TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN ĐIỆN TỬ - VIỄN THÔNG

BỘ MÔN ĐIỆN TỬ HÀNG KHÔNG VŨ TRỤ

Môn học:

LÝ THUYẾT MẬT MÃ

Giảng viên: PGS.TS. Đỗ Trọng Tuấn
Email: dotrongtuan@gmail.com

Mục tiêu học phần

Cung cấp kiến thức cơ bản về mật mã đảm bảo an toàn và bảo mật thông tin:

- ✓ Các phương pháp mật mã khóa đối xứng; Phương pháp mật mã khóa công khai;
- ✓ Các hệ mật dòng và vấn đề tạo dãy giả ngẫu nhiên;
- ✓ Lược đồ chữ ký số Elgamal và chuẩn chữ ký số ECDSA;
- ✓ Độ phức tạp xử lý và độ phức tạp dữ liệu của một tấn công cụ thể vào hệ thống mật mã;
- ✓ Đặc trưng an toàn của phương thức mã hóa;
- ✓ Thám mã tuyến tính, thám mã vi sai và các vấn đề về xây dựng hệ mã bảo mật cho các ứng dụng.

Nội Dung

1. Chương 1. Tổng quan
2. Chương 2. Mật mã khóa đối xứng
3. Chương 3. Hệ mật DES
4. Chương 4. Hệ mật AES
5. **Chương 5. Mật mã khóa công khai**
6. Chương 6. Kỹ thuật quản lý khóa

Tài liệu tham khảo

1. A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, *Handbook of applied cryptography*, CRC Press 1998.
2. B. Schneier, *Applied Cryptography*. John Wiley Press 1996.
3. M. R. A. Huth, *Secure Communicating Systems*, Cambridge University Press 2001.
4. W. Stallings, *Network Security Essentials, Applications and Standards,* Prentice Hall. 2000.

Nhiệm vụ của Sinh viên

1. Chấp hành nội quy lớp học
2. Thực hiện đầy đủ bài tập
3. Nắm vững ngôn ngữ lập trình Matlab

Chương 5. Mật mã khóa công khai

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

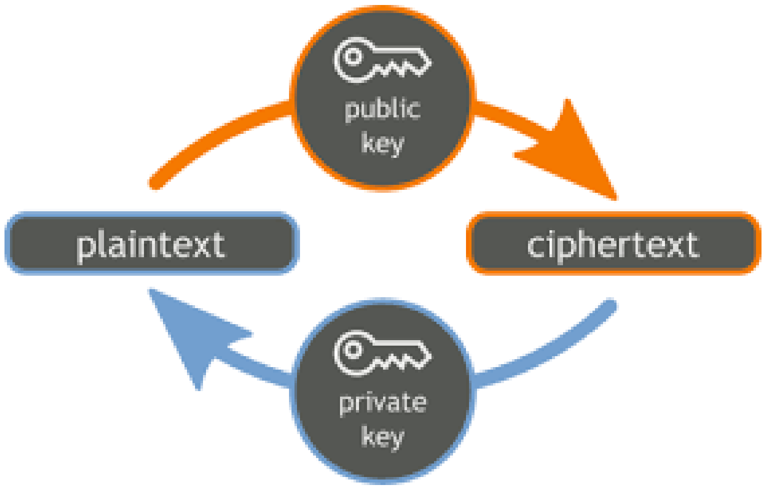
5.2. Hệ mật RSA

5.3. Hệ mật RABIN

5.4. Hệ mật Elgamal

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

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.



5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

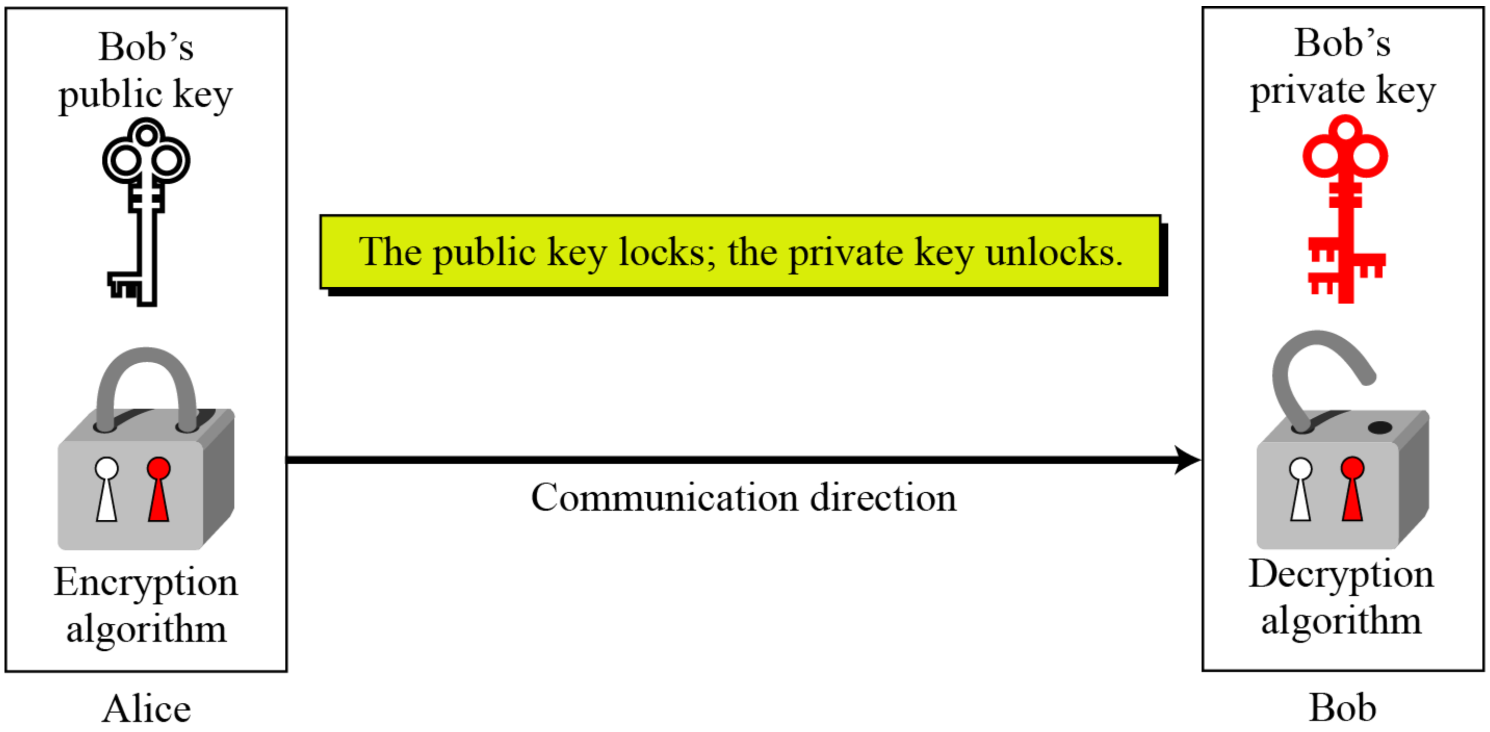
Symmetric and asymmetric-key cryptography will exist in parallel and continue to serve the community. We actually believe that they are complements of each other; the advantages of one can compensate for the disadvantages of the other.

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

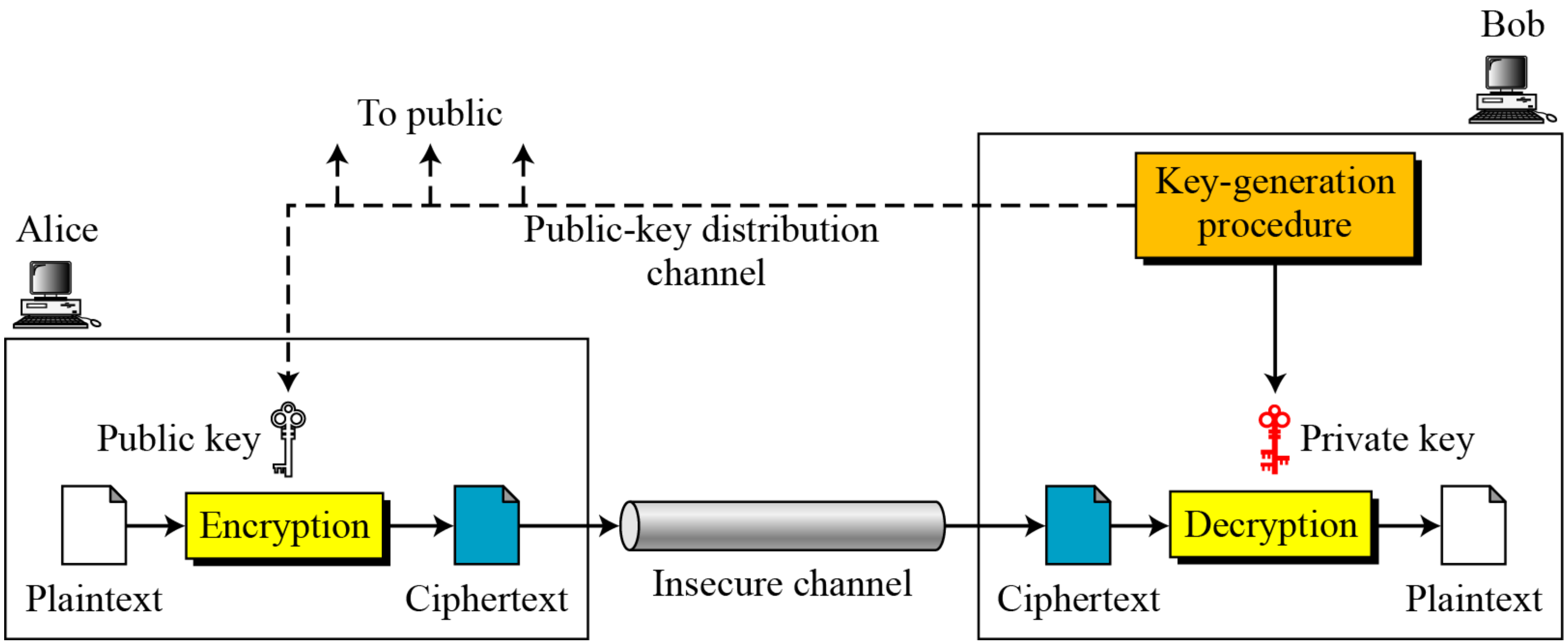
There is a very important fact that is sometimes misunderstood: The advent of asymmetric-key cryptography does not eliminate the need for symmetric-key cryptography.

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

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



5.1. Giới thiệu sơ lược hệ mật mã khóa công khai



5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

Plaintext/Ciphertext

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

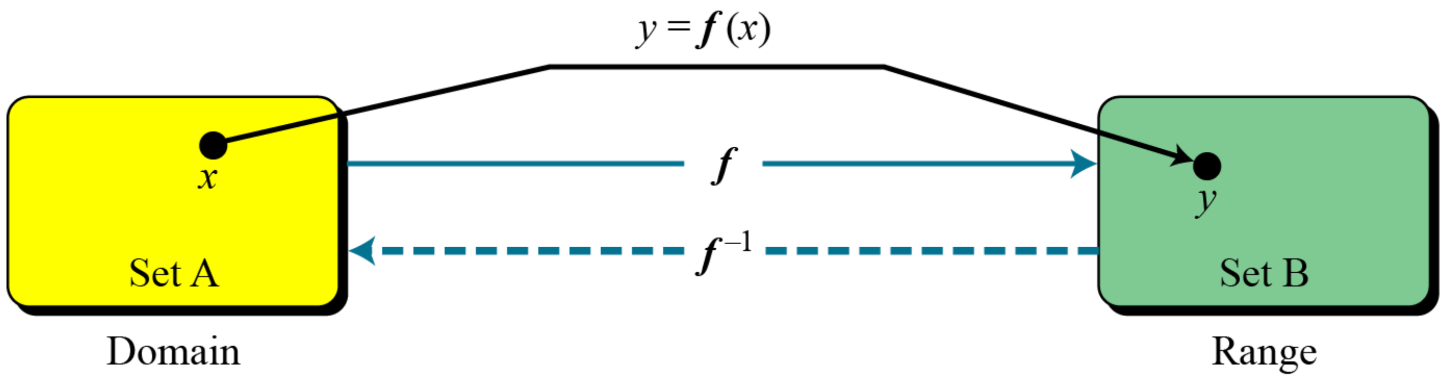
Encryption/Decryption

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

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

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

A function as rule mapping a domain to a range



5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

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.

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

Ví dụ

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.

Ví dụ

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.

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

Knapsack Cryptosystem

Definition

a = [a_1, a_2, …, a_k] and x = [x_1, x_2, …, x_k].

$$\boxed{s = knapsackSum\,(a, x) = x_1 a_1 + x_2 a_2 + \cdots + x_k a_k}$$

Given a and x, it is easy to calculate s. However, given s and a it is difficult to find x.

Superincreasing Tuple

$$\boxed{a_i \geq a_1 + a_2 + \ldots + a_{i-1}}$$

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

knapsacksum and inv_knapsackSum for a superincreasing k-tuple

knapsackSum (x [1 … k], a [1 … k]) { $s \leftarrow 0$ for ($i = 1$ to k) { $s \leftarrow s + a_i \times x_i$ } return s }	**inv_knapsackSum** (s, a [1 … k]) { for ($i = k$ down to 1) { if $s \geq a_i$ { $x_i \leftarrow 1$ $s \leftarrow s - a_i$ } else $x_i \leftarrow 0$ } return x [1 … k] }

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

Ví dụ

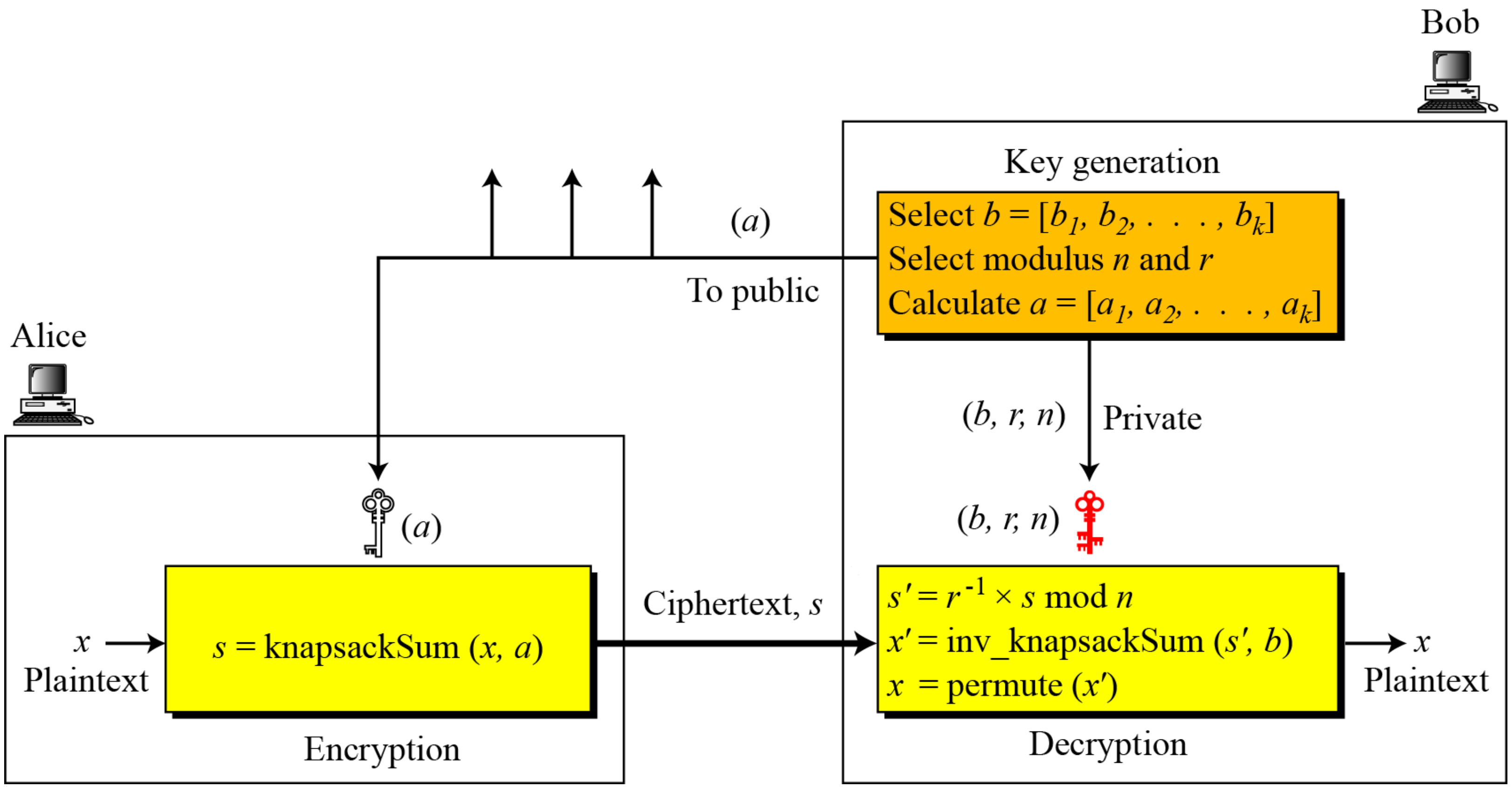
As a very trivial example, assume that *a* = [17, 25, 46, 94, 201,400] and *s* = 272 are given. Table 10.1 shows how the tuple *x* is found using inv_knapsackSum routine in Algorithm 10.1. In this case *x* = [0, 1, 1, 0, 1, 0], which means that 25, 46, and 201 are in the knapsack.

Table 10.1 *Values of i, a_i, s, and x_i in Example 10.3*

i	a_i	s	$s \geq a_i$	x_i	$s \leftarrow s - a_i \times x_i$
6	400	272	false	$x_6 = 0$	272
5	201	272	true	$x_5 = 1$	71
4	94	71	false	$x_4 = 0$	71
3	46	71	true	$x_3 = 1$	25
2	25	25	true	$x_2 = 1$	0
1	17	0	false	$x_1 = 0$	0

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

***Secret Communication with Knapsacks*.**



5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

Key Generation

a. Create a superincreasing k-tuple $b = [b_1, b_2, \ldots, b_k]$

b. Choose a modulus n, such that $n > b_1 + b_2 + \cdots + b_k$

c. Select a random integer r that is relatively prime with n and $1 \leq r \leq n - 1$.

d. Create a temporary k-tuple $t = [t_1, t_2, \ldots, t_k]$ in which $t_i = r \times b_i \bmod n$.

e. Select a permutation of k objects and find a new tuple $a = permute\ (t)$.

f. The public key is the k-tuple a. The private key is n, r, and the k-tuple b.

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

Encryption

Suppose Alice needs to send a message to Bob.

a. Alice converts her message to a k-tuple $x = [x_1, x_2, \ldots, x_k]$ in which x_i is either 0 or 1. The tuple x is the plaintext.

b. Alice uses the *knapsackSum* routine to calculate s. She then sends the value of s as the ciphertext.

Decryption

Bob receives the ciphertext s.

a. Bob calculates $s' = r^{-1} \times s \bmod n$.

b. Bob uses *inv_knapsackSum* to create x'.

c. Bob permutes x' to find x. The tuple x is the recovered plaintext.

5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

Encryption

Suppose Alice needs to send a message to Bob.

a. Alice converts her message to a k-tuple $x = [x_1, x_2, \ldots, x_k]$ in which x_i is either 0 or 1. The tuple x is the plaintext.

b. Alice uses the *knapsackSum* routine to calculate s. She then sends the value of s as the ciphertext.

Decryption

Bob receives the ciphertext s.

a. Bob calculates $s' = r^{-1} \times s \bmod n$.

b. Bob uses *inv_knapsackSum* to create x'.

c. Bob permutes x' to find x. The tuple x is the recovered plaintext.

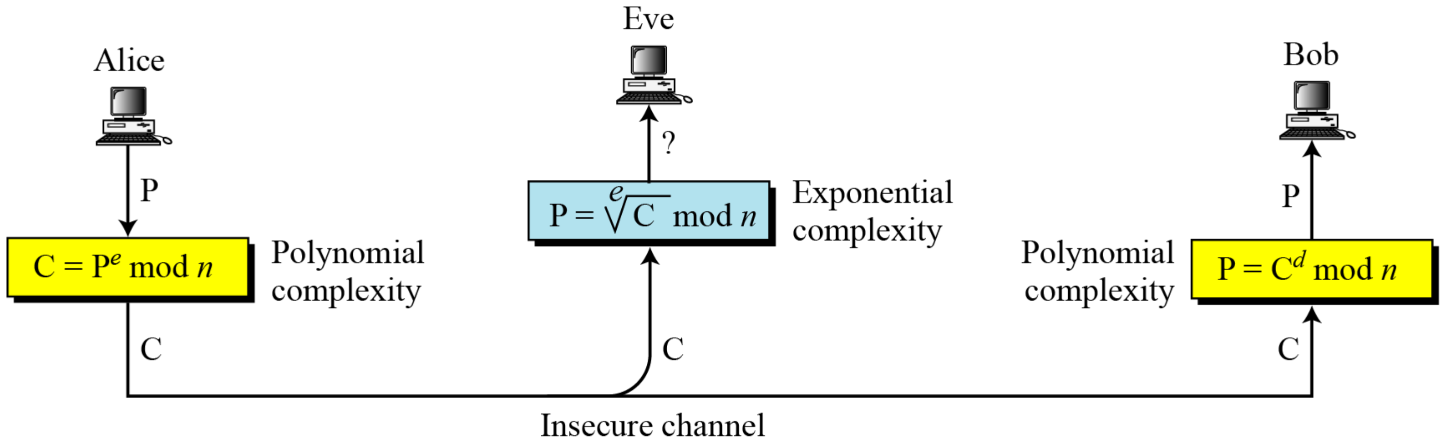
5.1. Giới thiệu sơ lược hệ mật mã khóa công khai

This is a trivial (very insecure) example just to show the procedure.

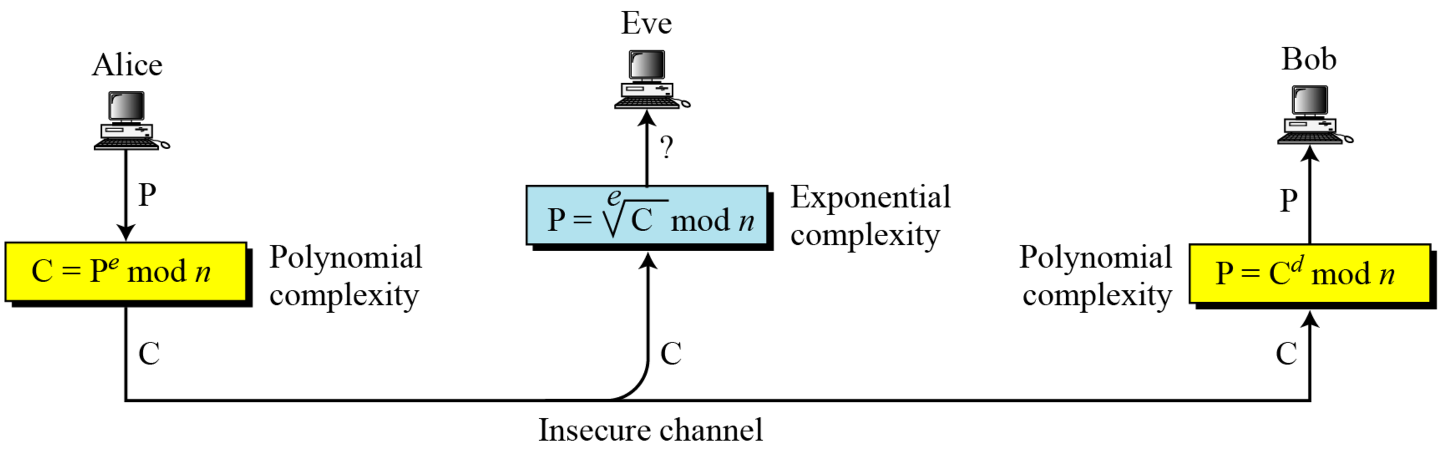
1. Key generation:
   a. Bob creates the superincreasing tuple $b = [7, 11, 19, 39, 79, 157, 313]$.
   b. Bob chooses the modulus $n = 900$ and $r = 37$, and [4 2 5 3 1 7 6] as permutation table.
   c. Bob now calculates the tuple $t = [259, 407, 703, 543, 223, 409, 781]$.
   d. Bob calculates the tuple a = permute (t) = [543, 407, 223, 703, 259, 781, 409].
   e. Bob publicly announces a; he keeps n, r, and b secret.
2. Suppose Alice wants to send a single character "g" to Bob.
   a. She uses the 7-bit ASCII representation of "g", $(1100111)_2$, and creates the tuple $x = [1, 1, 0, 0, 1, 1, 1]$. This is the plaintext.
   b. Alice calculates $s = knapsackSum\ (a, x) = 2165$. This is the ciphertext sent to Bob.
3. Bob can decrypt the ciphertext, $s = 2165$.
   a. Bob calculates $s' = s \times r^{-1} \bmod n = 2165 \times 37^{-1} \bmod 900 = 527$.
   b. Bob calculates $x' = Inv\_knapsackSum\ (s', b) = [1, 1, 0, 1, 0, 1, 1]$.
   c. Bob calculates $x = permute\ (x') = [1, 1, 0, 0, 1, 1, 1]$. He interprets the string $(1100111)_2$ as the character "g".

5.2. Hệ mật RSA

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

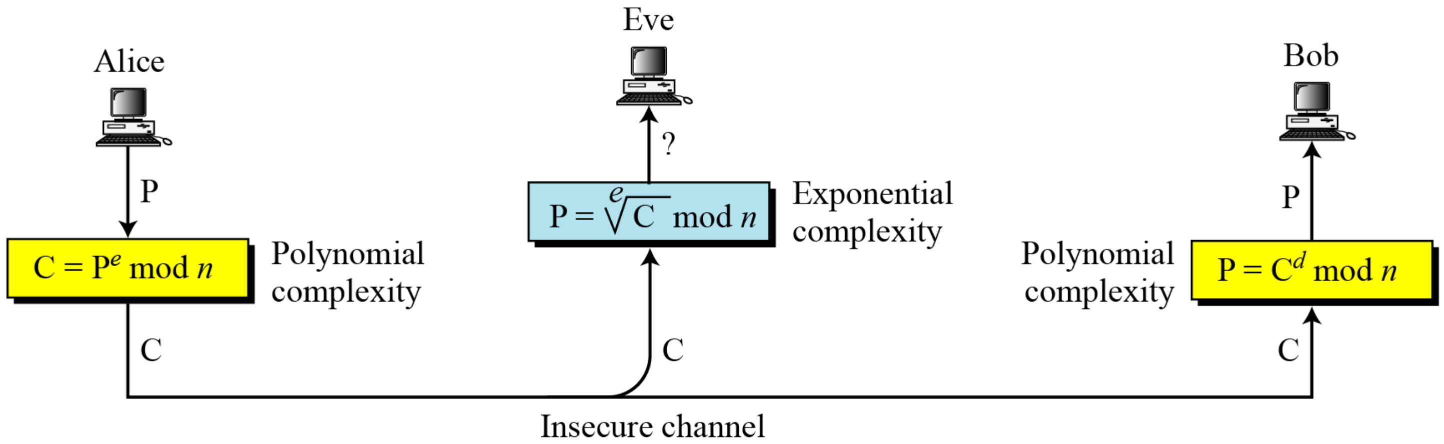


5.2. Hệ mật RSA



RSA uses two exponents, e and d, where e is public and d is private. Suppose P is the plaintext and C is the ciphertext. Alice uses $C = P^e \bmod n$ to create ciphertext C from plaintext P; Bob uses $P = C^d \bmod n$ to retrieve the plaintext sent by Alice. The modulus n, a very large number, is created during the key generation process.

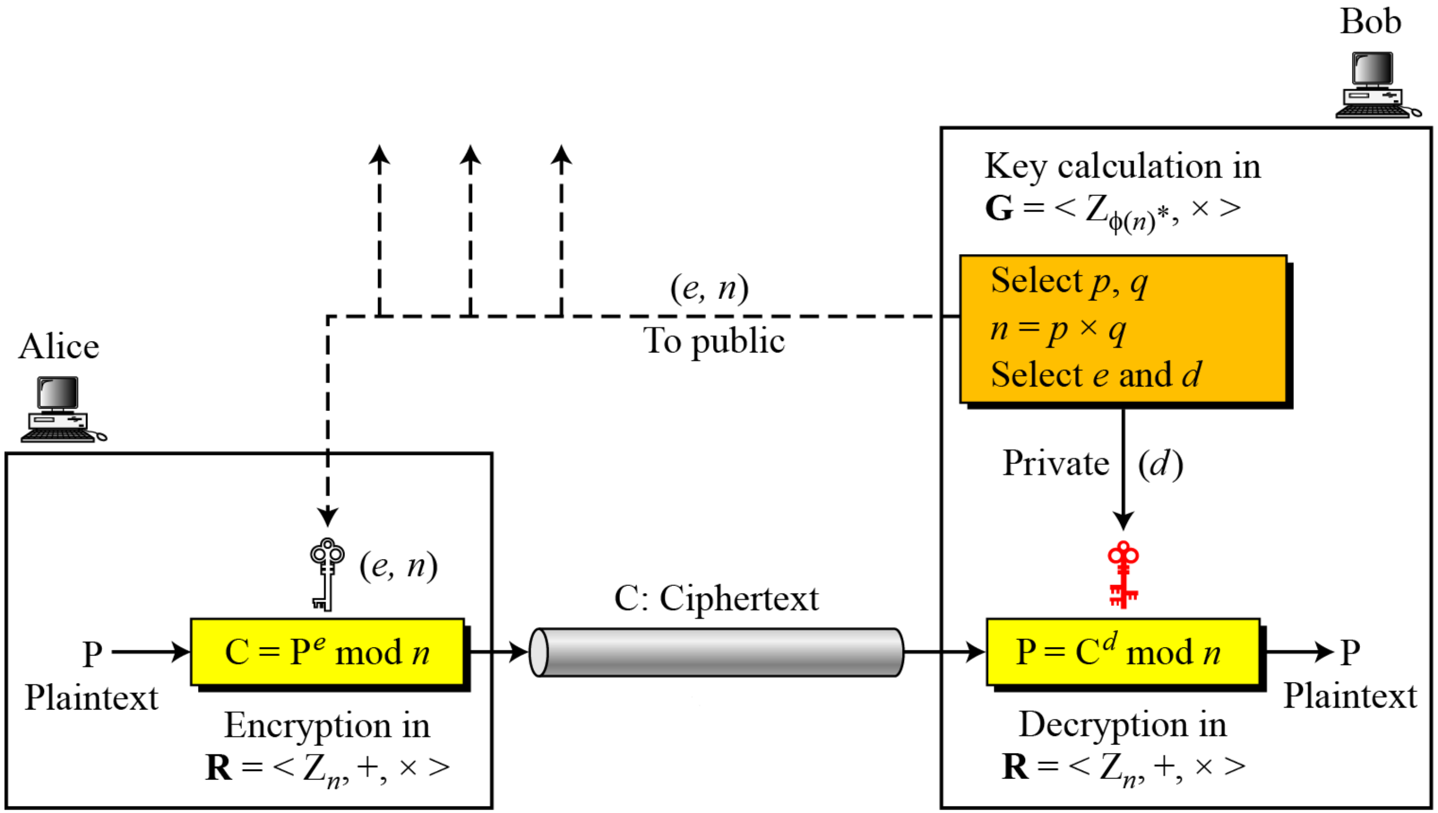
5.2. Hệ mật RSA



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

5.2. Hệ mật RSA

Encryption, decryption, and key generation in RSA



5.2. Hệ mật RSA

Two Algebraic Structures

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

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

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.

5.2. Hệ mật RSA

Encryption/Decryption Ring Encryption and decryption are done using the commutative ring $\mathbf{R} = <Z_n, +, \times>$ with two arithmetic operations: addition and multiplication. In RSA, this ring is public because the modulus n is public. Anyone can send a message to Bob using this ring to do encryption.

Key-Generation Group RSA uses a multiplicative group $\mathbf{G} = <\mathbf{Z}_{\phi(n)}{}^*, \times>$ for key generation. This group supports only multiplication and division (using multiplicative inverses), which are needed for generating public and private keys. This group is hidden from the public because its modulus, $\phi(n)$, is hidden from the public.

cryp

5.2. Hệ mật RSA

Euler's phi-function, $\phi(n)$, 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.

5.2. Hệ mật RSA

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

}

5.2. Hệ mật RSA

Encryption

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.

5.2. Hệ mật RSA

Decryption

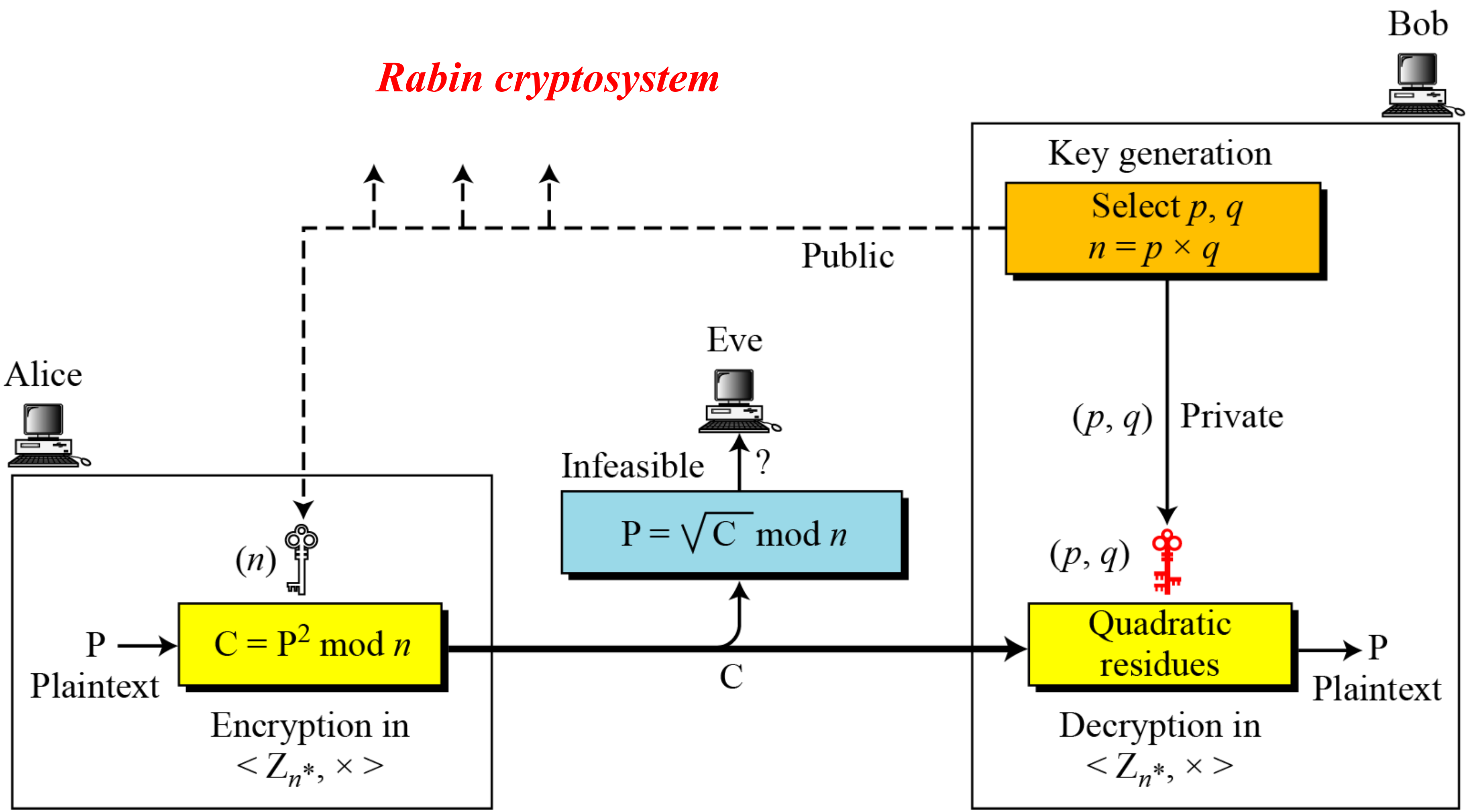
RSA decryption

```
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
}
```

5.3. Hệ mật RABIN

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 $P \equiv C^{1/2} \pmod{n}$.

5.3. Hệ mật RABIN



5.3. Hệ mật RABIN

Key Generation

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
}
```

5.3. Hệ mật RABIN

Encryption

Encryption in Rabin cryptosystem

Rabin_Encryption (n, P) // n is the public key; P is the ciphertext from $\mathbf{Z}_n{}^*$

{

C $\leftarrow$ P^2 mod n // C is the ciphertext

return C

}

Decryption

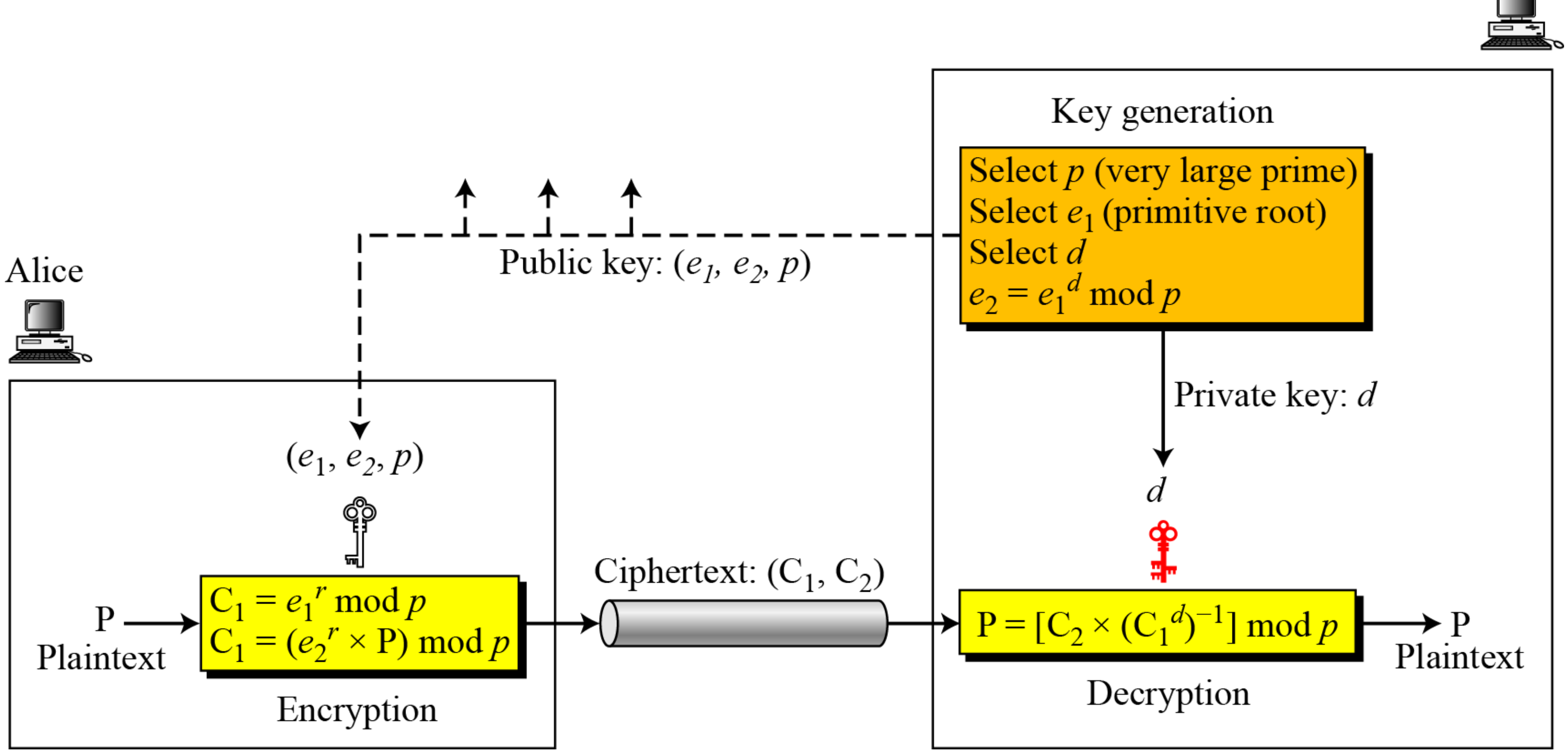
Decryption in Rabin cryptosystem

```
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
}
```

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

5.4. Hệ mật ELGAMAL

Key generation, encryption, and decryption in ElGamal



5.4. Hệ mật ELGAMAL

Key Generation

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 $1 \leq d \leq p - 2$

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

}

5.4. Hệ mật ELGAMAL

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

5.4. Hệ mật ELGAMAL

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
}
```

5.4. Hệ mật ELGAMAL

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
}
```

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