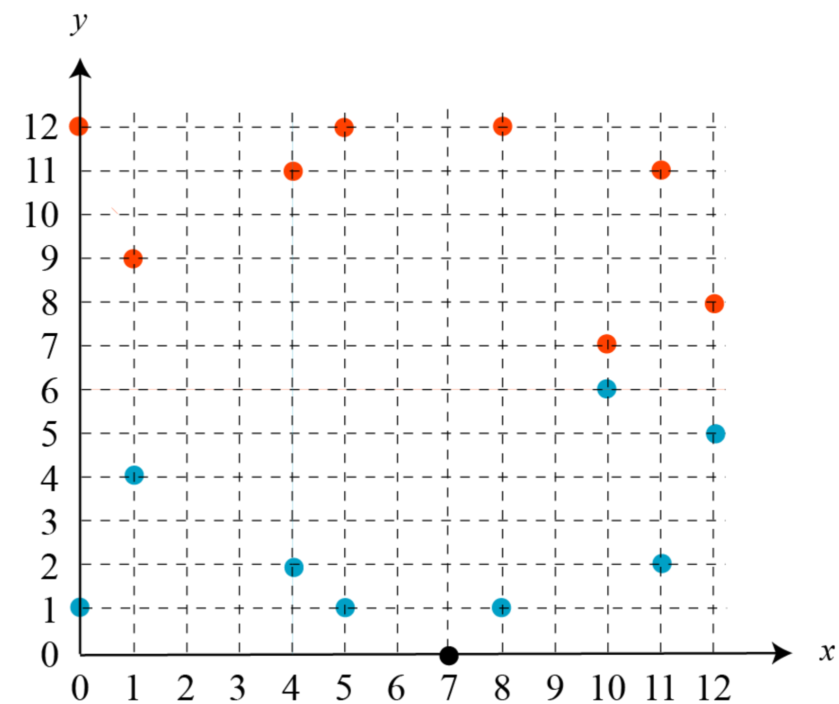
Example 10. 14

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

Points



Graph

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

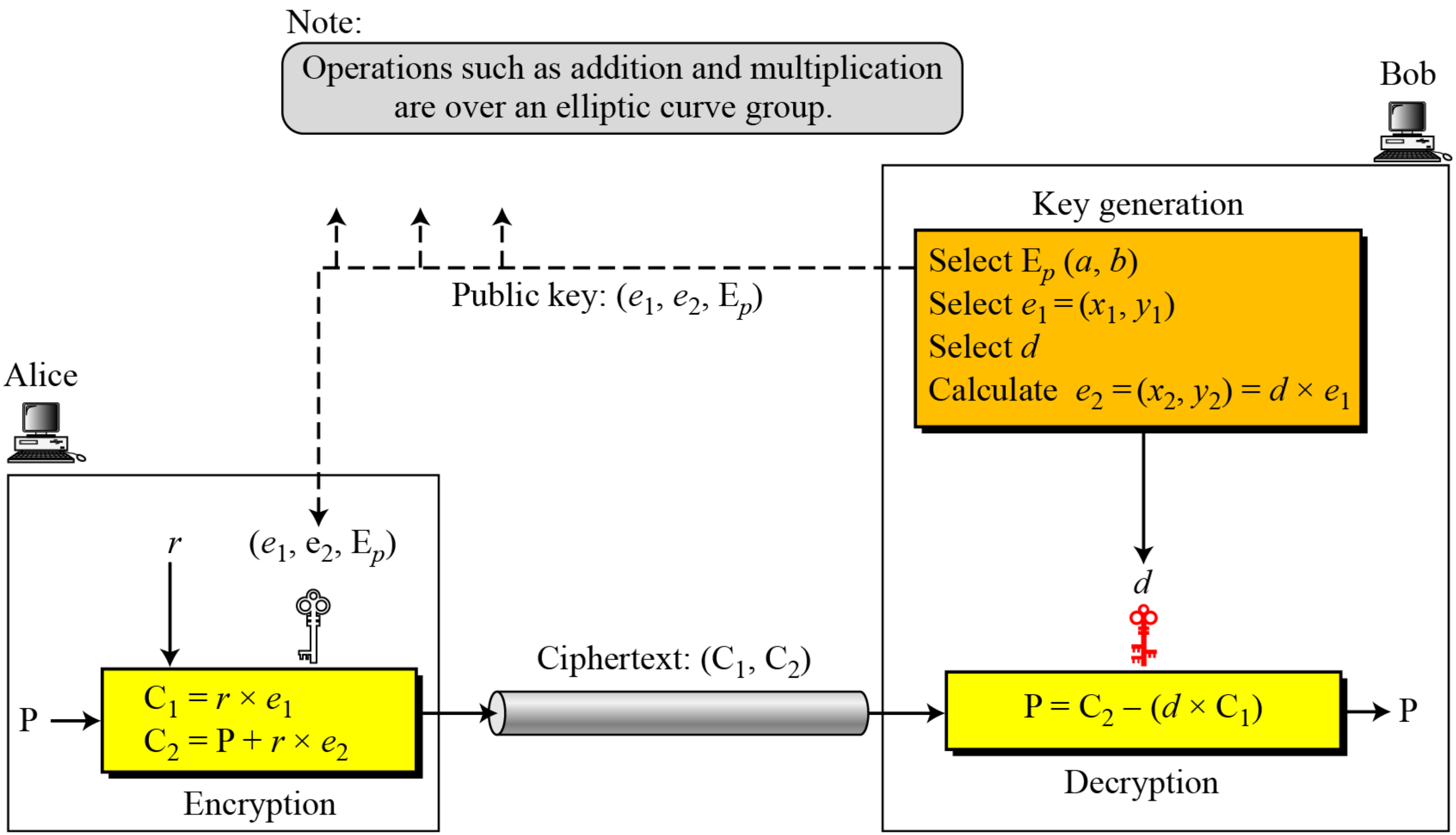
d. *R = (11, 2), which is a point on the curve in Example 10.14.*

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

P + Q?

2P?

Figure 10.16 *ElGamal cryptosystem using the elliptic curve*



Generating Public and Private Keys

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

Encryption

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

Decryption

$$\mathbf{P = C_2 - (d \times C_1)}$$

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).*