Chapter 10

Asymmetric-Key Cryptography

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 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.

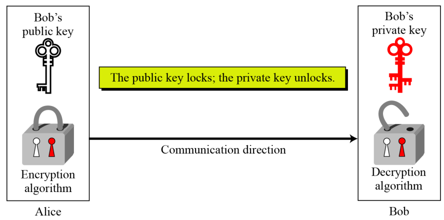
Note

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

10.1.1 Keys

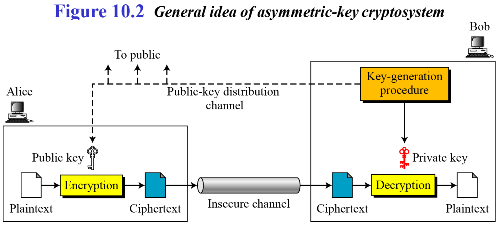
Asymmetric key cryptography uses two separate keys: one private and one public.

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.

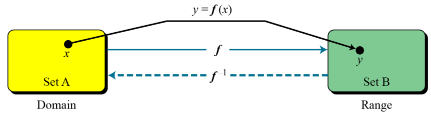
10.8

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***



10.9

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.

10.10

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.

10.11

10.1.5 Knapsack Cryptosystem

Definition

$a = [a_1, a_2, \ldots, a_k]$ and $x = [x_1, x_2, \ldots, x_k]$.

$$s = knapsackSum\ (a, x) = x_1a_1 + x_2a_2 + \cdots + x_ka_k$$

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

Superincreasing Tuple

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

10.12

10-2 RSA CRYPTOSYSTEM

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

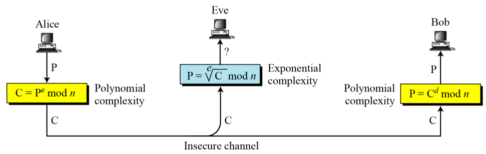
Topics discussed in this section:

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

10.17

10.2.1 Introduction

Figure 10.5 *Complexity of operations in RSA*

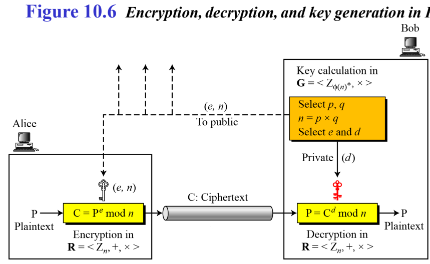


RSA uses modular exponentiation for encryption/decryption; To attack it, Eve needs to calculate $\sqrt[e]{C} \bmod n$.

10.18

10.2.2 Procedure

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



10.19

10.2.2 Continued

Two Algebraic Structures

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

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

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

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

10.20

10.2.2 Continued

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
}
```

10.21

10.2.2 Continued

Encryption

Algorithm 10.3 *RSA encryption*

```
RSA_Encryption (P, e, n)                     // P is the plaintext in Zn 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.

10.22

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
}
```

10.23

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)

10.24

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

10.25

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

10.26

10.2.3 Some Trivial Examples

Example 10. 7

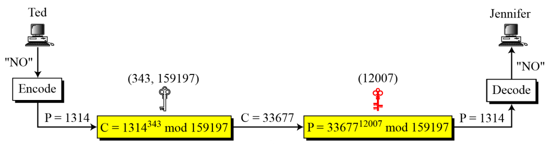
Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 159197$. She then calculates $\phi(n) = 158400$. She then chooses e = 343 and d = 12007. Show how Ted can send a message to Jennifer if he knows e and n.

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.

10.27

10.2.3 Continued

Figure 10.7 *Encryption and decryption in Example 10.7*



10.28

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 =	96130345313583504574191581280615427909309845594996215822583150879647940455056470638491257160180347503120986660649242019180878066742109606335421992666120 9
q =	120601919572314469182767942044508960015559250546370339360617983217314821484837646592153894532091752252732268301071206956046025138871455249690003596600456 17

10.31

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

10.32

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 =	580083028600377639360936612896779175946690620896509621804228661113 805938528223587317062869100300217108590443384021707298690876006115 306202524959884448047568240966247081485817130463240644077704833134 010850947385295645071936774061197326557424237217617674620776371642 076003370853332885321447088595513667029483 1

10.33

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 =	475309123646226827206365550610545180942371796070491716523239243054 452960613199328566617843418359114151197411252005682979794571736036 101278218847892741566090480023507190715277185914975188465888632101 148354103361657898467968386763733765777465625079280521148141844048 141844308127730590046928742485591664621086 56

10.34

10.2.6 Continued

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.

10.35

10-4 ELGAMAL CRYPTOSYSTEM

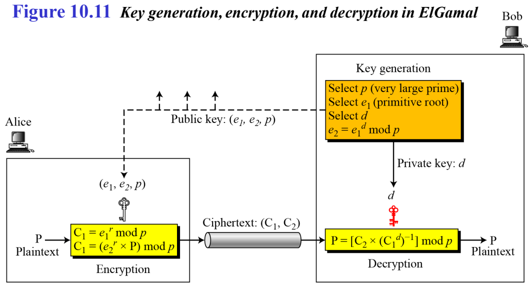
Besides RSA and Rabin, another public-key cryptosystem is ElGamal. ElGamal is based on the discrete logarithm problem discussed in Chapter 9.

Topics discussed in this section:

10.43

10.4.2 Procedure

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



10.44

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
}
```


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
}
```

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
C_1 =	887297069383528471022570471492275663120260067256562125018188351429 417223599712681114105363661705173051581533189165400973736355080295 736788569060619152881
C_2 =	708454333048929944577016012380794999567436021836192446961774506921 244696155165800779455593080345889614402408599525919579209721628879 6813505827795664302950

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

P =	3200

10.51