Cryptography (CS-501)

Asymmetric-Key Cryptography

Dr Samayveer Singh
Assistant Professor
Department of Computer Science & Engineering
National Institute Technology Jalandhar, Punjab, India
samays@nitj.ac.in

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.

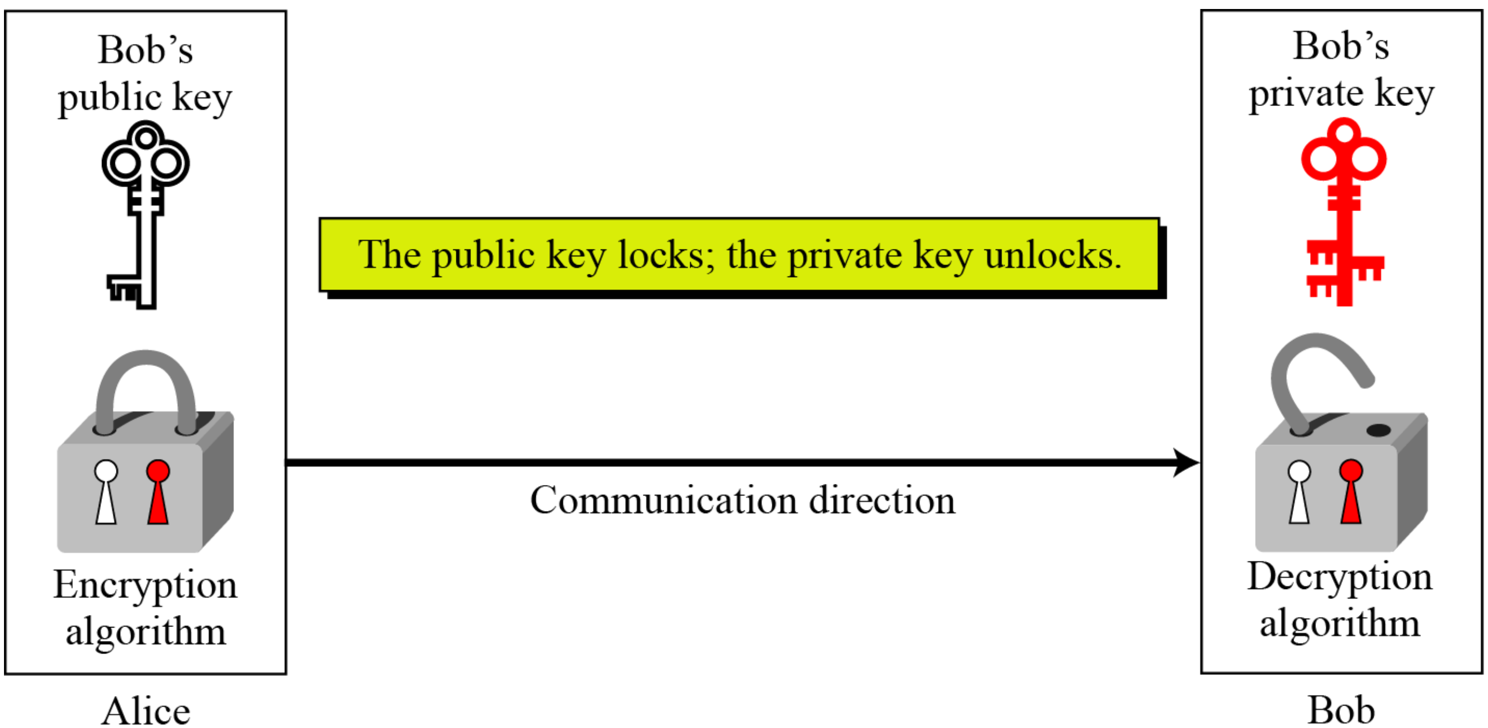
Note

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

10.1.1 Keys

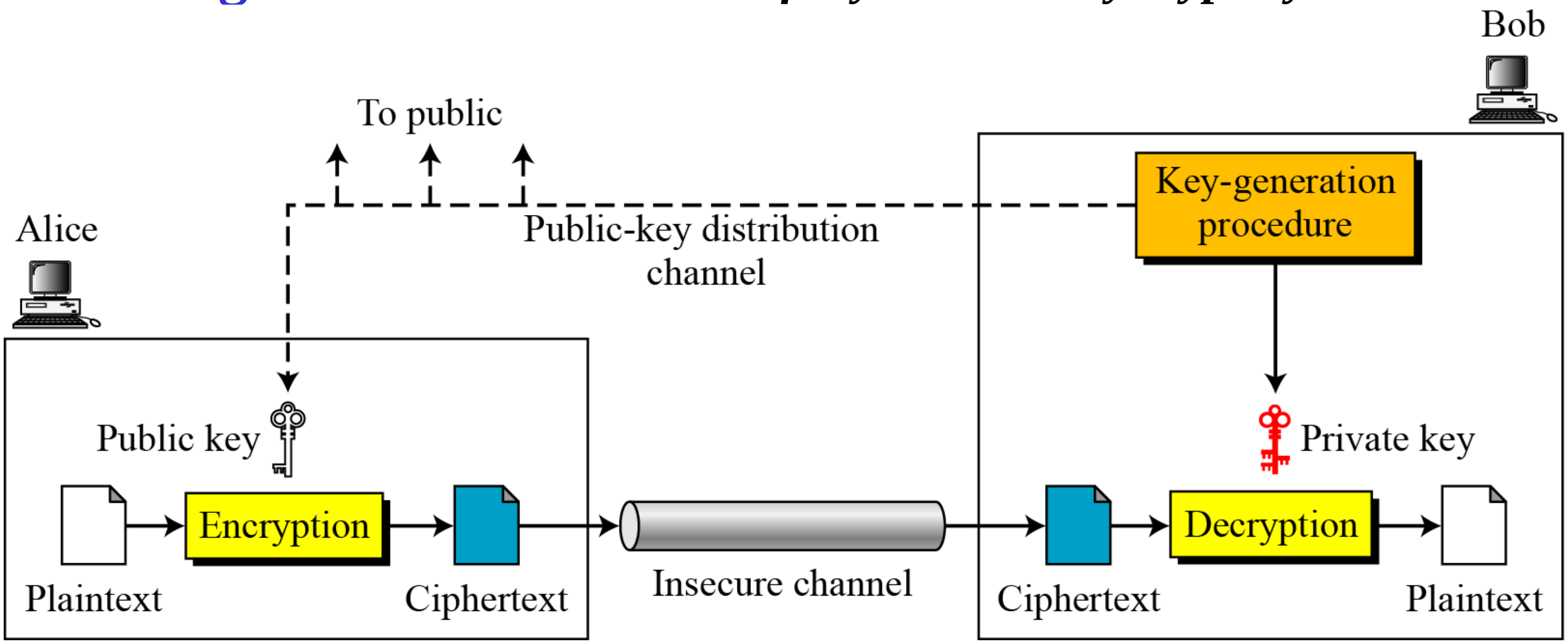
Asymmetric key cryptography uses two separate keys: one private and one public.

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

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

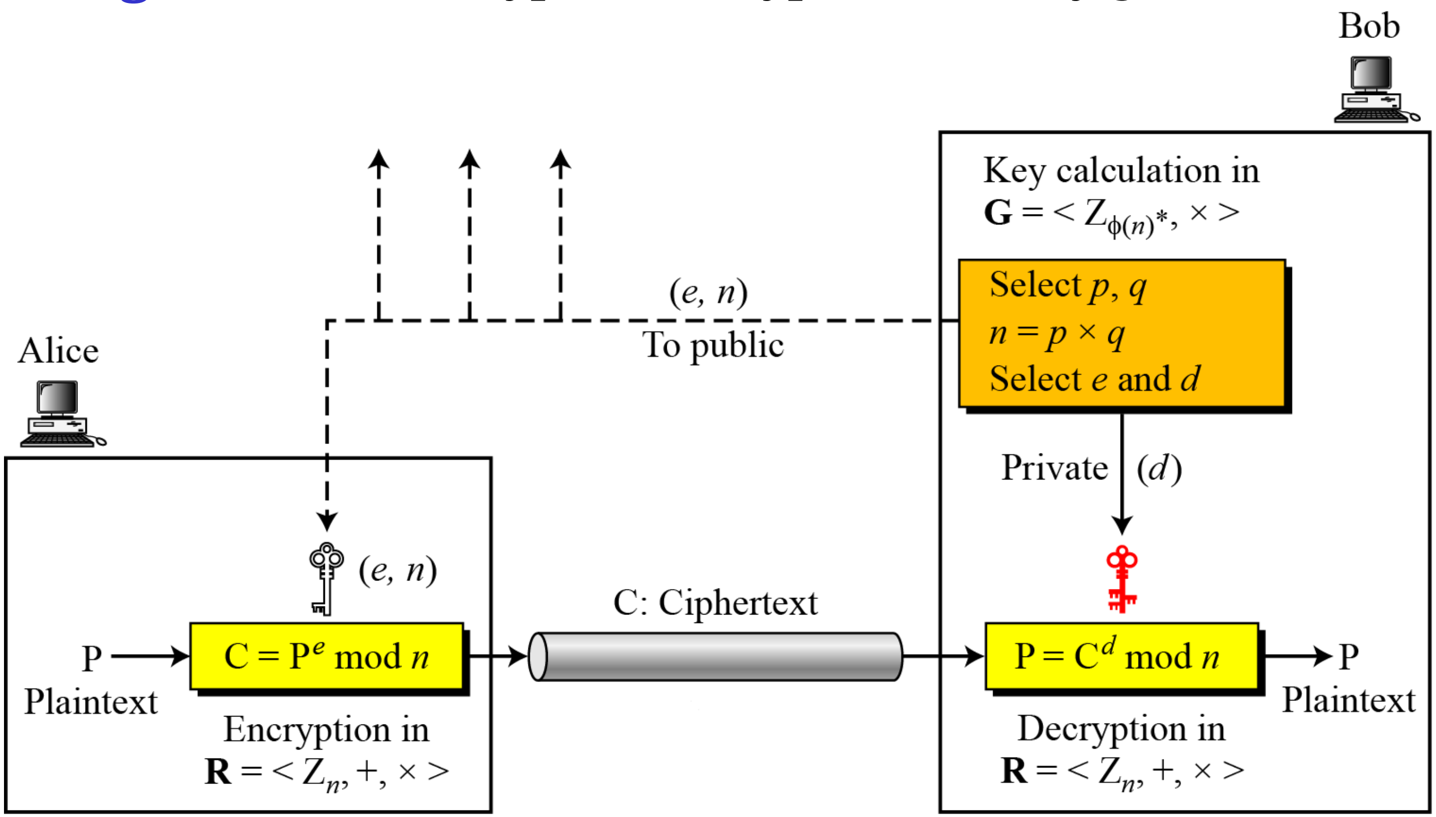
10-2 RSA CRYPTOSYSTEM

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 Procedure

Figure 10.3 ***Encryption, decryption, and key generation in RSA***



Two Algebraic Structures

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

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

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.

10.2.1 Continued

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

}

10.2.1 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.

10.2.1 Continued

Decryption

Algorithm 10.4 *RSA decryption*

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

10.2.2 Some Trivial Examples

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

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$. The example shows the use of these keys for a plaintext input of $M = 88$. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

10.2.2 Some Trivial Examples

$$
\begin{aligned}
88^7 \bmod 187 &= [(88^4 \bmod 187) \times (88^2 \bmod 187) \\
&\quad \times (88^1 \bmod 187)] \bmod 187 \\
88^1 \bmod 187 &= 88 \\
88^2 \bmod 187 &= 7744 \bmod 187 = 77 \\
88^4 \bmod 187 &= 59{,}969{,}536 \bmod 187 = 132 \\
88^7 \bmod 187 &= (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11
\end{aligned}
$$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$
\begin{aligned}
11^{23} \bmod 187 &= [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \\
&\quad \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187 \\
11^1 \bmod 187 &= 11 \\
11^2 \bmod 187 &= 121 \\
11^4 \bmod 187 &= 14{,}641 \bmod 187 = 55 \\
11^8 \bmod 187 &= 214{,}358{,}881 \bmod 187 = 33 \\
11^{23} \bmod 187 &= (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 \\
&= 79{,}720{,}245 \bmod 187 = 88
\end{aligned}
$$

10.2.2 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 $e \times d \bmod 60 = 1$ (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

10.2.2 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

10.2.2 Some Trivial Examples

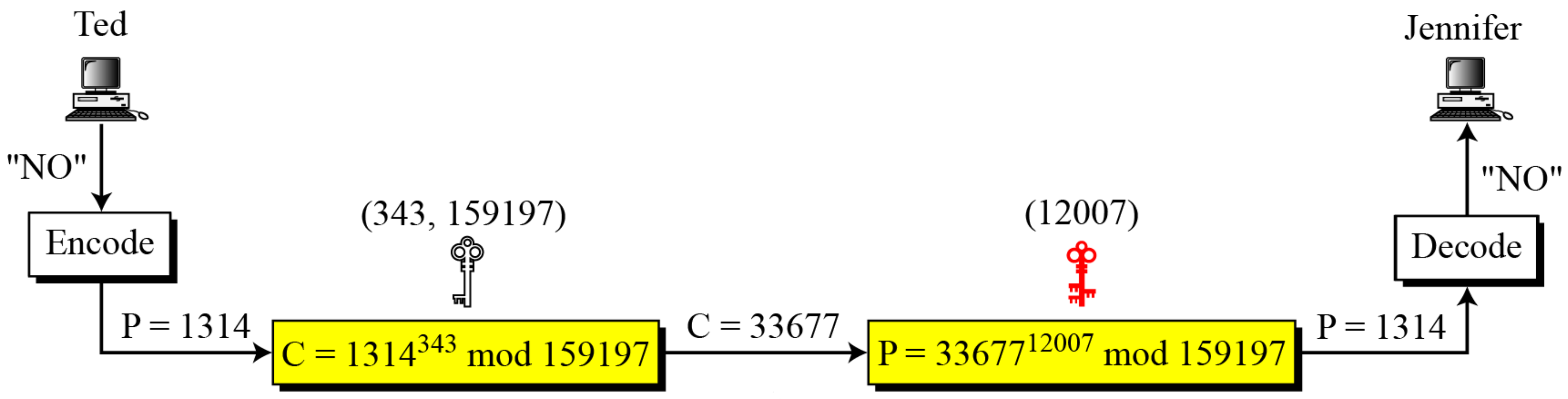
Example 10. 7

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.

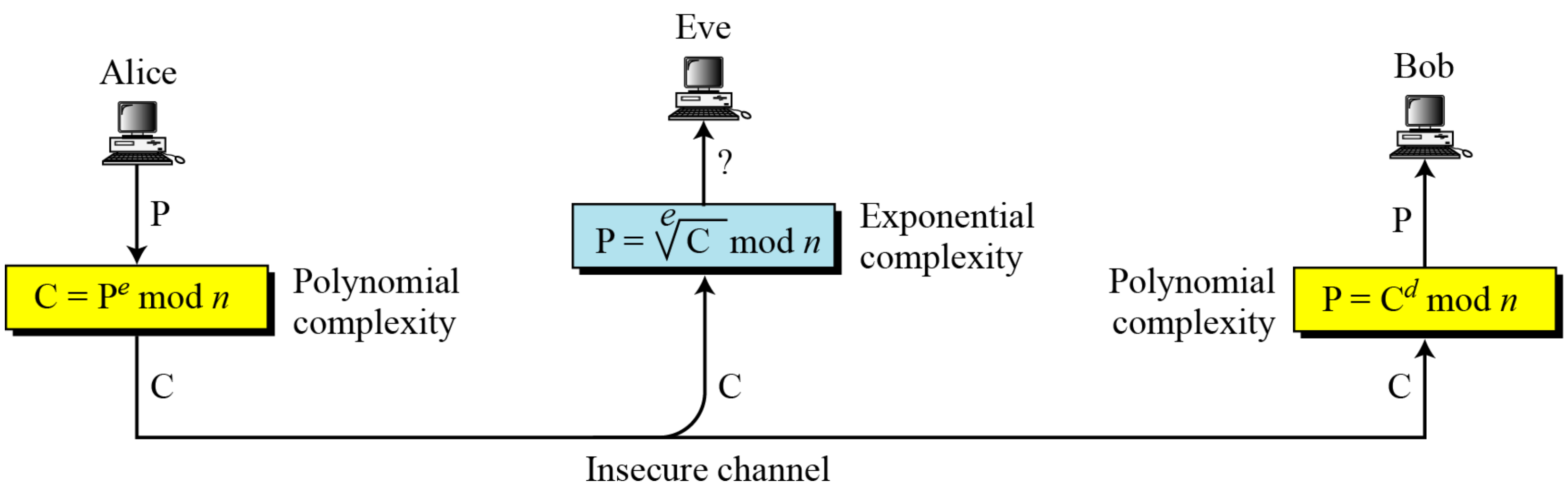
10.2.2 Continued

Figure 10.4 ***Encryption and decryption in Example 10.7***



10.2.3 Introduction

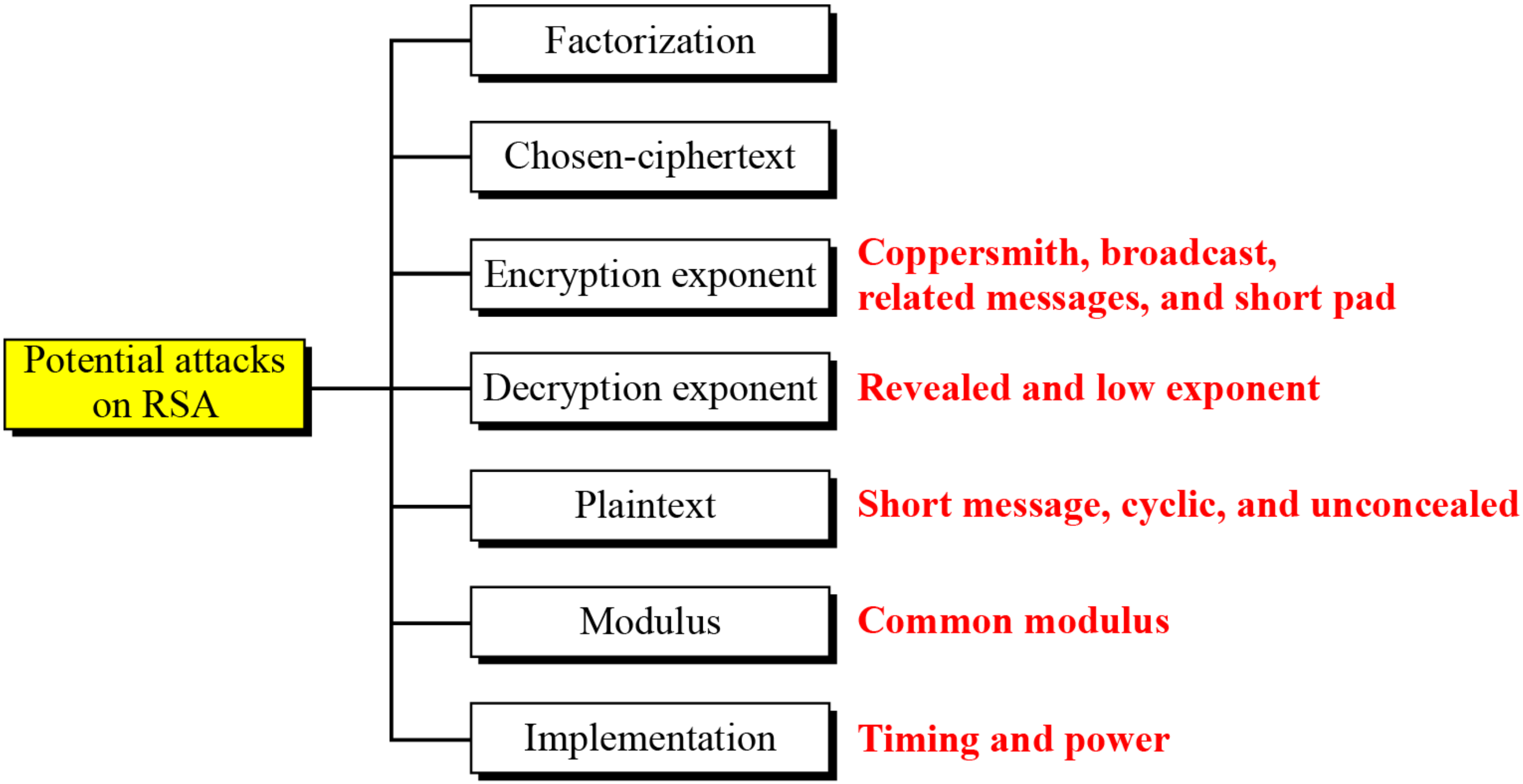
Figure 10.5 *Complexity of operations in RSA*



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

10.2.4 Attacks on RSA

Figure 10.8 *Taxonomy of potential attacks on RSA*



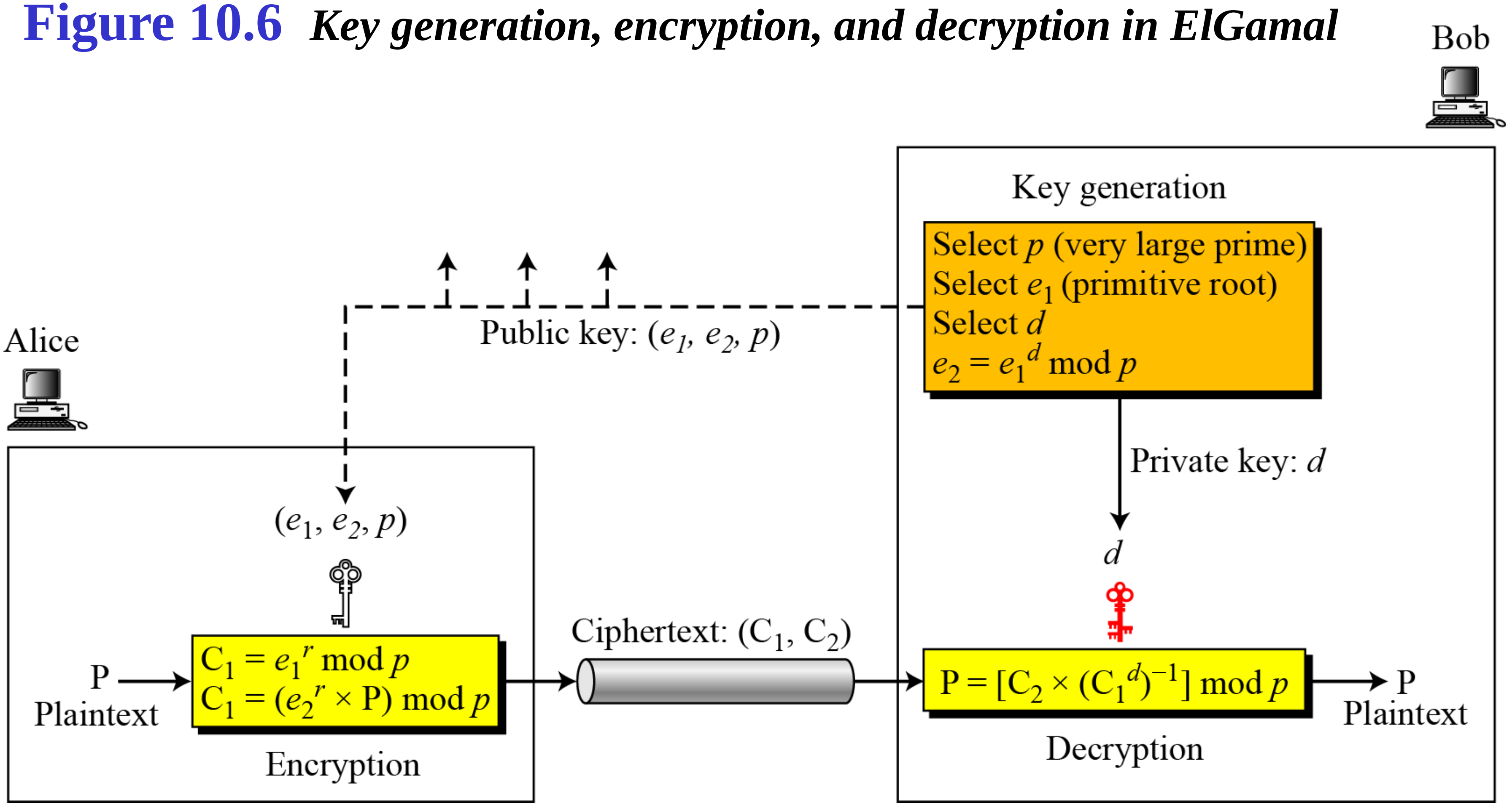
10-4 ELGAMAL CRYPTOSYSTEM

After RSA another public-key cryptosystem is ElGamal. ElGamal is based on the discrete logarithm problem.

Topics discussed in this section:

10.4.1 ElGamal Cryptosystem
10.4.2 Procedure
10.4.3 Proof

Figure 10.6 *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 $\mathbf{G} = \langle \mathbf{Z}_p{}^*, \times \rangle$ such that $1 \le d \le p - 2$

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

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

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

}

10.4.2 Continued

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
}
```

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

Note

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

10.4.3 Continued

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

10.4.3 Continued

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
$\mathbf{C_1 =}$	887297069383528471022570471492275663120260067256562125018188351429 417223599712681114105363661705173051581533189165400973736355080295 736788569060619152881
$\mathbf{C_2 =}$	708454333048929944577016012380794999567436021836192446961774506921 244696155165800779455593080345889614402408599525919579209721628879 6813505827795664302950

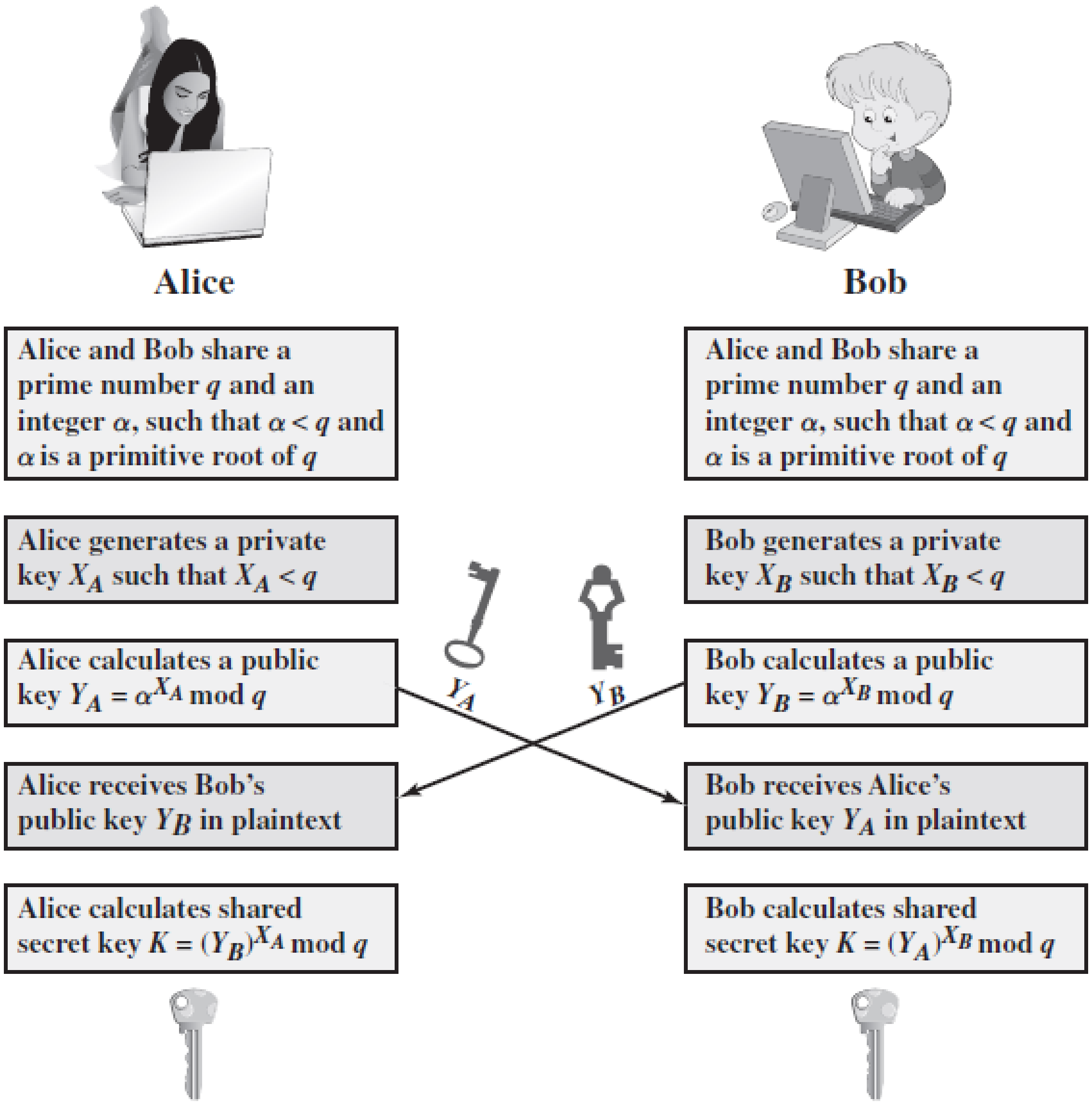
Bob calculates the plaintext $P = C_2 \times ((C_1)^d)^{-1} \bmod p = 3200 \bmod p$.

P =	3200

10-5 Diffie Hellman Key Exchange Algorithm

The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent asymmetric encryption of messages.

10.5 Procedure



10.5 Procedure

Key exchange is based on the use of the prime number $q = 353$ and a primitive root of 353, in this case $\alpha = 3$. A and B select private keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.

B computes $Y_B = 3^{233} \bmod 353 = 248$.

After they exchange public keys, each can compute the common secret key:

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

We assume an attacker would have available the following information:

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