UNIT- III:
Asymmetric Encryption
Mathematics of Asymmetric Key Cryptography, Asymmetric Key Cryptography

Ch-9 – Mathematics of Asymmetric-Key Cryptography

9.1 Primes

Definition

The positive integers can be divided into three groups
1. The number 1
2. Primes
3. Composites

As shown in figure 9.1.

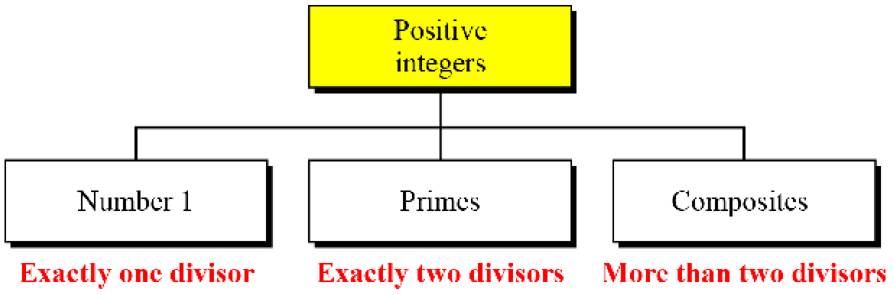


Figure 9.1 Three groups of positive integers

Example:
What is the smallest prime?

The smallest prime is 2, which is divisible by 2 (itself) and 1.

Example:
List the primes smaller than 10.

There are four primes less than 10: 2, 3, 5, and 7. It is interesting to note that the percentage of primes in the range 1 to 10 is 40%. The percentage decreases as the range increases.

Coprime or relatively prime
Two positive integers, a and b, are relatively prime or coprime, if gcd (a, b) = 1.

9.2 Cardinality of Primes

Infinite number of primes the number of primes is infinite.

Number of primes
We can use infinite Number of Primes.

Number of Primes
$\pi(n)$ is the number of primes less than or equal to n. π is not similar to mathematics π.

The primes under 25 are 2, 3, 5, 7, 11, 13, 17, 19 and 23 so π(3) = 2, π(10) = 4 and π(25) = 9.

$$\boxed{[n / (\ln n)] \quad < \quad \pi(n) \quad < \quad [n/(\ln n - 1.08366)]}$$

Guass discovered the upper limit; Lagrange discovered the lower limit.

9.3 Euler's Phi-Function

Euler's phi-function, Φ(n), which is sometimes called the ***Euler's totient function***, plays a very important role in cryptography. The function finds the number of integers that are both smaller than n and relatively prime to n.

The following helps to find Φ(n)

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

The difficulty of finding Φ(n) depends on the difficulty of finding the factorization of n.

Example:
What is the value of Φ(13)?

Because 13 is a prime, Φ(13) = (13-1) = 12

Example:
What is the value of Φ(10)?

We can use the third rule: Φ(10) = Φ(2) x Φ(5) = 1 x 4 = 4, 2 and 5 are primes.

Example:
What is the value of Φ(240)?

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:
Can we say that Φ(49) = Φ(7) x Φ(7) = 6 x 6 = 36?

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

The answer is Φ(14) = Φ(7) x Φ(2) = 6 x 1 = 6. The members are 1, 3, 5, 9, 11, and 13.

The interesting point: if n > 2, the value of Φ(n) is even.

Exercise: Find the value of Φ(29), Φ(32), Φ(80), Φ(100), Φ(101).

9.4 Fermat's little theorem

Fermat's little theorem plays a very important role in number theory and cryptography. We introduce two versions of the theorem here.

First version The first version says that if p is a prime and a is an integer such that p does not divide a, then **$a^{p-1} \equiv 1$ mod p**.

Second version The second version removes the condition on a. it says that if p is prime and a is an integer, then **$a^p \equiv a$ mod p**.

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

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

6^{11} mod 11 = 1 mod 11 = 1

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

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

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

Exercise:
Find the results of the following, using Fermat's little theorem.
a. 5^{15} mod 13 b. 15^{18} mod 17 c. 456^{17} mod 17 d. 145^{102} mod 101

Multiplicative Inverses
A very interesting application of Fermat's theorem is in finding some multiplicative inverses quickly if the modulus is a prime.

If p is a prime and a is an integer such that p does not divide a ($p \nmid a$) then **a^{-1} mod p = a^{p-2} mod p**.

Take Fermat's little theorem first version and divide with a on both sides then,

$$\frac{(a^{p-1}\, mod\, p)}{a} = \frac{1\, mod\, p}{a}$$

$$\Rightarrow \qquad a^{p-2}\, mod\, p = a^{-1}\, mod\, p$$

This application eliminates the use of extended Euclidian algorithm for finding some multiplicative inverses.

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

Example:

How to calculate multiplicative inverse of 5 modulo 23 that is 5^{-1} mod 23?

Solution:

5^{-1} mod 23 = 5^{23-2} mod 23 (Ref: a^{-1} mod p= a^{p-2} mod p) 5^{23-2} mod 23 = 5^{21} mod 23

Calculate following to solve 5^{21} mod 23:

5^1 mod 23 = 5

5^2 mod 23 = 25 mod 23=2

5^4 mod 23 = $(5^2)^2$ mod 23= $(2)^2$ mod 23 = 4

5^8 mod 23 = $(5^4)^2$ mod 23 $(4)^2$ mod 23 =16

5^{16} mod 23 = $(5^8)^2$ mod 23 $(16)^2$ mod23 = 256 mod 23=3

5^{21} mod 23 = (5^{16} x 5^4 x 5^1) mod 23 = (3x4x5) mod 23 = 60 mod 23 = 14 mod 23.

Finally 5^{-1} mod 23 = 5^{21} mod 23 = 14 mod 23

Exercise:

Find the results of the following, using Euler's theorem:

a. 5^{-1} mod 13 b. 15^{-1} mod 17 c. 27^{-1} mod 41 d. 70^{-1} mod 101

(Note that all moduli are primes)

9.5 Euler's theorem

First version if a and n are coprime, then $a^{\Phi(n)} \equiv 1 \bmod n$

Second version if n = p x q, a < n, and k an integer, then $a^{k \times \Phi(n)+1} \equiv a \bmod n$

The second version of Euler's theorem is used in the RSA cryptosystem in chapter 10.

Exponentiation Euler's theorem sometimes is helpful for quickly finding a solution to some exponentiations. The following examples show the idea.

Example:

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

Solution:

We have 6^{24} mod 35 = $6^{\Phi(35)}$ mod 35 = 1

Example:

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

Solution:

If we let k=1 on the second version, we have 20^{62} mod 77 = $\left[(20 \text{ mod } 77)\ (20^{\Phi(77)+1} \text{ mod } 77)\right]$ mod 77 = (20)(20) mod 77 = 15.

Multiplicative Inverses
Fermat's little theorem can be used to find multiplicative inverses modulo a prime. Euler's theorem can be used to find multiplicative inverses modulo a composite.

If *n* and *a* are coprime, take Euler's first version

$a^{\Phi(n)} \equiv 1$ mod n

$\Rightarrow$ $a^{\Phi(n)}$ mod n = 1 mod n

divide with a on both sides

$\Rightarrow$ ($a^{\Phi(n)}$ mod n) / a = (1 mod n) / a

$\Rightarrow$ ($a^{\Phi(n)-1}$ mod n) = (a^{-1} mod n)

Example:
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} mod 77 = $8^{\Phi(77)-1}$ mod 77 = 8^{59} mod 77 = 29 mod 77
b. 7^{-1} mod 15 = $7^{\Phi(15)-1}$ mod 15 = 7^{7} mod 15 = 13 mod 15
c. 60^{-1} mod 187 = $60^{\Phi(187)-1}$ mod 187 = 60^{159} mod 187 = 53 mod 187
d. 71^{-1} mod 100 = $71^{\Phi(100)-1}$ mod 100 = 71^{39} mod 100 = 31 mod 100

Exercise:
a. 12^{-1} mod 77 b. 16^{-1} mod 323 c. 20^{-1} mod 403 d. 44^{-1} mod 667

9.6 Factorization

Factorization plays a very important role in the security of several public-key cryptosystems.

9.6.1 Fundamental theorem of arithmetic

According to fundamental theorem of arithmetic, any positive integer greater than one can be written uniquely in the following prime factorization form where p1, p2, …, pk are primes and e1, e2, …, ek are positive integers.

$$\mathbf{n = p_1^{e1} \times p_2^{e2} \times \ldots \times p_k^{ek}}$$

Example: 24 = 2 x 2 x 2 x 3

9.6.2 Factorization Methods
In this section, we give a few simple algorithms that factor a composite number. The purpose is to make clear that the process of factorization is time consuming.

9.6.2.1 Trail Division Method
By far, the simplest and least efficient algorithm is the trail division factorization method. We simply try all the positive integers, starting with 2, to find one that divides n. we know that if n is composite, then **it will have a prime p <=** $\sqrt{n}$. Algorithm below shows the pseudocode for this method. The algorithm has tow loops, one outer and one inner. The outer loop finds unique factors; the inner

loops finds duplicates of a factor. For example, 24 = 2^3 x 3. The outer loop finds the factors 2 and 3. The inner loop finds that 2 is a multiple factor.

```
Trial_Division_Factorization (n)            // n is the number to be factored
{
    $a \leftarrow 2$
    while ($a \leq \sqrt{n}$)
    {
        while ($n \bmod a = 0$)
        {
            output $a$                      // Factors are output one by one
            $n = n / a$
        }
        $a \leftarrow a + 1$
    }
    if ($n > 1$) output $n$                 // n has no more factors
}
```

Example:

Use the trail division algorithm to find the factors of 1233.

We run a program based on the algorithm and get the following result.

1233 = 3^2 x 137

Example:

Use the trail division algorithm to find the factors of 1523357784.

We run a program based on the algorithm and get the following result.

1523357784 = 2^3 x 3^2 x 13 x 37 x 43987

9.6.2.2 Fermat method

The Fermat factorization method (below algorithm) divides a number n into two positive integers a and b (not necessarily a prime) so that n = a x b.

```
Feramat_Factorization (n)                   // n is the number to be factored
{
    $x \leftarrow \sqrt{n}$                 // smallest integer greater than $\sqrt{n}$
    while ($x < n$)
    {
    $w \leftarrow x^2 - n$
    if (w is perfect square)   $y \leftarrow \sqrt{w}$;  $a \leftarrow x + y$;  $b \leftarrow x - y$;   return $a$ and $b$
    $x \leftarrow x + 1$
    }
}
```

The Fermat method is based on the fact that if we can find x and y such that n = $x^2 - y^2$, then we have

$$N = x2 - y2 = a \times b \qquad \text{with } a = (x + y) \text{ and } b = (x - y)$$

9.6.2.3 Pollard p-1 method

The below algorithm shows the pseudocode for Pollard p-1 factorization method.

```
Pollard_ (p − 1) _Factorization (n, B)              // n is the number to be factored
{
    a ← 2
    e ← 2
    while (e ≤ B)
    {
        a ← a^e mod n
        e ← e + 1
    }
    p ← gcd (a − 1, n)
    if 1 < p < n    return p
    return failure
}
```

9.6.2.4 Pollard rho method

The below algorithm shows the pseudocode for Pollard rho factorization method.

```
Pollard_ rho _Factorization (n, B)              // n is the number to be factored
{
    x ← 2
    y ← 2
    p ← 1
    while (p = 1)
    {
        x ← f(x) mod n
        y ← f (f (y) mod n) mod n
        p ← gcd (x − y, n)
    }
    return p                                    // if p = n, the program has failed
}
```

9.7 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:

$$
\begin{aligned}
x &\equiv a_1 \pmod{m_1}\\
x &\equiv a_2 \pmod{m_2}\\
&\dots\\
&\dots\\
x &\equiv a_k \pmod{m_k}
\end{aligned}
$$

The Chinese remainder theorem states that the above equations have a unique solution if the moduli are relatively prime.

Example:

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

$$
\begin{aligned}
x &\equiv 2 \pmod{3}\\
x &\equiv 3 \pmod{5}\\
&\dots\\
&\dots\\
x &\equiv 2 \pmod{7}
\end{aligned}
$$

the solution to this set of equations is given in the next example; for the movement, 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 steps:

The solution to the set of equations follow these steps:

1. Find $M = m_1 * m_2 * \dots * m_k$. This is 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}, $^{-1}$ M_2^{-1}, … M_k^{-1}.
4. The solution to the simultaneous equation is

$$x = [\ (a_1 * M_1 * M_1^{-1}) + (a_2 * M_2 * M_2^{-1}) + \dots + (a_k * M_k * M_k^{-1})\ ] \bmod M$$

Note that the set of equations can have a solution even if the moduli are not relatively prime but meet other conditions. However, in cryptography, we are only interested in solving equations with coprime moduli.

Example:

Find the solution to the simultaneous equations:

$$
\begin{aligned}
x &\equiv 3 \pmod{7}\\
x &\equiv 3 \pmod{13}\\
&\dots\\
&\dots\\
x &\equiv 0 \pmod{12}
\end{aligned}
$$

Solution:

$M = m_1 \times m_2 \cdots \times m_k$

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

$M_1 = \frac{M}{m_1} = \frac{105}{3} = 35$

$M_2 = \frac{M}{m_2} = \frac{105}{5} = 21$

$M_3 = \frac{M}{m_3} = \frac{105}{7} = 15$

Finding inverses of M1, M2, M3

$(M_1 \times M_1^{-1}) \bmod 3 = 1$

$(35 \times 2) \bmod 3 = 1$

$70 \bmod 3 = 1$

$\Rightarrow \boxed{M_1^{-1} = 2}$

M_2^{-1}

$(M_2 \times M_2^{-1}) \bmod 5 = 1$

$(21 \times 1) \bmod 5 = 1$

$\Rightarrow \boxed{M_2^{-1} = 1}$

M_3^{-1}

$(M_3 \times M_3^{-1}) \bmod 7 = 1$

$(15 \times 1) \bmod 7 = 1$

$\Rightarrow \boxed{M_3^{-1} = 1}$

The solution is

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

$= \left[(2 \times 35 \times 2) + (3 \times 21 \times 1) + (2 \times 15 \times 1)\right] \bmod 105$

$= (140 + 63 + 30) \bmod 105$

$= 233 \bmod 105$

$= 23 \bmod 105$

$= \underline{\underline{23}}$

Example:

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 then to find the value of x.

```
x ≡ 2 (mod 3)
x ≡ 3 (mod 5)
      …
      …
x ≡ 2 (mod 7)
```

$M = m_1 \times m_2 \times \cdots \times m_k$

$M = 7 \times 13 \times 12$

$= 1092$

$M_1 = \frac{M}{m_1} = \frac{1092}{7} = 156$

$M_2 = \frac{M}{m_2} = \frac{1092}{13} = 84$

$M_3 = \frac{M}{m_3} = \frac{1092}{12} = 91$

$\underline{M_1^{-1}}$

$(M_1 \times M_1^{-1}) \bmod 7 = 1$

$(156 \times M_1^{-1}) \bmod 7 = 1$

$(156 \times 4) \bmod 7 = 1$

$624 \bmod 7 = 1$

$\Rightarrow \boxed{M_1^{-1} = 4}$

$\underline{M_2^{-1}}$

$(M_2 \times M_2^{-1}) \bmod 13 = 1$

$(84 \times M_2^{-1}) \bmod 13 = 1$

$(84 \times 11) \bmod 13 = 1$

$924 \bmod 13 = 1$

$\Rightarrow \boxed{M_2^{-1} = 11}$

$\underline{M_3^{-1}}$

$(M_3 \times M_3^{-1}) \bmod 12 = 1$

$(91 \times M_3^{-1}) \bmod 12 = 1$

$(91 \times 7) \bmod 12 = 1$

$637 \bmod 12 = 1$

$\Rightarrow \boxed{M_3^{-1} = 7}$

The Solution is

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

$= [(3 \times 156 \times 4) + (3 \times 84 \times 11) + (0 \times 91 \times 7)] \bmod 1092$

$= [1872 + 2772 + 0] \bmod 1092$

$= 4644 \bmod 1092$

$= 276$

Applications:

1. The Chinese remainder theorem has several applications in cryptography. One is to solve quadratic congruence as discussed in the next section.
2. The other is to represent a very large integer in terms of a list of small integers.

Exercise:

Find the value of x for the following sets of congruence using the Chinese remainder theorem.

a. x ≡ 2 mod 7, and x ≡ 3 mod 9
b. x ≡ 4 mod 5, and x ≡ 10 mod 11
c. x ≡ 7 mod 13, and x ≡ 11 mod 12

9.8 Primitive roots of a group

Definition

In the group G = <Z_n^*, x>, when the order of an element is the same as Φ(n), the element is called the primitive root of the group.

Example:
Find primitive roots of the group $G = \langle Z_5^*, \times \rangle$

In this group, $\Phi(5) = 4$

The following table shows all the powers of modulo 5 for all positive a<5. The length of the sequence for each base value is indicated by shading.

a	a mod 5	a^2 mod 5	a^3 mod 5	a^4 mod 5
1	1	1	1	1
2	2	4	3	1
3	3	4	2	1
4	4	1	4	1

The above table shows that only two elements, 3 and 5, have the order $\Phi(5) = 4$. Therefore, this group has only two primitive roots: 3 and 5.

Note:
It has been proved that the group $G = \langle Z_n^*, \times \rangle$ **has primitive roots only if n is 2, 4, p^t, or $2p^t$**, in which p is an odd prime (not 2) and t is an integer.

Example:
Find primitive roots of the group $G = \langle Z_7^*, \times \rangle$

In this group, $\Phi(7) = 6$

a	a mod 7	a^2 mod 7	a^3 mod 7	a^4 mod 7	a^5 mod 7	a^6 mod 7
1	1	1	1	1	1	1
2	2	4	1	2	4	1
3	3	2	6	4	5	1
4	4	2	1	4	2	1
5	5	4	6	2	3	1
6	6	1	6	1	6	1

From the above table primitive roots are: 3 and 5 because they have the order $\Phi(7) = 6$.

Example:
For which value of n, does the group $G = \langle Z_n^*, \times \rangle$ have primitive roots: 17, 20, 38, and 50?

a. $G = \langle Z_{17}^*, \times \rangle$ has primitive roots, because 17 is a prime (p^t where t is 1)
b. $G = \langle Z_{20}^*, \times \rangle$ has no primitive roots.
c. $G = \langle Z_{38}^*, \times \rangle$ has primitive roots, because 38 = 2 x 19 and 19 is a prime.
d. $G = \langle Z_{50}^*, \times \rangle$ has primitive roots, because $50 = 2 \times 5^2$ and 5 is a prime.

Note:
If a group has a primitive root, then it normally has several of them. The number of **primitive roots can be calculated as $\Phi(\Phi(n))$**. For example, the number of primitive roots of $G = \langle Z_{17}^*, \times \rangle$ is $\Phi(\Phi(17)) = \Phi(16) = 8$. Note that we should first check to see if the group has any primitive root, before we find the number of roots.

9.9 Discrete logarithms

Discrete logarithms are fundamental to a number of public-key algorithms, including Diffie–Hellman key exchange and the digital signature algorithm (DSA). This section provides a brief overview of discrete logarithms.

The Powers of an integer, Modulo n

Recall from Euler's theorem that, for every a and n that are relatively prime,

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

where Φ(n), Euler's totient function, is the number of positive integers less than n and relatively prime to n. Now consider the more general expression:

$$a^{m} \equiv 1 \pmod{n} \qquad (1)$$

If a and n are relatively prime, then there is at least one integer m that satisfies Equation (1), namely, m = Φ(n). The least positive exponent m for which Equation (1) holds is referred to in several ways:

- The order of a (mod n)
- The exponent to which a belongs (mod n)
- The length of the period generated by a

To see this last point, consider the powers of 7, modulo 19:

$$7^1 \equiv 7 \bmod (19)$$
$$7^2 = 49 \equiv 11 \bmod (19)$$
$$7^3 = 343 \equiv 1 \bmod (19)$$
$$7^4 = 2401 \equiv 7 \bmod (19)$$
$$7^5 = 16807 \equiv 11 \bmod (19)$$

There is no point in continuing because the sequence is repeating. This can be proven by noting that the sequence is periodic, and the length of the period is the smallest positive exponent m such that $7^m \equiv 1 \bmod 19$.

Table below shows all the powers of a, modulo 19 for all positive a < 19. The length of the sequence for each base value is indicated by shading. Note the following point:

Some of the sequences are of length 18. In this case, it is said that the base integer a generates (via powers) the set of nonzero integers modulo 19. Each such integer is called a **primitive root** of the modulus 19.

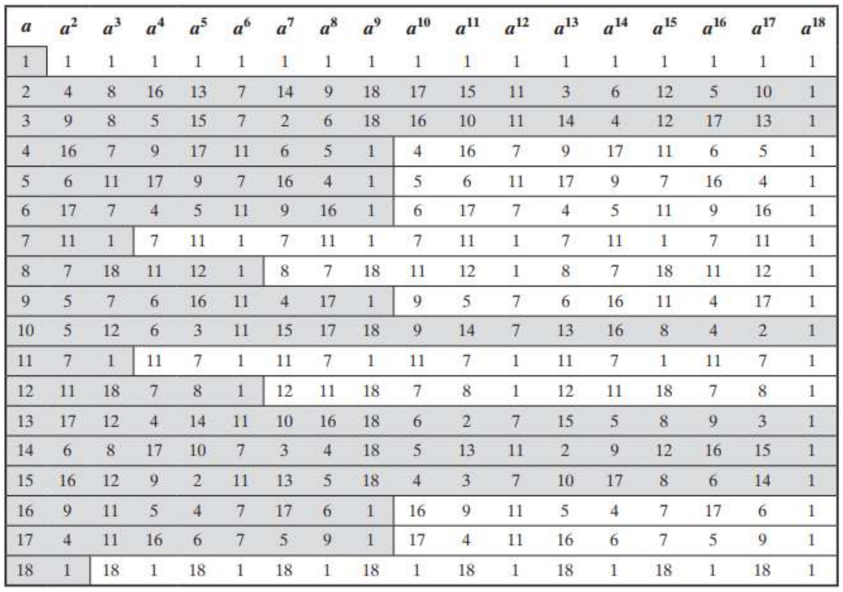
More generally, we can say that the highest possible exponent to which a number can belong (mod n) is Φ(n). If a number is of this order, it is referred to as a primitive root of n. The importance of this notion is that if a is a primitive root of n, then its powers

$$a, a^2, \ldots a^{\Phi(n)}$$

are distinct (mod n) and are all relatively prime to n. In particular, for a prime number p, if a is a primitive root of p, then

$$a, a^2, \ldots a^{p-1}$$

are distinct (mod p). For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15.



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

Table: Powers of Integers, Modulo 19

Logarithms for Modular Arithmetic
With ordinary positive real numbers, the logarithm function is the inverse of exponentiation. An analogous function exists for modular arithmetic

Let us briefly review the properties of ordinary logarithms. The logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal the number. That is, for base x and for a value y,

$$y = x^{\log_x(y)}$$

The properties of logarithms include

$$\log_x(1) = 0$$
$$\log_x(x) = 1$$

$$\log_x(yz) = \log_x(y) + \log_x(z)$$

$$\log_x(y^r) = r \times \log_x(y)$$

Consider a primitive root a for some prime number p (the argument can be developed for nonprimes as well). Then we know that the powers of a from 1 through (p - 1) produce each integer from 1 through (p - 1) exactly once. We also know that any integer b satisfies

Discrete Logarithms
It follows that for any integer ‘b’ and a primitive root ‘a’ of prime number ‘p’, we can find a unique exponent ‘i’, such that

$b \equiv a^i \bmod p$ where $0 <= i <= (p-1)$

this exponent i is referred to as the **discrete logarithm** of the number ‘b’ for the base a mod p. we denote this value as $\text{dlog}_{a,p}(b)$.

Ch-10 – Asymmetric-Key Cryptography

1. Introduction to asymmetric-key cryptography

Public-key algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic:

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms, such as RSA, also exhibit the following characteristic:

- Either of the two related keys can be used for encryption, with the other used for decryption.

A public-key encryption scheme has six ingredients

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and Private Key:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different cipher texts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

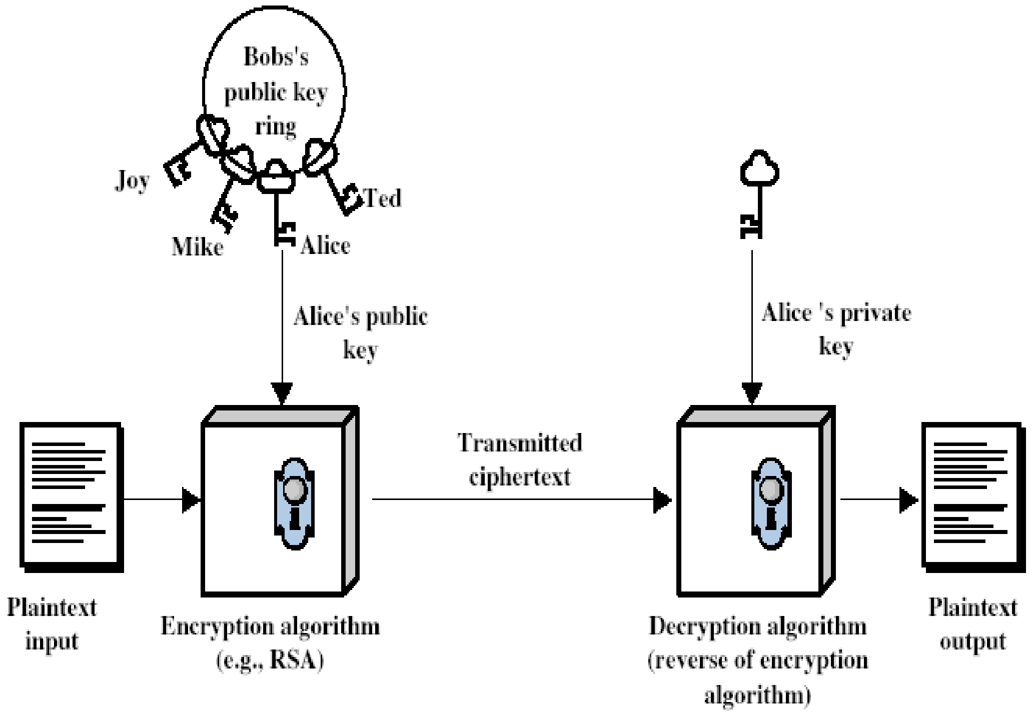


Figure 10.1: Public Key Cryptography (Secrecy)

The essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 10.1 suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as system controls its private key, its incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

Let us take a closer look at the essential elements of a public-key encryption scheme, using Figure 10.2. There is some source A that produces a message in plaintext X. The message is intended for destination B. B generates a related pair of keys: a public key, KU_b, and a private key, KR_b. KR_b is known only to B, whereas KU_b is publicly available and therefore accessible by A.

With message X and the encryption key KU_b as input, A forms the ciphertext Y.

$$Y = E_{KUb}(X)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = D_{KRb}(Y)$$

An opponent, observing Y having access to KU_b and does have knowledge of encryption (E) and decryption (D) algorithms. But he does not have access to KR_b. so he can't generate the original message.

So, in this case secrecy is there but authentication fails. Anybody can send message using B's public key but receiver confuses that which one is the original A's message.

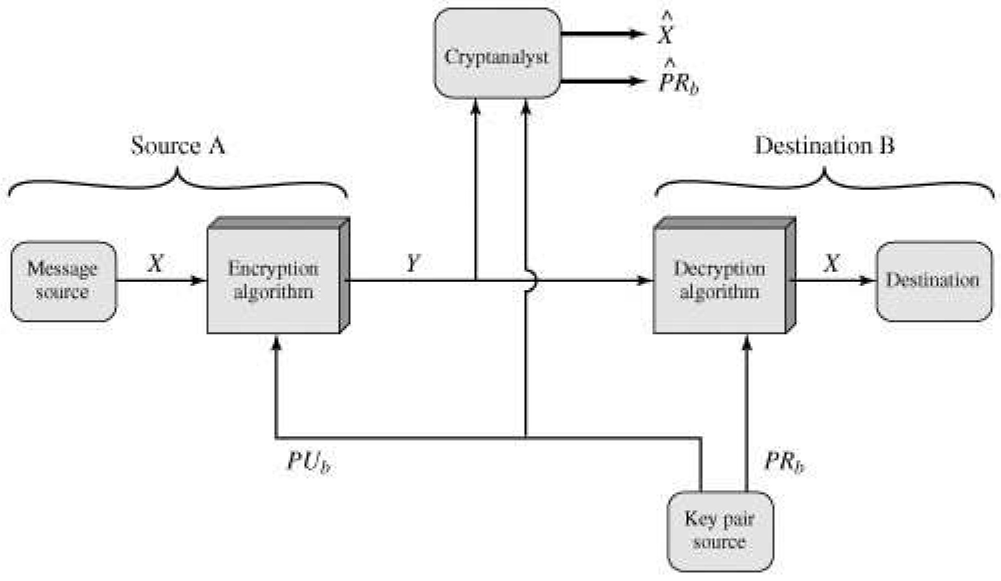


Figure 10.2: Public Key Cryptosystem: Secrecy

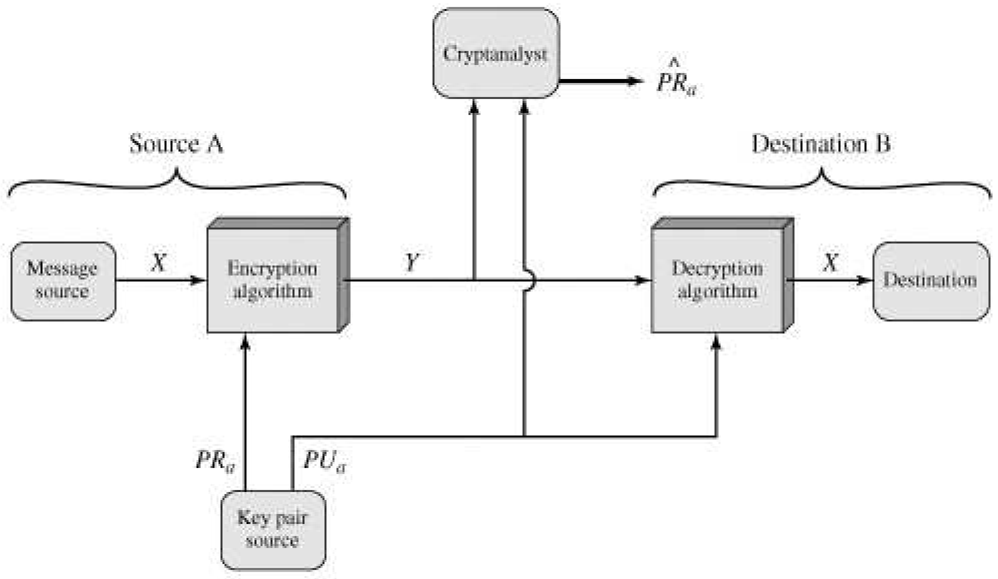


Figure 10.3: Public Key Cryptosystem: Authentication

Another scheme illustrated in Figure 10.3 show the use of public key encryption to provide authentication.

$$Y = E_{KRa}(X)$$
$$X = D_{KUa}(Y)$$

In this case, A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message.

It is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

In this scheme authentication is there but no secrecy. Anybody can decrypt the message and obtain the plaintext using A's public key.

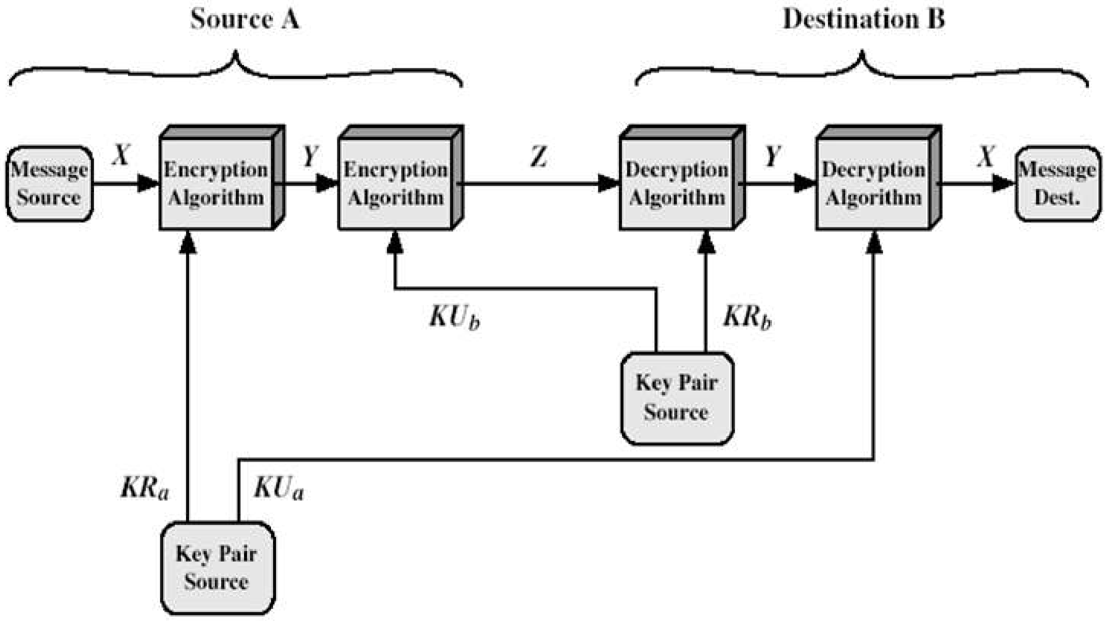


Figure 10.4: Public key cryptosystem: Secrecy and Authentication

It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme: (Figure 10.4)

$$Z = E_{KUb}[E_{KRa}(X)]$$
$$X = D_{KUa}[D_{KRb}(Z)]$$

In this case, we begin as before by encryption a message, using the sender's private key. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided. The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

Difference between symmetric-key and asymmetric-key cryptosystems

- The conceptual difference between the two systems are based on ***how these systems keep a secret***. In symmetric-key cryptography, the secret must be shared between two persons. In asymmetric-key cryptography, the secret is personal (unshared); each person creates and keeps his or her own secret.
- In a community of n people, ***n(n-1)/2 shared secrets*** are needed for symmetric-key cryptography; ***only n personal secrets*** are needed in asymmetric-key cryptography. For a community with a population of 1 million, symmetric-key cryptography would require half a billion shared secrets; asymmetric-key cryptography would require 1 million personal secrets.
- Symmetric-key cryptography is based on ***substitution and permutation of symbols*** (characters or bits), asymmetric-key cryptography is based on applying ***mathematical functions to numbers.***
- In symmetric key cryptography, the plaintext and ciphertext are thought of as ***a combination of symbols.*** Encryption and decryption permute these symbols or substitute a symbol for another. In asymmetric-key cryptography, the ***plaintext and ciphertext are numbers***; encryption and decryption are mathematical functions that are applied to numbers to create other numbers.

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

In symmetric-key cryptography, symbols are permuted or substituted;
In asymmetric-key cryptography, numbers are manipulated.

Need for both

There is a very important fact that is sometimes misunderstood: The advent of asymmetric-key (public-key) cryptography does not eliminate the need for symmetric-key cryptography. The reason is that asymmetric-key cryptography, which uses mathematical functions for encryption and decryption, is much slower than symmetric-key cryptography. For encipherment of large messages, symmetric-key cryptography is still needed. On the other hand, the speed of symmetric-key cryptography does not eliminate the need for asymmetric-key cryptography. Asymmetric-key cryptography is still needed for authentication, digital signatures, and secret-key exchanges. This means that, to be able to use all aspects of security today, we need both symmetric-key and asymmetric-key cryptography.

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

A function is a rule that associates (maps) one element in set A, called the domain, to one element in set B, called the range, as shown in the below figure 10.5.

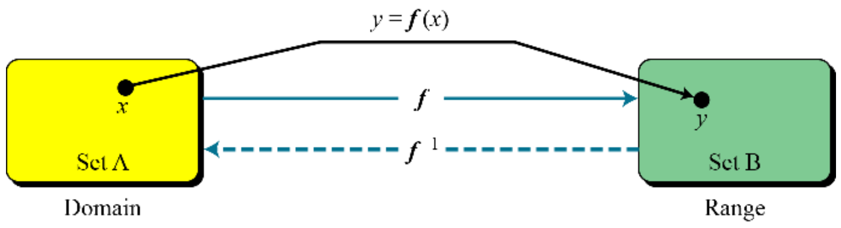


Figure 10.5: A function as rule mapping a domain to a range

One-way function A one-way function (OWF) is a function that satisfies the following two properties:

1. f is easy to compute. In other words, given x, y = f(x) can be easily computed.
2. f^{-1} is difficult to compute. In other words, given y, it is computationally infeasible to calculate $x = f^{-1}(y)$

A **trapdoor one-way function (TOWF)** is a one-way function with a third property:

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

Knapsack Cryptosystem

Suppose we are given two k-tuples, $a = [a_1, a_2, \ldots, a_k]$ and $x = [x_1, x_2, \ldots, x_k]$.

The first tuple is the predefined set; the second tuple, in which x_i is only 0 or 1, defines which elements of *a* are to be dropped in the knapsack. The sum of elements in the knapsack is

$$s = \text{knapsackSum}(a, x) = a_1x_1 + a_2x_2 + \ldots\ldots + a_kx_k$$

Given a and x, it is easy to calculate s. however, given *s* and *a* it is difficult to find x. in other words,

s = knapsackSum(x, a) is easy to calculate, but

x = inv_knapsackSum(s, a) is difficult.

Superincreasing Tuple It is easy to compute knapsackSum and inv_knapsackSum if the k-tuple a is superincreasing.

In a superincreasing tuple, $a_i \geq a_1 + a_2 + \ldots + a_{i-1}$.

In other words, each element (except a_1) is greater than or equal to the sum of all previous elements.

```
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]
}
```

Algorithm: knapsackSum and inv_knapsackSum for a superincreasing k-tuple

Example:
Assume that a = [17, 25, 46, 94, 201,400] and s = 272 are given. Show that the tuple x is found using inv_knapsackSum routine.

Solution:

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

In this case x = [0, 1, 1, 0, 1, 0], which means that 25, 46, and 201 are in the knapsack.

Secret Communication with Knapsacks
Let us see how Alice can send a secret message to Bob using a knapsack cryptosystem. The idea is shown in Figure 10.6.

Key Generation

a. Create a superincreasing k-tuple b = [b_1, b_2, …, b_k].
b. Choose a modulus n, such that n > b_1 + b_2 + … + b_k.
c. Select a random integer r that is relatively prime with n and 1 <= r <= n-1.
d. Create temporary k-tuple t = [t_1, t_2, … t_k] in which t_i = r x b_i nod 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.

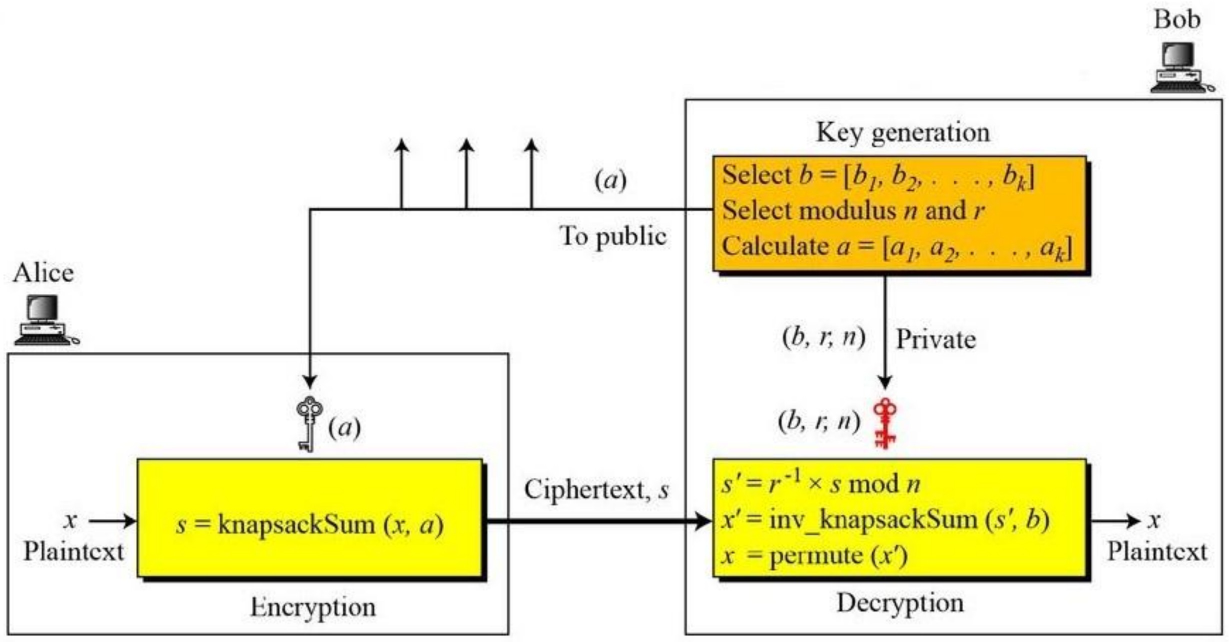

Bob
Key generation
Select $b = [b_1, b_2, \ldots, b_k]$
Select modulus n and r
Calculate $a = [a_1, a_2, \ldots, a_k]$
(a)
To public
Alice
(b, r, n)
Private
(a)
(b, r, n)
x
Plaintext
s = knapsackSum (x, a)
Ciphertext, s
$s' = r^{-1} \times s \bmod n$
x' = inv_knapsackSum (s', b)
x = permute (x')
x
Plaintext
Encryption
Decryption

Figure 10.6: Secret Communication with Knapsack cryptosystem

Encryption
Suppose Alice needs to send a message to Bob.

a. Alice converts her message to a k-tuple x = [x_1, x_2, …, x_k] in which xi 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} x s mod n.
b. Bob uses inv_knapsackSum to create x'.
c. Bob permutes x' to find x. the tuple x is the recovered plaintext.

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".

2. RSA Cryptosystem

The pioneering paper by Diffie and Hellman introduced a new approach to cryptography and, in effect, challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems. One of the first of the responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. The **Rivest-Shamir-Adleman** (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between *0* and *n-1* for some *n*. A typical size for *n* is 1024 bits, or 309 decimal digits.

In RSA, p and q must be at least 512 bits; n must be at least 1024 bits.

Assume:
C- Ciphertext, M-Plaintext, e-encryption key, d-decryption key.

Both sender and receiver must know the value of *n*. The sender knows the value of *e*, and only the receiver knows the value of *d*. thus, this is a public-key encryption algorithm with a public key of KU = {*e, n*} and a private key of KR = {*d, n*}.

1. Take two large Prim numbers p & q.
2. Calculate n = p * q.
3. Calculate $\Phi(n) = (p-1)(q-1)$ from Euler's totient function.
4. Select e from $\Phi(n)$. [therefore $\gcd(\Phi(n), e) = 1;\ 1<e< \Phi(n)$]
5. Take plaintext M.
6. Calculate ciphertext $C = M^e \bmod n$.
7. Calculate plaintext $M = C^d \bmod n$, where $d = e^{-1} \bmod \Phi(n)$.

$e = 7;\ d = 3;\ \Phi(n) = 20$ $ed = \Phi(n) + 1$ $7*3 = 1+20$
$ed \equiv 1 \bmod \Phi(n)$ $7*3 \equiv 1 \bmod 20$ $21 \equiv 1 \bmod 20$

To find d value:

$$C = M^e \bmod n$$
$$M = C^d \bmod n.$$

Substitute C value then,

$$(M^e \bmod n)^d \bmod n$$
$$\equiv [(M^e)^d \bmod n] \bmod n$$
$$M \equiv M^{ed} \bmod n$$

(or) $M^{ed} \equiv M \bmod n$ …(1)

From Euler's theorem

$M^{\Phi(n)+1} \equiv M \bmod n$ … (2)

// M value is below 255 and n, p, q values are very large so m is relatively prime with n.
// M and n are relatively prime then from Euler's theorem $M^{\Phi(n)+1} \equiv M \bmod n$

From (1) and (2) we can write

$M^{ed} = M^{\Phi(n)+1}$

➡ $ed = \Phi(n) + 1$

➡ $ed \equiv 1 \bmod \Phi(n)$

➡ $d = e^{-1} \bmod \Phi(n)$

Example 1: Perform encryption and decryption using the RSA algorithm, for the following

a) p=17; q=11; e=7; M=88;

M = 88

1. p = 17, q = 11
2. n = 17 * 11 = 187
3. $\Phi(n) = 16 * 10 = 160$
4. Select e = 7
5. $C = M^e \bmod n = (88)^7 \bmod 187$

From the third property of modular arithmetic

$[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$

$\Longrightarrow [(a \bmod n) * (a \bmod n)] \bmod n = (a * a) \bmod n$

$\Longrightarrow [(a \bmod n) * (a \bmod n)] \bmod n = a^2 \bmod n$

from the above formula

$(88)^7 \bmod 187 = [(88^4 \bmod 187) * (88^2 \bmod 187) * (88^1 \bmod 187)] \bmod 187$

$88^1 \bmod 187 = 88$

$88^2 \bmod 187 = 88 \bmod 187 * 88 \bmod 187$

$= 77$

$88^4 \bmod 187 = 88^2 \bmod 187 * 88^2 \bmod 187$

$= 77 * 77 \bmod 187$

$= 132$

$\mathbf{88^7 \bmod 187} = (132 * 77 * 88) \bmod 187$

$= 11.$

1. Calculating d:
   The formula is $ed \equiv 1 \bmod \Phi(n)$

   $7 * d \equiv 1 \bmod 160$

   $7 * 23 \equiv 1 \bmod 160$ [Since 7*23=161]

   $\Longrightarrow d = 23$
2. for decryption, we calculate $M = C^d \bmod n$

$\Longrightarrow M = 11^{23} \bmod 187$

$= [(11^1 \bmod 187) * (11^2 \bmod 187) * (11^4 \bmod 187) * (11^8 \bmod 187) * (11^8 \bmod 187)] \bmod 187$

$11^1 \bmod 187 = 11$

$11^2 \bmod 187 = 121$

$11^4 \bmod 187 = 55$

$11^8 \bmod 187 = 33$

$11^{23} \bmod 187 = (11 * 121 * 55 * 33 * 33) \bmod 187$

$= 88.$

⟹ Public key KU ={7, 187}
⟹ Private key KR = {23, 187}

Example 2: Perform encryption and decryption using the RSA algorithm, for the following

a)	**p=3;**	**q=11;**	**e=7;**	**M=5.**
b)	**p=5;**	**q=11;**	**e=3;**	**M=9.**
c)	**p=11;**	**q=13;**	**e=11;**	**M=7.**

Solution:

a) p=3; q=11; e=7; M=5.

$n = p*q = 3 * 11 = 33$

$\Phi(n) = (p-1)\,(q-1) = (3-1)\,(11-1) = 2*10 = 20$

$ed \equiv 1 \bmod \Phi(n)$

$7*d \equiv 1 \bmod 20$ ⇨ $(7 * d) \bmod 20 = 1 \bmod 20$

⇨ $(7 * 3) \bmod 20 = 1 \bmod 20$

⇨ $21 \bmod 20 = 1 \bmod 20$

⇨ $d = 3$

For Encryption:

$C = M^e \bmod n$

$= 5^7 \bmod 33$

$= 78125 \bmod 33$

$= 14$

For Decryption

$M = C^d \bmod n$

$= 14^3 \bmod 33$

$= 2744 \bmod 33$

$= 5$

b) p=5; q=11; e=3; M=9.

$n = p*q = 5 * 11 = 55$

$\Phi(n) = (p-1)*(q-1) = (5-1) * (11-1) = 4*10 = 40$

$ed \equiv 1 \bmod \Phi(n)$

$3*d \equiv 1 \bmod 40$ ⇨ $(3 * d) \bmod 40 = 1 \bmod 40$

⇨ $(3 * 27) \bmod 40 = 1 \bmod 40$

⇨ $81 \bmod 40 = 1 \bmod 40$

⇨ $d = 27$

For Encryption:

$C = M^e \bmod n$

$= 9^3 \bmod 55$

= 729 mod 55

= 14

For Decryption

$M = C^d \bmod n$

$= 14^{27} \bmod 55$

$= 5$

c) p=11; q=13; e=11; M=7.

$n = p*q = 11 * 13 = 143$

$\Phi(n) = (p-1)*(q-1) = (11-1) * (13-1) = 10*12 = 120$

$ed \equiv 1 \bmod \Phi(n)$

$11*d \equiv 1 \bmod 120 \Rightarrow (11 * d) \bmod 120 = 1 \bmod 120$

$\Rightarrow (11 * 11) \bmod 120 = 1 \bmod 120$

$\Rightarrow 121 \bmod 120 = 1 \bmod 120$

$\Rightarrow d = 11$

For Encryption:

$C = M^e \bmod n$

$= 7^{11} \bmod 143$

$7^2 \bmod 143 = 49 \bmod 143$

$7^4 \bmod 143 = (49*49) \bmod 143$

$= 113$

$7^8 \bmod 143 = (113*113) \bmod 143$

$= 12769 \bmod 143$

$= 42$

$7^3 \bmod 143 = (7^2*7) \bmod 143$

$= 343 \bmod 143$

$= 57$

$7^{11} \bmod 143 = (42*57) \bmod 143$

$= 2394 \bmod 143$

$= 106$

For Decryption

$M = C^d \bmod n$

$= 106^{11} \bmod 143$

$= 7$

Exercise: Perform encryption and decryption using the RSA algorithm, for the following

a) p=7; q=11; e=17; M=8.

b) p=17; q=31; e=7; M=2.

Example:

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* × *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 $\quad$ C = 5^{13} mod 77 = 26 mod 77 $\quad$ Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26 $\quad$ P = 26^{37} mod 77 = 5 mod 77 $\quad$ Plaintext: 5

Example:
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 $\quad$ C = 63^{13} mod 77 = 28 mod 77 $\quad$ Ciphertext: 28

Bob receives the ciphertext 28 and uses his private key 37 to decipher the ciphertext:

Ciphertext: 28 $\quad$ P = 28^{37} mod 77 = 63 mod 77 $\quad$ Plaintext: 63

Example:
Jennifer creates a pair of keys for herself. She chooses *p* = 397 and *q* = 401. She calculates *n* = 159197. She then calculates Φ(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.

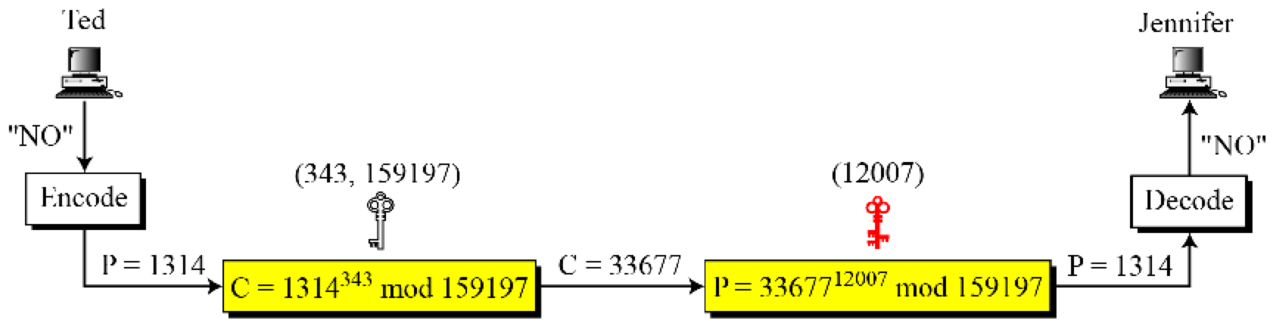


Figure 10.7 Encryption and decryption for the above example

Attacks on RSA

Security of RSA:-
These are explained as following below.

1. Plain text attacks:
It is classified into 3 subcategories:-

- **(i) Short message attack:**
  In this we assume that attacker knows some blocks of plain text and tries to decode cipher text with the help of that. So, to prevent this pad the plain text before encrypting.
- **(ii) Cycling attack:**
  In this attacker will think that plain text is converted into cipher text using permutation and

he will apply right for conversion. But attacker does not right plain text. Hence will keep doing it.

- **(iii) Unconcealed Message attack:**
  Sometimes happened that plain text is same as cipher text after encryption. So it must be checked it cannot be attacked.

2. Choosen cipher attack:
In this attacker is able to find out plain text based on cipher text using Extended Euclidian Algorithm.

3. Factorization attack:
If attacker will able to know P and Q using N, then he could find out value of private key. This can be failed when N contains at least 300 longer digits in decimal terms, attacker will not able to find. Hence it fails.

4. Attacks on Encryption key:
If we take smaller value of E in RSA this may occur so to avoid this take value of E = 2^16+1 (at least).

5. Attacks on Decryption key:

- **(i) Revealed decryption exponent attack:**
  If attacker somehow guess decryption key D, not only the cipher text generated by encryption the plain text with corresponding encryption key is in danger, but even future messages are also in danger. So, it is advised to take fresh values of two prime numbers (i.e; P and Q), N and E.
- **(ii) Low decryption exponent attack:**
  If we take smaller value of D in RSA this may occur so to avoid this take value of D = 2^16+1(at least).

3. RABIN Cryptosystem

The Rabin cryptosystem, devised by M. Rabin, is a variation of the RSA cryptosystem.

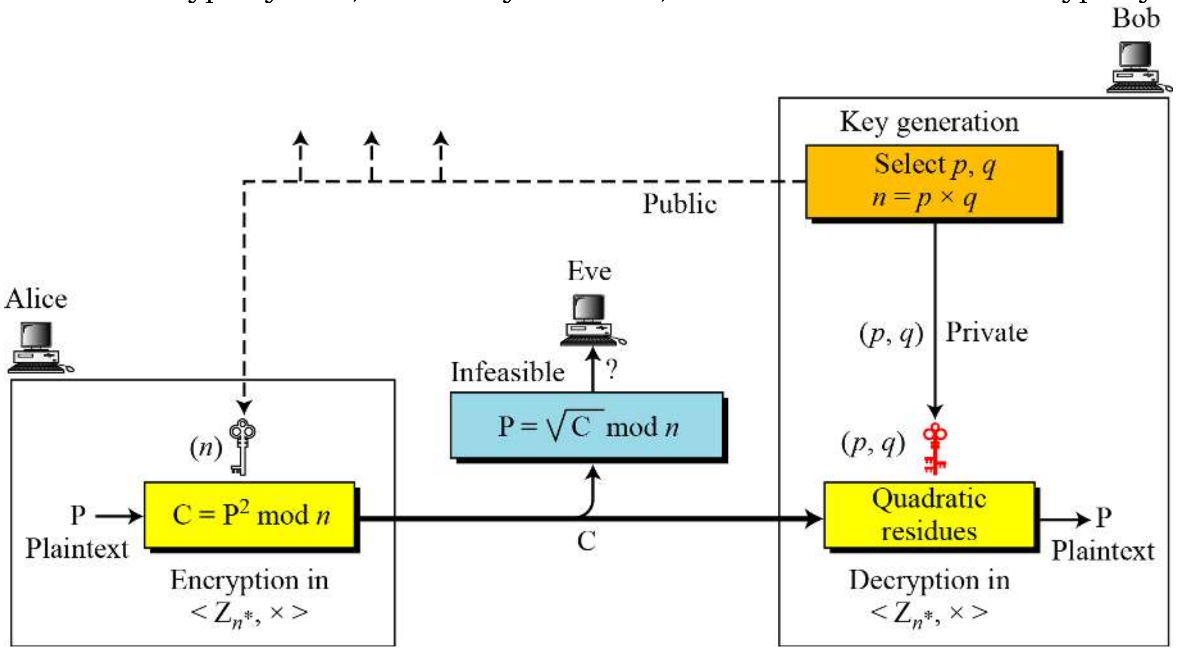


Figure 10.8 Rabin cryptosystem

The Rabin cryptosystem can be thought of as an RSA cryptosystem in which the value of e and d are fixed; **$e = 2$ and $d = ½$.** The encryption is $C \equiv P^2 \pmod{n}$ and the decryption is $P \equiv C^{1/2} \pmod{n}$.

The public key in the Rabin cryptosystem is n; the private key is the tuple (p, q). Everyone can encrypt a message using n; only Bob can decrypt the message using p and q. decryption of the message is infeasible for Eve because she does not know the values of p and q. Figure 10.8 shows the encryption and decryption.

```
Rabin_Key_Generation
{
    Choose two large primes $p$ and $q$ in the form $4k + 3$ and $p \neq q$.
    $n \leftarrow p \times q$
    Public_key $\leftarrow$ $n$                          // To be announced publicly
    Private_key $\leftarrow (q, n)$                      // To be kept secret
    return Public_key and Private_key
}
```

Algorithm: Key generation for 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
}
```

Algorithm: Encryption in Rabin cryptosystem

```
Rabin_Decryption ($p$, $q$, C)                 // C is the ciphertext; $p$ and $q$ are private keys
{
    $a_1 \leftarrow +(C^{(p+1)/4}) \bmod p$
    $a_2 \leftarrow -(C^{(p+1)/4}) \bmod p$
    $b_1 \leftarrow +(C^{(q+1)/4}) \bmod q$
    $b_2 \leftarrow -(C^{(q+1)/4}) \bmod q$
    // The algorithm for the Chinese remainder algorithm is called four times.
    $P_1 \leftarrow$ Chinese_Remainder $(a_1, b_1, p, q)$
    $P_2 \leftarrow$ Chinese_Remainder $(a_1, b_2, p, q)$
    $P_3 \leftarrow$ Chinese_Remainder $(a_2, b_1, p, q)$
    $P_4 \leftarrow$ Chinese_Remainder $(a_2, b_2, p, q)$
    return $P_1$, $P_2$, $P_3$, and $P_4$
}
```

Algorithm: Decryption in Rabin cryptosystem

The most important point about the Rabin system is that it is to deterministic. The decryption has four answers. The receiver chooses one among the four plaintexts that is meaningful/suitable for the situation.

The Robin cryptosystem is not deterministic; Decryption creates four equally probable plaintexts.

Example:

Using the Robin cryptosystem with p = 7, q = 11:

a. Encrypt P = 10 to find the ciphertext.
b. Use the Chinese Remainder theorem to find four possible plaintexts

Solution:

p = 7, q = 11 Note that both are congruent to 3 mod 4.

n = p * q = 7 * 11 = 77

Note that 77 and 10 are relatively prime.

Encryption:

$C = p^2 \bmod n$

$= 10^2 \bmod 77$

$= 100 \bmod 77$

$= 23$

Decryption:

$a1 = + (C^{(p+1)/4)} \bmod p$

$a2 = - (C^{(p+1)/4)} \bmod p$

$b1 = + (C^{(q+1)/4)} \bmod q$

$b2 = - (C^{(q+1)/4)} \bmod q$

$a1 = + 23^{(7+1)/4} \bmod 7$

$= 23^2 \bmod 7$

$= 529 \bmod 7$

$= 4 \bmod 7$

$= 4$

$a2 = -4 \bmod 7$

$= (-4+7) \bmod 7$

$= 3 \bmod 7$

$= 3$

$b1 = + 23^{(11+1)/4} \bmod 11$

$= 23^3 \bmod 11$

$= 12167 \bmod 11$

= 1 mod 11

= 1

b2 = - 1 mod 11

= (-1+11) mod 11

= 10 mod 11

= 10

Now Chinese Remainder Theorem will be called for 4 times to produce four plaintexts.

P1 ← Chinese_Remainder(a_1, b_1, p, q)
P2 ← Chinese_Remainder(a_1, b_2, p, q)
P3 ← Chinese_Remainder(a_2, b_1, p, q)
P4 ← Chinese_Remainder(a_2, b_2, p, q)

P1 ← Chinese_Remainder (a_1, b_1, p, q)
P1 ← Chinese_Remainder (4, 1, 7, 11)

$x \equiv a_1 \bmod p$

$x \equiv b_1 \bmod q$

$M = p * q = 7 * 11 = 77 \quad \Rightarrow \quad M = 77$

$M_1 = \frac{M}{m1} = \frac{77}{7} = 11 \quad \Rightarrow \quad M_1 = 11$

$M_2 = \frac{M}{m2} = \frac{77}{11} = 7 \quad \Rightarrow \quad M_2 = 7$

$(M_1 * M_1^{-1}) \bmod p = 1$

$(11 * \underline{2}) \bmod 7 = 1 \quad \Rightarrow \quad M_1^{-1} = 2$

$(M_2 * M_2^{-1}) \bmod q = 1$

$(7 * \underline{8}) \bmod 11 = 1 \quad \Rightarrow \quad M_2^{-1} = 8$

P1 = $[(a_1 * M_1 * M_1^{-1}) + (b_1 * M_2 * M_2^{-1})] \bmod M$

= [(4 * 11 * 2) + (1 * 7 * 8)] mod 77

= (88 + 56) mod 77

= 144 mod 77

= 67

P2 ← Chinese_Remainder(a_1, b_2, p, q)
P2 ← Chinese_Remainder(4, 10, 7, 11)

$x \equiv 4 \bmod 7$

$x \equiv 10 \bmod 11$

As moduli p and q are common for all above four cases, M, M_1, M_2, M_1^{-1} and M_2^{-1} also common.

$P2 = [(a_1 * M_1 * M_1^{-1}) + (b_2 * M_2 * M_2^{-1})] \bmod M$

$= [(4 * 11 * 2) + (10 * 7 * 8)] \bmod 77$

$= (88 + 560) \bmod 77$

$= 144 \bmod 77$

$= 32$

P3 ← Chinese_Remainder(a_2, b_1, p, q)
P3 ← Chinese_Remainder(3, 1, 7, 11)

$x \equiv 3 \bmod 7$

$x \equiv 1 \bmod 11$

$P3 = [(a_2 * M_1 * M_1^{-1}) + (b_1 * M_2 * M_2^{-1})] \bmod M$

$= [(3 * 11 * 2) + (1 * 7 * 8)] \bmod 77$

$= (66 + 56) \bmod 77$

$= 122 \bmod 77$

$= 45$

P4 ← Chinese_Remainder(a_2, b_2, p, q)
P4 ← Chinese_Remainder(3, 10, 7, 11)

$x \equiv 3 \bmod 7$

$x \equiv 10 \bmod 11$

$P4 = [(a_2 * M_1 * M_1^{-1}) + (b_2 * M_2 * M_2^{-1})] \bmod M$

$= [(3 * 11 * 2) + (10 * 7 * 8)] \bmod 77$

$= (66 + 560) \bmod 77$

$= 626 \bmod 77$

$= 10$

Therefore, the four possible plaintexts are P1 = 67, P2 = 32, P3 = 45, P4 = 10. Among these, P4 = 10 is the original plaintext.

Example:

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

5. Bob receives 93 and calculates four values:

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

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

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

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

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.

Calculations are shown below:

P1 ← Chinese_Remainder (a_1, b_1, p, q)
P1 ← Chinese_Remainder (1, 4, 23, 7)

$x \equiv a_1 \bmod p$

$x \equiv b_1 \bmod q$

$M = p * q = 23 * 7 = 161 \quad \Rightarrow \quad M = 161$

$M_1 = \frac{M}{m1} = \frac{161}{23} = 7 \quad \Rightarrow \quad M_1 = 7$

$M_2 = \frac{M}{m2} = \frac{161}{7} = 23 \quad \Rightarrow \quad M_2 = 23$

$(M_1 * M_1^{-1}) \bmod p = 1$

$(7 * \underline{10}) \bmod 23 = 1 \quad \Rightarrow \quad M_1^{-1} = 10$

$(M_2 * M_2^{-1}) \bmod q = 1$

$(23 * \underline{4}) \bmod 7 = 1 \quad \Rightarrow \quad M_2^{-1} = 4$

$P1 = [(a_1 * M_1 * M_1^{-1}) + (b_1 * M_2 * M_2^{-1})] \bmod M$

$= [(1 * 7 * 10) + (4 * 23 * 4)] \bmod 161$

$= (70 + 368) \bmod 161$

$= 438 \bmod 161$

$= 116$

P2 ← Chinese_Remainder(a_1, b_2, p, q)
P2 ← Chinese_Remainder(1, 3, 23, 7)

$x \equiv 1 \bmod 23$

$x \equiv 3 \bmod 7$

As moduli p and q are common for all above four cases, M, M_1, M_2, M_1^{-1} and M_2^{-1} also common.

$P2 = [(a_1 * M_1 * M_1^{-1}) + (b_2 * M_2 * M_2^{-1})] \bmod M$

$= [(1 * 7 * 10) + (3 * 23 * 4)] \bmod 161$

$= (70 + 276) \bmod 161$

= 346 mod 161

= 24

P3 ← Chinese_Remainder(a_2, b_1, p, q)
P3 ← Chinese_Remainder(22, 4, 23, 7)

x ≡ 22 mod 23

x ≡ 4 mod 7

P3 = $[(a_2 * M_1 * M_1^{-1}) + (b_1 * M_2 * M_2^{-1})]$ mod M

= [(22 * 7 * 10) + (4 * 23 * 4)] mod 161

= (1540 + 368) mod 161

= 1908 mod 161

= 137

P4 ← Chinese_Remainder(a_2, b_2, p, q)
P4 ← Chinese_Remainder(22, 3, 23, 7)

x ≡ 22 mod 23

x ≡ 3 mod 7

P4 = $[(a_2 * M_1 * M_1^{-1}) + (b_2 * M_2 * M_2^{-1})]$ mod M

= [(22 * 7 * 10) + (3 * 23 * 4)] mod 161

= (1540 + 276) mod 161

= 1816 mod 161

= 45

Therefore, the four possible plaintexts are P1 = 116, P2 = 24, P3 = 137, P4 = 45. Among these, P2 = 24 is the original plaintext.

Exercise:
Using the Robin cryptosystem with p = 47, q = 11:
a. Encrypt P = 17 to find the ciphertext.
b. Use the Chinese Remainder theorem to find four possible plaintexts

4. ElGamal Cryptosystem

Besides RSA and Rabin, another public-key cryptosystem is ElGamal, named after its inventor, Taher ElGamal. ElGamal is based on the discrete logarithm problem discussed in the previous chapter.

Figure 10.9 shows key generation, encryption, and decryption in ElGamal.

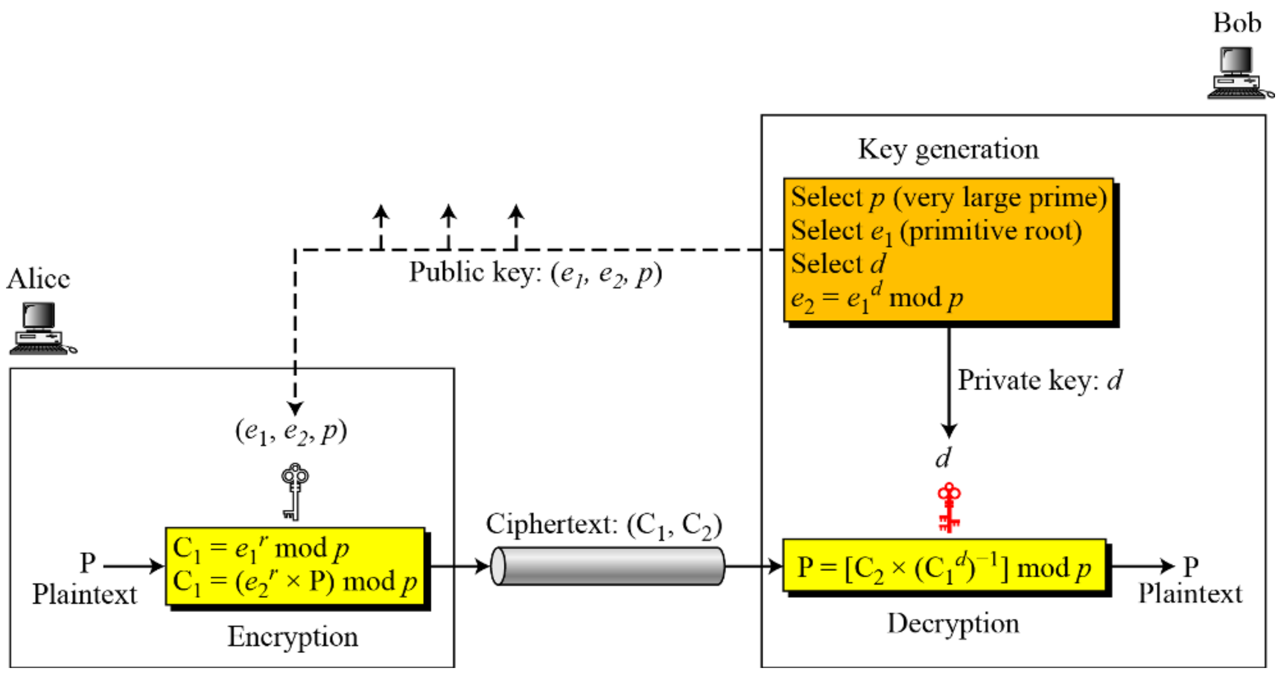


Figure 10.9 Key generation, encryption, and decryption in ElGamal

```
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: ElGamal key generation

```
ElGamal_Encryption $(e_1, e_2, p, P)$                  // P is the plaintext
{
    Select a random integer $r$ in the group $\mathbf{G} = <\mathbf{Z}_p^*, \times>$
    $C_1 \leftarrow e_1^r \bmod p$
    $C_2 \leftarrow (P \times e_2^r) \bmod p$                    // $C_1$ and $C_2$ are the ciphertexts
    return $C_1$ and $C_2$
}
```

Algorithm: ElGamal encryption

```
ElGamal_Decryption ($d$, $p$, $C_1$, $C_2$)          // $C_1$ and $C_2$ are the ciphertexts
{
    P ← $[C_2 (C_1^d)^{-1}] \bmod p$                 // P is the plaintext
    return P
}
```

Algorithm: ElGamal decryption

Example:

Here is a trivial example. Bob chooses $p = 11$ and $e_1 = 2$ and $d = 3$; $e_2 = e_1^d \bmod 11 = 8$. 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.

Encryption

$C1 = e_1^r \bmod 11 = 2^4 \bmod 11 = 5 \bmod 11$

$C2 = P \times e_2^r \bmod 11 = 7 \times 8^4 \bmod 11 = 7 \times 4096 \bmod 11 = 6 \bmod 11$

Ciphertext: (5, 6)

Bob receives the ciphertext (5 and 6) and calculates the plaintext.

$P = C_2 \times (C_1^d)^{-1} \bmod p$

$= C_2 \times (C_1)^{p-1-d} \bmod p$ [according to Fermat's little theorem]

$= 6 \times 5^{11-1-3} \bmod 11$

$= 7$

For the ElGamal cryptosystem, *p* must be at least 300 digits and *r* must be new for each encipherment.

5. Elliptic Curve Cryptosystems

Although RSA and ElGamal are secure asymmetric-key cryptosystems, their security comes with their large keys. Elliptic curve cryptosystem (ECC) gives the same level of security with smaller key sizes.

Comparable key sizes for equivalent security

Symmetric scheme (key size in bits)	ECC-based scheme (size of n in bits)	RSA/DSA (modulus size in bits)
56	112	512
:	:	:

: 128 : : 256	: 256 : : 512	: 3072 : : 15360

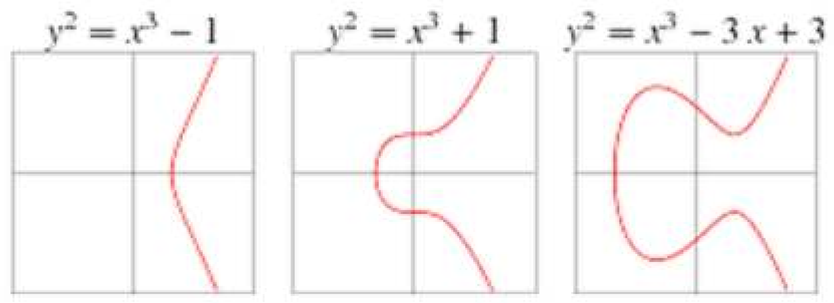
The elliptic curves over real numbers is of the form

$y^2 = x^3 + ax + b$ [for some fixed values for the parameters a and b]

The curve is non-singular which means that the curve has no self-intersections.

Since the curve is symmetrical about the x-axis, given any point P, we can take −P to be the point opposite it. We take −O to be just O.

The example elliptic curves are given here



Expressing the elliptic curve in the form $E_p(a, b)$ means that we have an elliptic curve with parameters a and b and we need to operate over mod p where p is a prime number.

Example:
Find points on the elliptic curve $E_{13}(1, 6)$

Solution:

In the above problem a = 1, b = 6, p = 11

The general elliptic curve form is $y^2 = x^3 + ax + b$

Substitute a and b values the equation changed to $y^2 = x^3 + x + 6$

The calculation should be done over mod 11.

So x and y values should only vary from 0 to 10 - because $Z_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

We need to find the values of x and y such that LHS equals RHS. When LHS = RHS that point will lie on the elliptic curve.

RHS	
x	$x^3 + x + 6 \pmod{11}$
0	6
1	8
2	5
3	3
4	8
5	4
6	8
7	4
8	9
9	7
10	4

LHS	
y	$y^2 \pmod{11}$
0	0
1	1
2	4
3	9
4	5
5	3
6	3
7	5
8	9
9	4
10	1

The points on the elliptic curve are

(2, 4)	(2, 7)
(3, 5)	(3, 6)
(5, 2)	(5, 9)
(7, 2)	(7, 9)
(8, 3)	(8, 8)
(10, 2)	(10, 9)

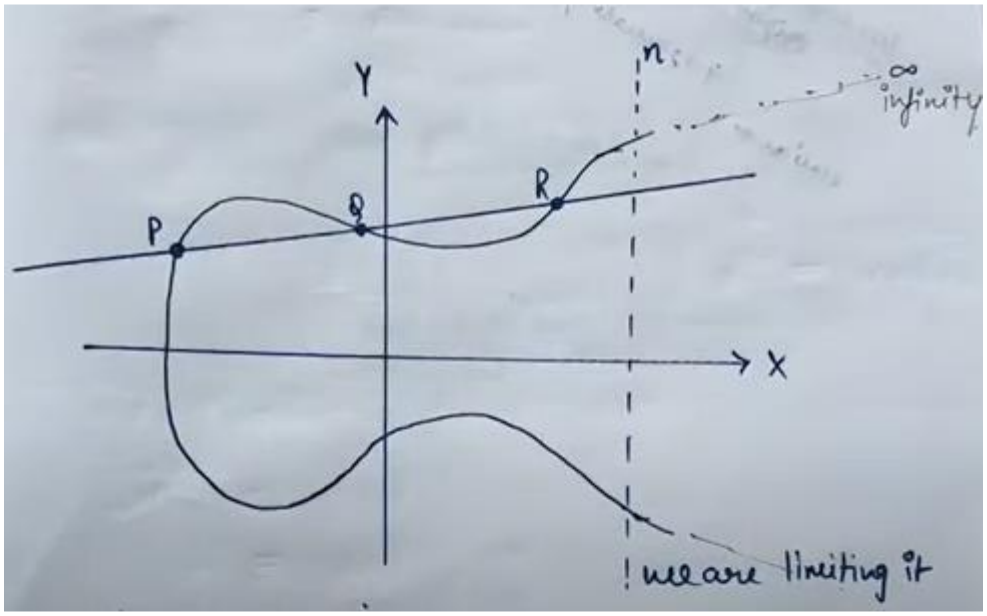
Exercise:
Find the points on the elliptic curve $y^2 = x^3 + x + 1$ calculation should be done over modulo 13.

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, -2 mod 13) = (4, 11).

Elliptic Curve Cryptography (ECC)

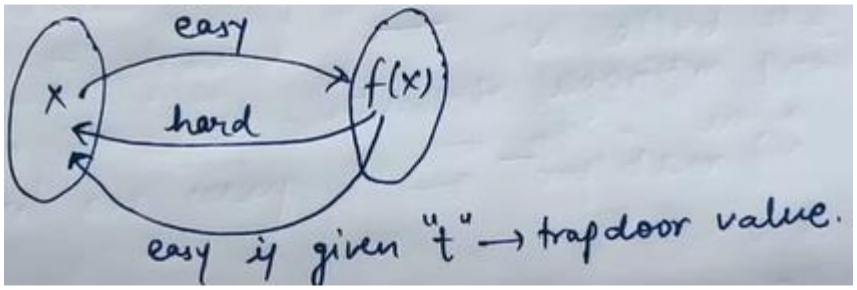
(https://www.youtube.com/watch?v=0NGPhAPKYv4)

- It is asymmetric | public key cryptosystem
- It provides equal security with smaller key size (ex: as compared to RSA) as compared to non-ECC algorithms. i.e. small key size and high security



- It makes use of Elliptic curves
- Elliptic curves are symmetric about x-axis and both ends of the curve goes to infinity but we are limiting it to n
- Elliptic curves are defined by some mathematical functions – cubic function
- Ex: $y^2 = x^3 + ax + b$ //a and b are constants, and it is equation of degree 3
- Symmetric to x – axis
- If we draw a line it will touch a maximum of three points

A trapdoor function
A trapdoor function is a function that is easy to compute in one direction, yet difficult to compute in the opposite direction (finding its inverse) without a special information, called the trapdoor.



- Let $E_p(a, b)$ be the elliptic curve
- Consider the equation $Q = kP$ where Q and P are points on curve and $k < n$.
- If k and P given, it should be easy to find Q but if we know Q and P, it should be extremely difficult to find k. This is called the discrete logarithm problem for elliptic curves. i.e. it is a one-way function or trap door junction. $A \rightarrow B$ is very easy but coming $B \rightarrow A$ is very difficult.

Elliptic Curve Cryptography (ECC) Algorithm

Global public elements
- **$E_q(a, b)$**: Elliptic curve with parameters a, b and q where q is prime number or of the form 2^m
- **G**: point on the elliptic curve whose order is greater than value of n (>n).

(Let us assume A is sender and B is receiver)

User A key generation
- Select private key n_A such that $n_A < n$
- Calculate public key of A $P_A = n_A * G$

User B key generation
- Select private key n_B such that $n_B < n$
- Calculate public key $P_B = n_B * G$

Calculation of secret key by user A
- $k = n_A * P_B$

Calculation of secret key by user B
- $k = n_B * P_A$

ECC Encryption
Let the message be M

First encode this message M into a point on elliptic curve

Let this point be P_m, now this point is encrypted

For encryption, choose a random positive integer k then the cipher point will be

$C_m = \{kG, P_m + kP_B\}$

This point will be sent to the receiver.

ECC Decryption

For decryption, multiply 1st coordinate in the point with receiver's secret key

i.e. $kG * n_B$ (for decryption private key of B is used)

then subtract it from 2nd coordinate in the pair

i.e $P_m + kP_B - (kG * n_B)$

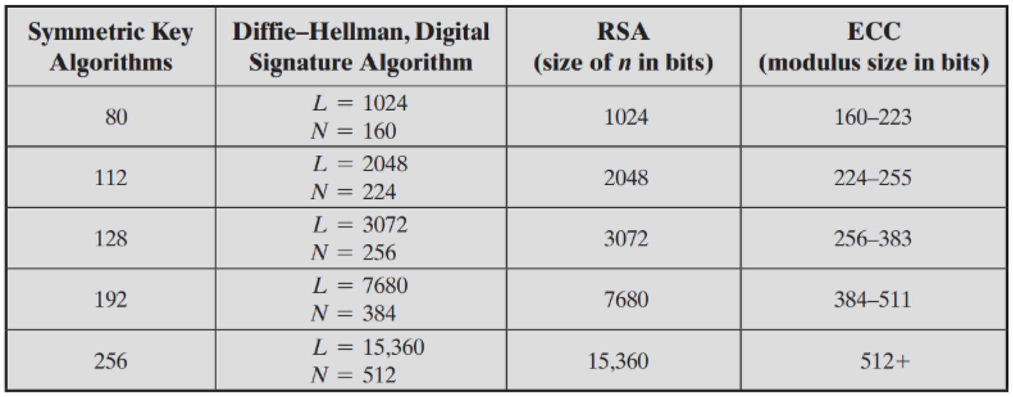
but we know $P_B = n_B * G$

so, $= P_m, - kP_B - kP_B$

$= P_m$ (Original Point)

So, receiver gets the same point

Security of Elliptic Curve Cryptography



Symmetric Key Algorithms	Diffie–Hellman, Digital Signature Algorithm	RSA (size of *n* in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

Note: L = size of public key, N = size of private key.

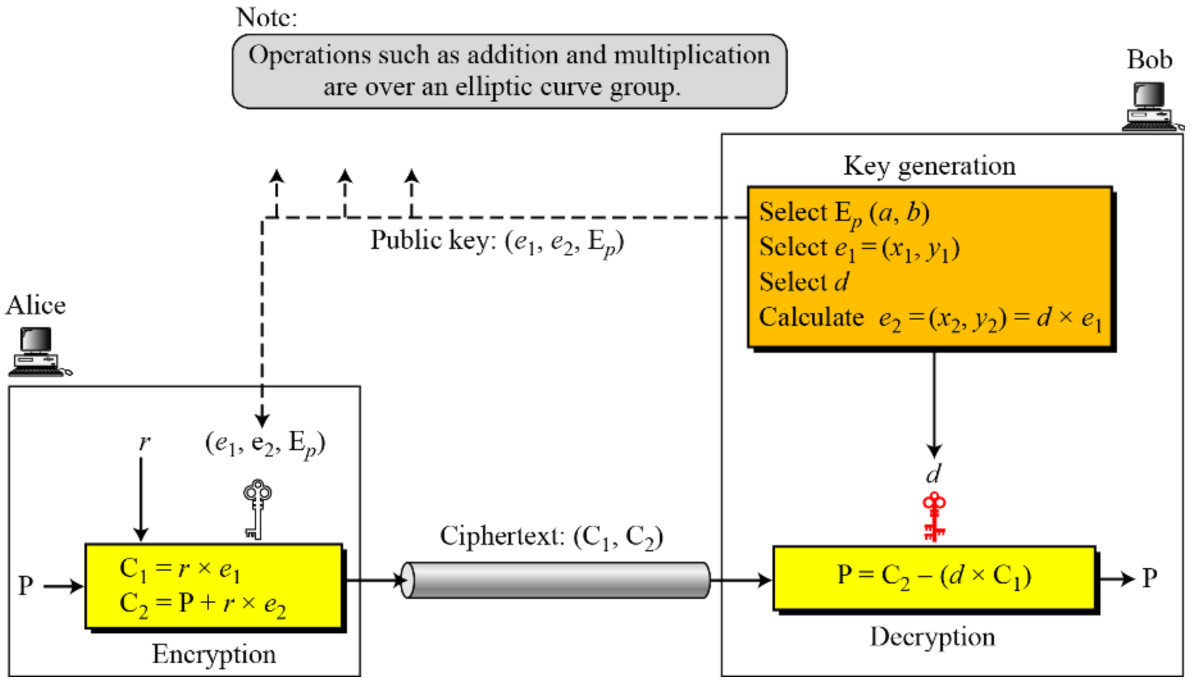
Table 1: Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis

The security of ECC depends on how difficult it is to determine k given kP and P. This is referred to as the elliptic curve logarithm problem. Table 1, compares various algorithms by showing comparable key sizes in terms of computational effort for cryptanalysis. As can be seen, a considerably smaller key size can be used for ECC compared to RSA.

Analysis indicates that for equal key lengths, the computational effort required for ECC and RSA is comparable. Thus, there is a computational advantage to using ECC with a shorter key length than a comparably secure RSA.

Elliptic Curve Cryptography Simulating ElGamal

Several methods have been used to encrypt and decrypt using elliptic curves. The common one is to simulate the ElGamal cryptosystem using an elliptic curve over GF(p) or GF(2^n), as shown in figure 10.10.



Note:
Operations such as addition and multiplication are over an elliptic curve group.
Bob
Key generation
Select $E_p(a, b)$
Select $e_1 = (x_1, y_1)$
Select d
Calculate $e_2 = (x_2, y_2) = d \times e_1$
Public key: (e_1, e_2, E_p)
Alice
r
(e_1, e_2, E_p)
P
$C_1 = r \times e_1$
$C_2 = P + r \times e_2$
Encryption
Ciphertext: (C_1, C_2)
d
$P = C_2 - (d \times C_1)$
P
Decryption

Figure 10.10 ElGamal cryptosystem using the elliptic curve

Generating Public and Private Keys

1. Bob chooses E(a, b) with an elliptic curve over FG(p) or GF(2^n).
2. Bob chooses a point on the curve, $e_1(x_1, y_1)$
3. Bob chooses an integer d.
4. Bob calculates $e_2(x_2, y_2) = d \times e_1(x_1, y_1)$
5. Bob announces E(a, b), $e_1(x_1, y_1)$, and $e_2(x_2, y_2)$ as his public key; he keeps d as his private key

Encryption

Alice selects P, a point on the curve, as her plaintext, P. She then calculates a pair of points on the text as ciphertexts:

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

Decryption

Bob, after receiving C1 and C2, calculates P, the plaintext using the following formula.

$$P = C_2 - (d \times C_1)$$

Example:

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 4 x C_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).