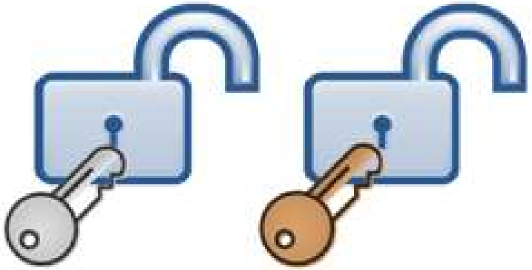
Chapter 10

Asymmetric-Key Cryptography



Chapter 10

Objectives

- Present asymmetric-key cryptography.
- Distinguish between symmetric-key cryptography and asymmetric-key cryptography.
- Introduce trapdoor one-way functions and their use in asymmetric-key cryptosystems
- Discuss the RSA cryptosystem

10-1 INTRODUCTION

The advent of asymmetric-key cryptography does not eliminate the need for symmetric-key cryptography. 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:

Keys
General Idea
Asymmetric Cryptography Practices
Symmetric Cryptography Versus Asymmetric Cryptography
Trapdoor One-Way Function

10.1.1 Keys

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

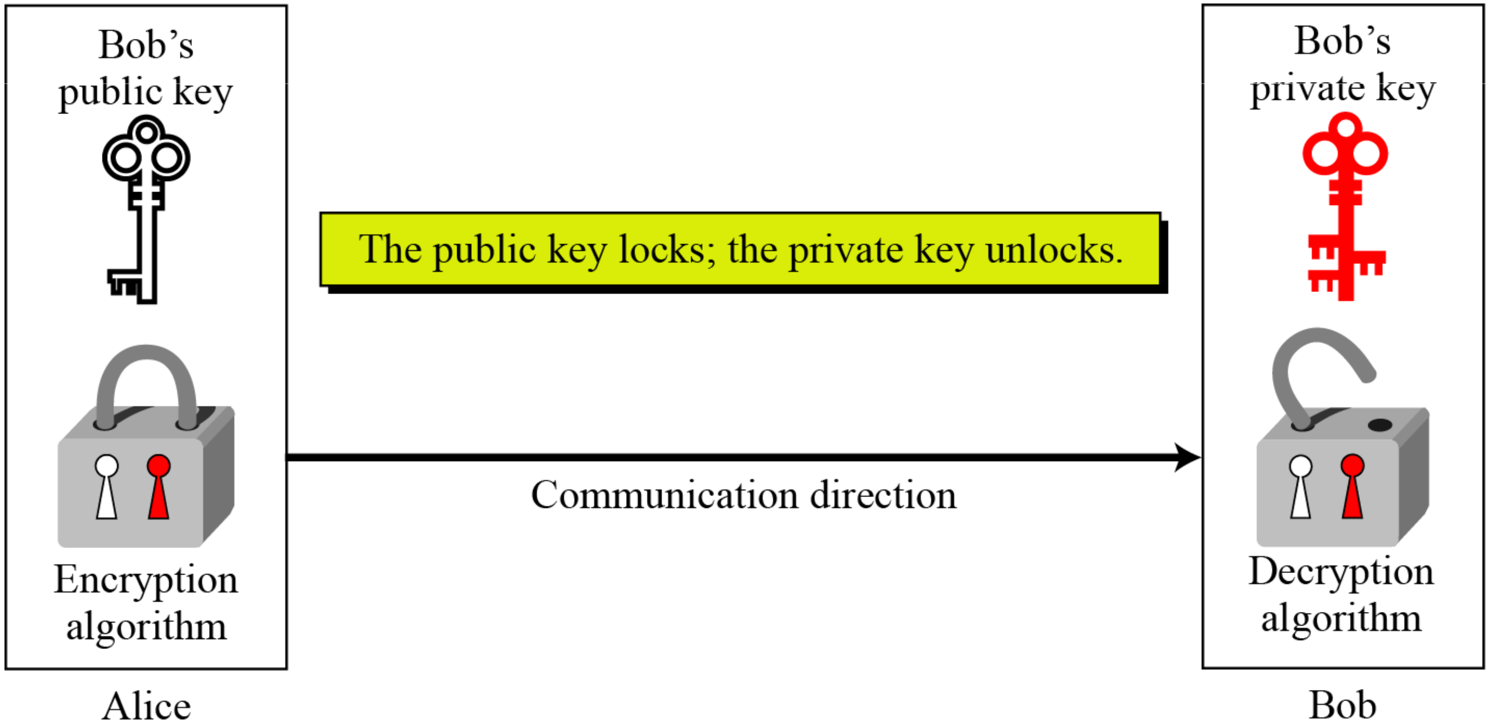
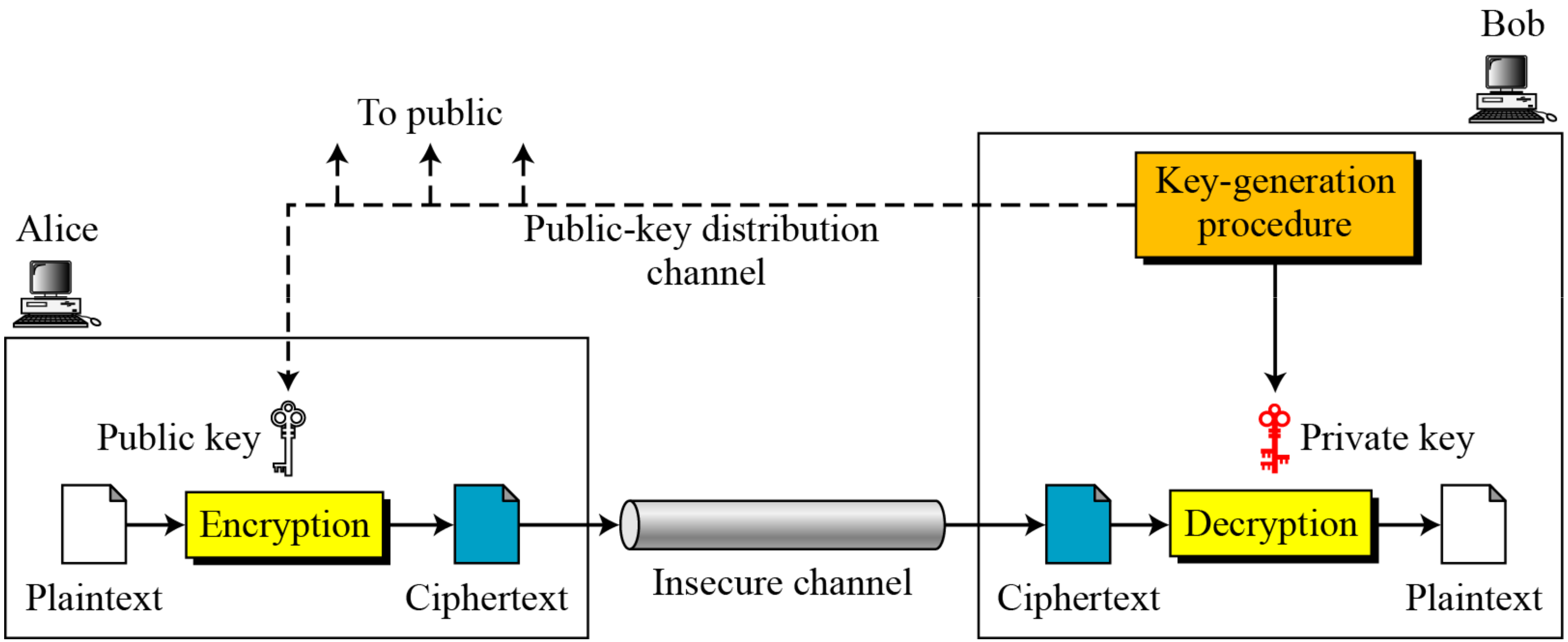


Figure 10.1 *Locking and unlocking in asymmetric-key cryptosystem*

10.1.2 General Idea

Figure 10.2 General idea of asymmetric-key cryptosystem



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

Asymmetric Cryptography Practices

Action	Whose Key to Use	Which Key to Use	Explanation
Bob wants to send Alice an encrypted message	Alice's key	Public key	Whenever an encrypted message is to be sent the recipient's key is always used and never the sender's keys.
Alice wants to read an encrypted message sent by Bob	Alice's key	Private key	An encrypted message can only be read by using the recipient's private key.
Bob wants to send a copy to himself of the encrypted message that he sent to Alice	Bob's key	Public key to encrypt Private key to decrypt	An encrypted message can only be read by the recipient's private key. Bob would need to encrypt it with his own public key and then use his private key to decrypt it.
Bob receives an encrypted reply message from Alice	Bob's key	Private key	The recipient's private key is used to decrypt received messages.
Bob wants Susan to read Alice's reply message that he received	Susan's key	Public key	The message should be encrypted with Susan's key for her to decrypt and read it with her private key.

Symmetric Cryptography Versus Asymmetric Cryptography

Note-1

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

Note-2

In symmetric-key cryptography system, the number of keys needed for each user is 1.
In asymmetric-key cryptography system, the number of keys needed for each user is 2.

Symmetric Cryptography Versus Asymmetric Cryptography

Note-3

In symmetric-key cryptography, symbols in plaintext and ciphertext are permuted or substituted. In asymmetric-key cryptography, plaintext and ciphertext are treated as integers.

Note-4

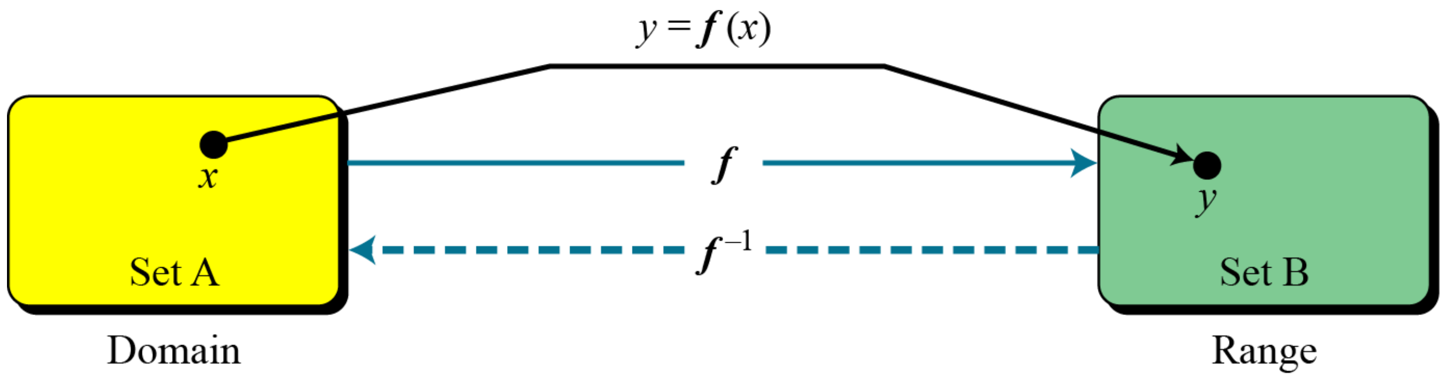
Symmetric-key cryptography is appropriate for long messages, and the speed of encryption/decryption is fast. Asymmetric-key cryptography is appropriate for short messages, and the speed of encryption/decryption is slow.

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*



10.1.4 Continued

One-Way Function (OWF)

1. *f is easy to compute* → $y = f(x)$
2. f^{-1} *is difficult to compute* → $x = f^{-1}(y)$

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.

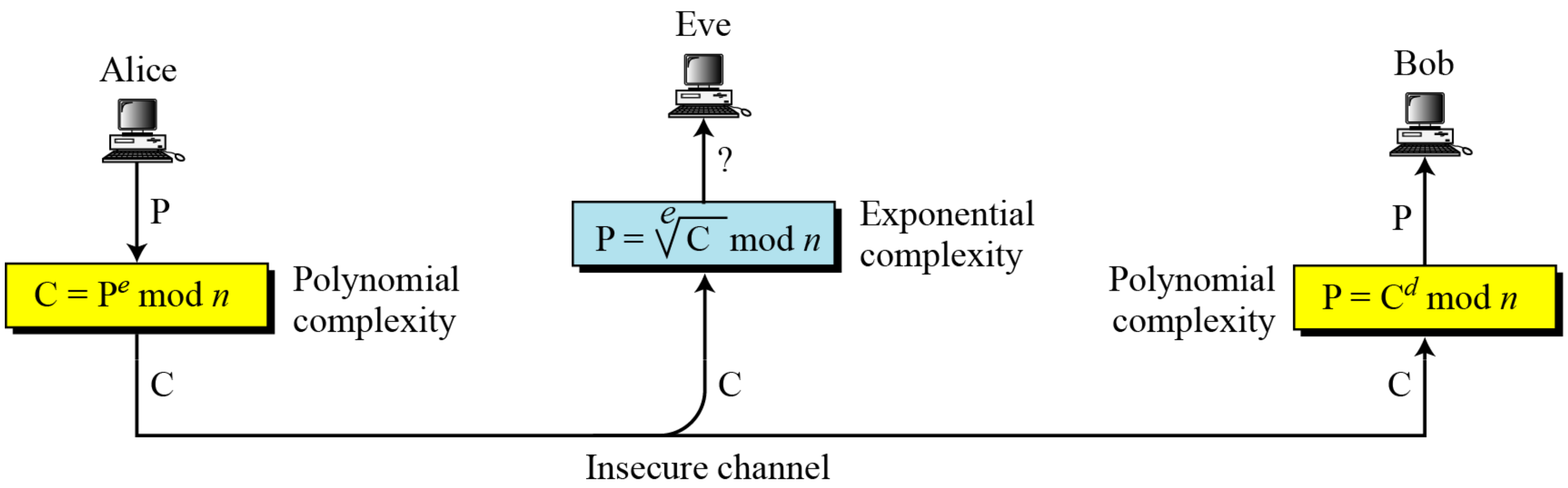
10-2 RSA CRYPTOSYSTEM

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

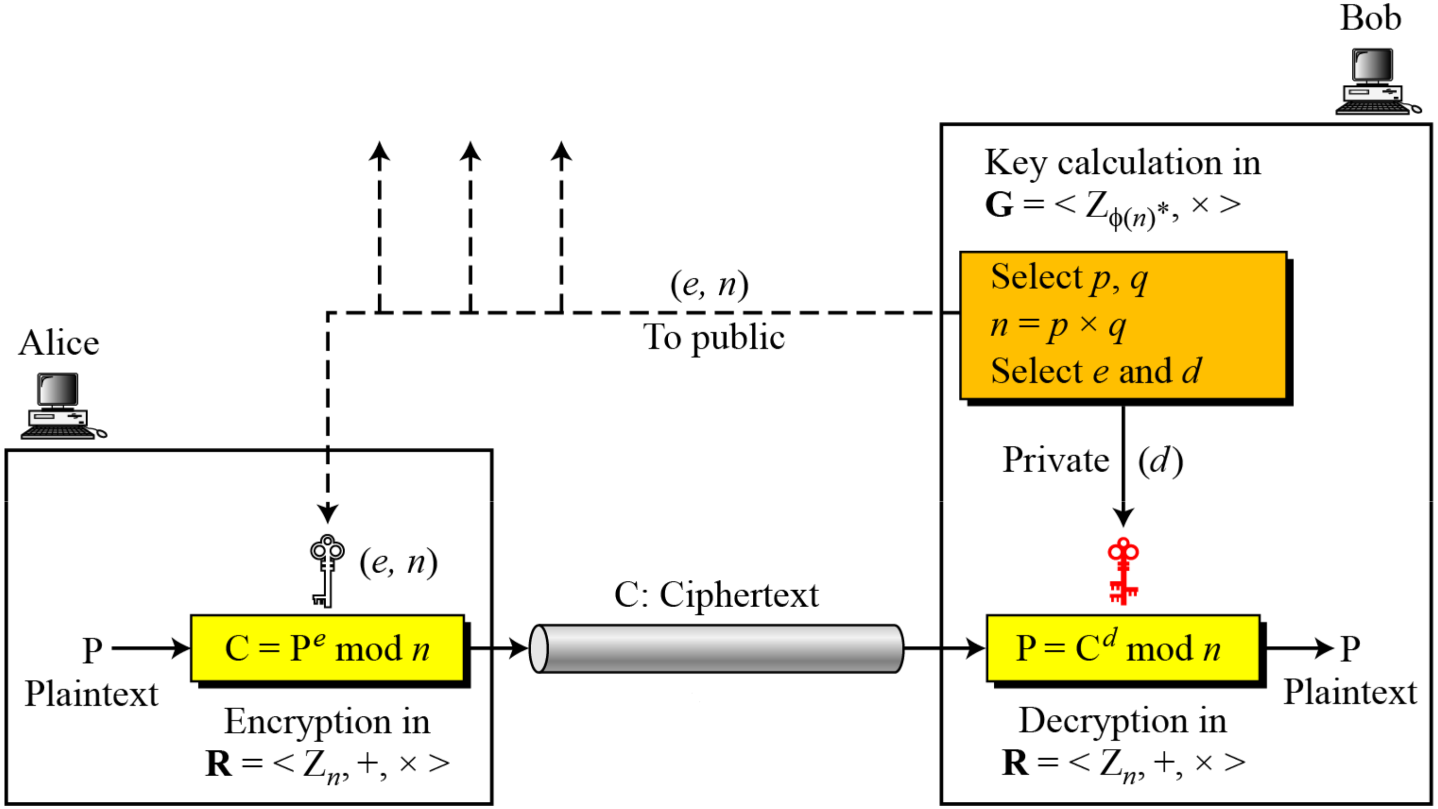
Figure 10.5 ***Complexity of operations in RSA***



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*



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

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

10.2.2 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

}

10.2.2 Continued

When Alice wants Bob to send her a message, she:

- Selects two (large) primes p, q, **TOP SECRET,**
- Computes $n = pq$ and $\phi(n) = (p\text{-}1)(q\text{-}1)$. $\phi(n)$ is **TOP SECRET**.
- Selects an integer e, $1 < e < \phi(n)$, such that $\gcd(e, \phi(n)) = 1$,
- Computes d, such that $d*e \pmod{\phi(n)}=1$, d also **TOP SECRET**,
- Gives **public key (e, n)** to Bob, and keeps her **private key (d, n)**.

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.

If the plaintext P is larger than n, then P has to be encrypted letter by letter.

10.2.2 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
}
```

10.2.3 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

10.2.2 Continued

Example 10. 5 (cont.)

Calculate $5^{13} \bmod 77$:

$5^1 = 5 \bmod 77 = 5$

$5^2 = 25 \bmod 77 = 25$

$5^4 = 625 \bmod 77 = 9$

$5^8 = 390625 \bmod 77 = 4$

$5^{13} = 5^1 * 5^4 * 5^8 = 180 \bmod 77 = 26$

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

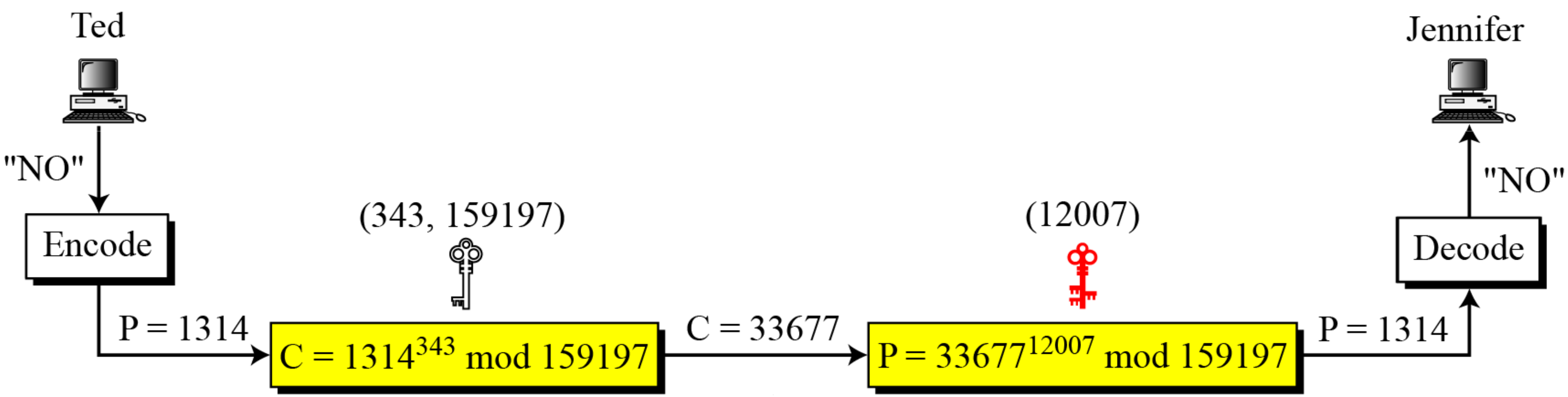
Example 10. 7

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.

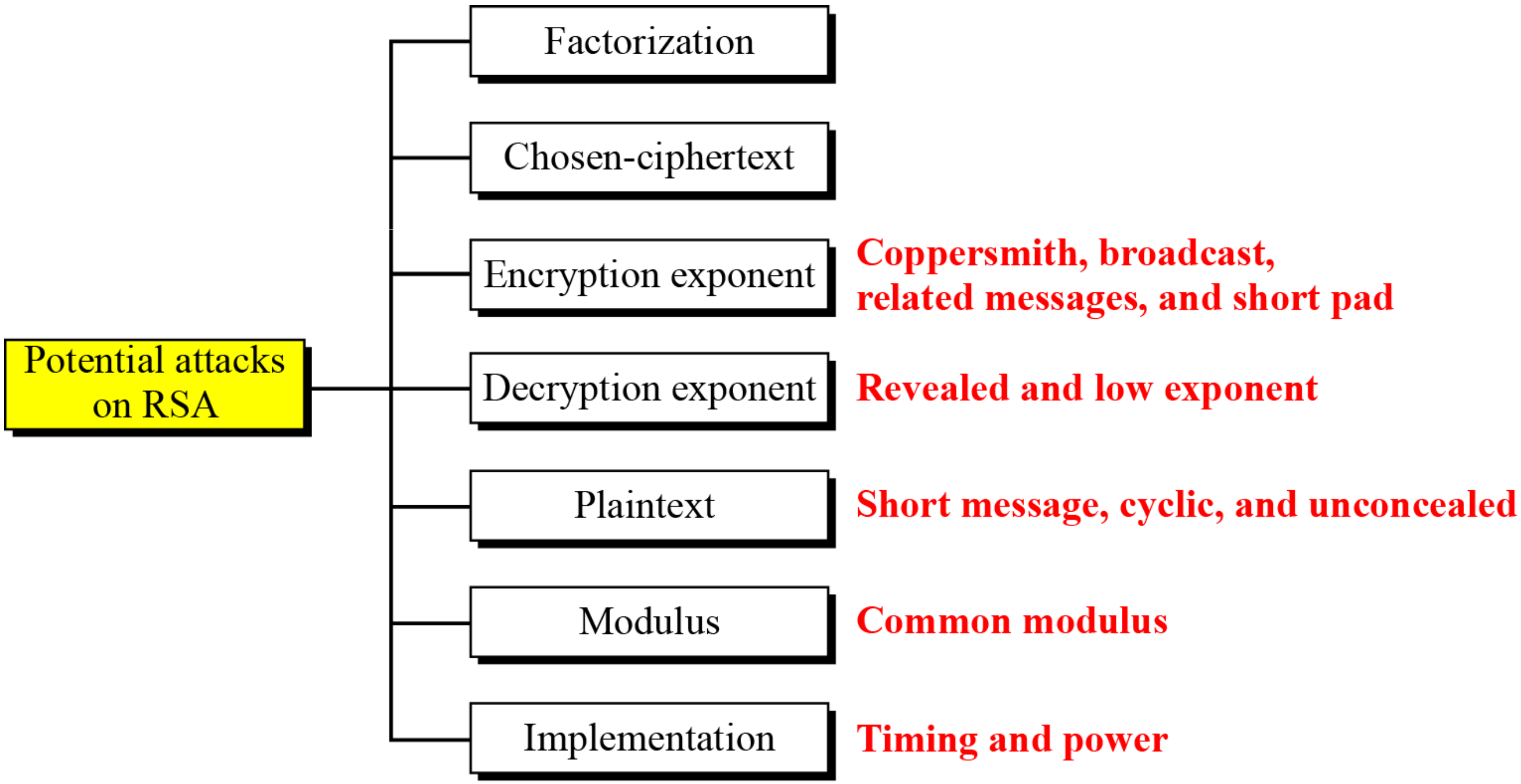
10.2.3 Continued

***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



10.2.4 Continued

Factorization Attack

- The security of RSA is based on the idea that the modulus is so large that is infeasible to factor it in reasonable time.
- Even though n is public, p & q are secret. If Eve can factor n and get p & q, she can calculate $\Phi(n)$. Then she can calculate $d=e \bmod \Phi(n)$ because e is public.

Recommendations

- The number of bits in n should be at least 1024.
- Two primes p & q must be 512 bit at least.
- p & q should not be close to each other.
- Modulus n must not be shared.
- If d is leaked, immediately change n, e and d.
- Message must be padded by OAEP.