UNIT II

INTEGER ARITHMETIC

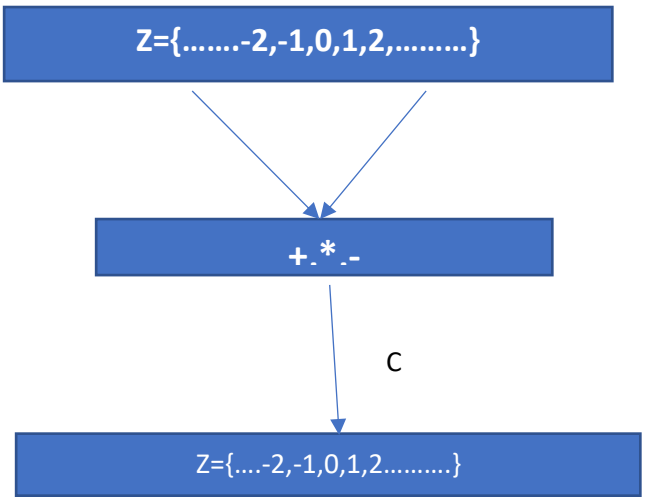
An integer arithmetic we use a set of two operations like set of integers.

Set of integers is denoted by Z contains all integer numbers from –ve infinity to +ve inifinity.

Z={-5,-4,-3,-2,-1,0.1.2.3.4,5}

Binary Operation :

In cryptography we are interested in 3 binary operations applied to the set of integers a binary operations will takes two inputs and creates one output. 3 common binary operations are Addition, Substraction, Multiplication. The two inputs comes from the set of integers the output goes into the set of integers.



Ex : : 2+3=5, 2*4=8, 5-2=3

Integer Division

In integer arithmetic the division operation is different from the add, sub,mul because in division we are taking 2 inputs and we are getting 2 outputs means, if we divide a/n. we get Q & r the relationship between these 4 integers are

a=n*Q+r

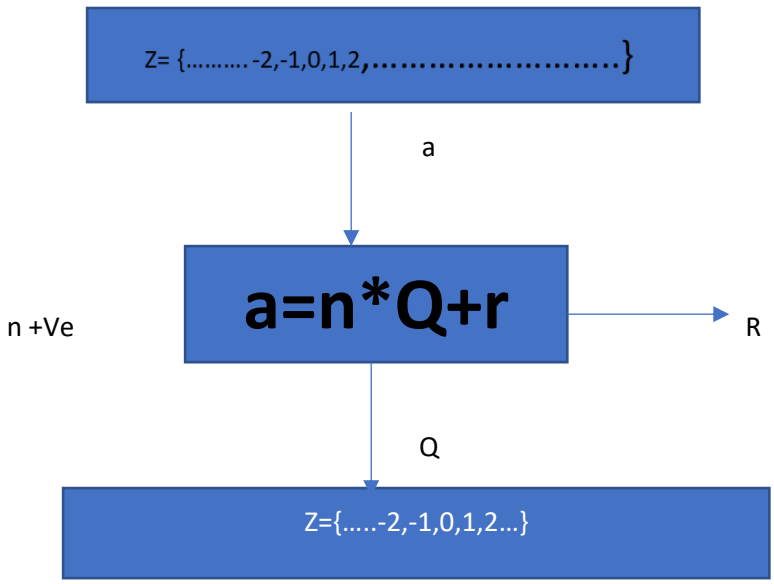
In this a is the dividend n is divisor Q is quotient & r is remainder.

Two Restrictions :

We have two restrictions for in the division of cryptography are :

i. Divisor a +ve integer

ii. The remainder should be non negative integer

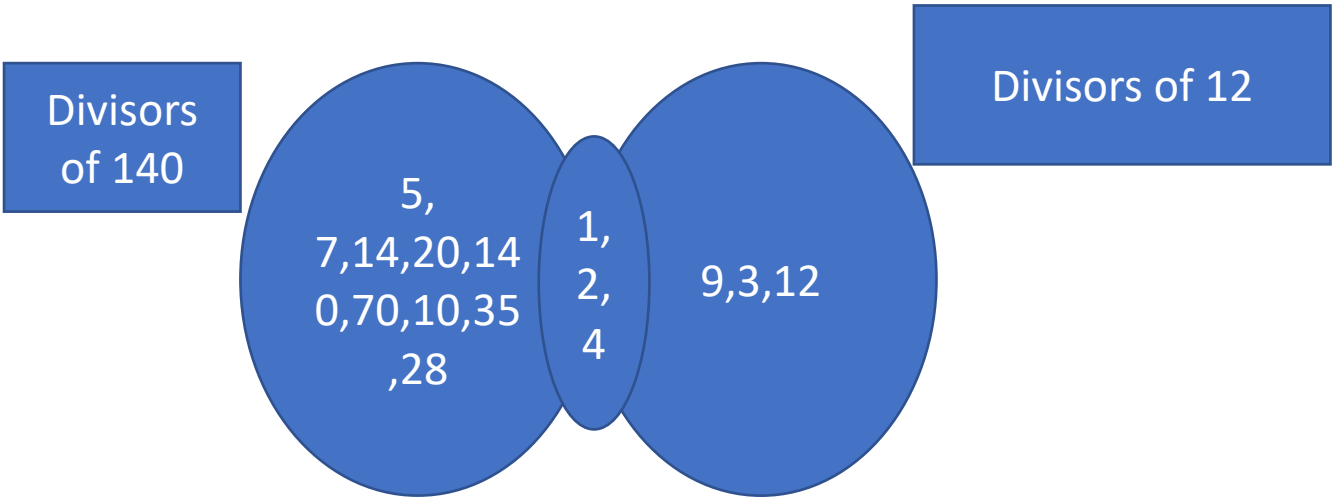


Divisibility means if A is not zero & r=0 in the division relation we get A=Q*n . We say that n divides a or a is divisible by n. if we are not interested in the value of Q. we can write the above relationship as a/n. if the remainder is not zero then n doesnot divide a and write the relationship as a / n.

Greatest Common Divisor -

One integer often needed in cryptography is the GCD of two +ve integers may have many common divisors but only one greatest common divisor.

Def of GCD: the GCD of 2 +ve integers is the largest integer that can divide both integers. For example: the common divisors of 12 & 140 are 1,2 & 4 among this greatest common divisor is 4.



Eucledian Algorithm:

Find the GCD of 2 +ve integers by listing all common divisors is not practical. When the two integers are large more than 2000 years ago a mathematician named Euclid developed an algorithm that can find the GCD of 2+ve integers the Euclidian algorithm is based on the following 2 facts.

Fact 1: gcd(a,0)=a

Fact 2: gcd(a,b)=gcd(b,r) where r is the remainder.

The first fact tells that if 2nd integer is 0 then gcd is the 1st one.

The second fact tells that it allows us to change the value of a,b until b becomes zero.

For eg: N=gcd(36,10)

Gcd(36,10)=2

Gcd(36,10)= gcd(10,6) =gcd(6,4)= gcd(4,2)=gcd(2,0)

So gcd(36,10)=2

Euclidean Algorithm

r1-→a, r2→b

while(r2>0)

{

q1←r1/r2;

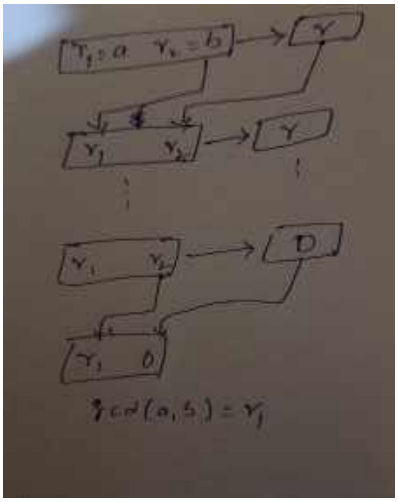
r←r1-q*r2;

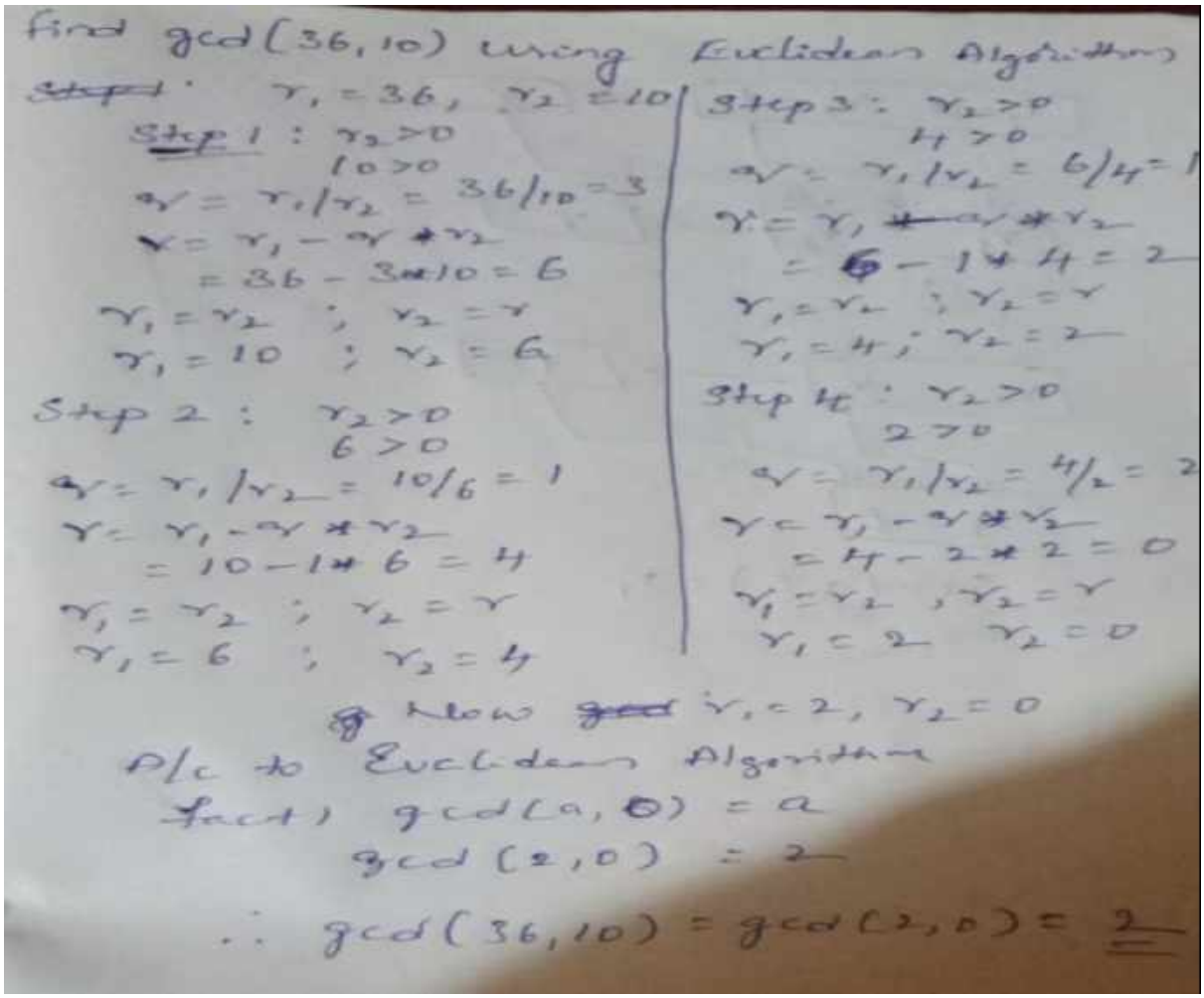
r1←r2;

r2←r;

}

Eucledean Process:





Extended Euclidean Algorithm

Extended Euclidean algorithm consists of two integers a & b will be given. we often find other two integers such that s*a+t*b=gcd(a,b) the extended Euclidean algorithm can calculate the gcd and at the same time it calculates the value of s & t. the extended algorithm uses the same number of steps

as the Euclidean algorithm in each step it uses 3 sets of calculations and exchanges instead of 1. the algorithm for extended Euclidean algorithm is

```
r1 ← a, r2 ←b
s1←1 , s2← 0
t1←0, t2←1
while(r2>0)
{
  q←r1/r2
R=r1-q*r2
   r1←r2; r2←r;
s←s1-q*s2;
s1←s2; s2←s;
t←t1-q*t2;
t1←t2; t2←t
}
gcd(a,b) ←r1;
s←s1; t←t1;
```

Example: if a=161, b=28 find gcd(a,b) & find values of s & t

A=161 b=28
S1=1 , s2=0
T1=0, t2=1
Q=r1/r2=161/28=5
R=r1-q*r2
R=161-(5*25)=21
R1=r2; r2=r
R1=28, r2=21
s=s1-q*s2
=1-(5*0)=1
S1=s2, s2=s
S1=0, s2=1
T=t1-q*t2
=>0-(5*1)=-5
T1=t2; t2=t
T1=1; t2=-5
S1=0, s2=1, t1=1, t2=-5
q=r1/r2=1
R=28-(1*21)=7
R1=21, r2=7
S=s1-q*s2
=0-(1*1)= -1
S1=1, s2=-1
T=t1-q*t2

=1-(1*(-5))=6
T1=-5 t2=6
R1=21, r2=7
S1=1, s2=-1 & t1=-5 & t2=6
Q=r1/r2= 21/7=3
R=r1-q*r2
=21-(3*7)=0
R1=r2, r2=r
R1=7 , r2=0
S=s1-q*s2
=1-(3*(-1))=4
T=t1-q*t2
=-5-(3*6)
=-23
T1=t2, t2=t
T1=6, t2=-23
S*a+t*b=gcd(161,28)
-1*161+6*2 8=7
There fore 7=7

Linear Diophantine Equation

It is very important application of the extended Euclidean algorithm in this two variables an equation of type ax+by=c will be considered & we need to find the integer values for x & y that satisfy the equation in this type we get either no solution or an infinity number of solutions.

Let D=gcd(a,b) if d is not divided by c then the equation has no solution or if d is divided by c then we have an infinite number of solutions one of them is called particular and the rest are general.

Particular Solutions:

If d is divided by c is a particular solution can be found using the following steps

i. Reduce the equation to a1x+b1y=c1 by dividing both sides of the equation by d
ii. Solve for s & t in the relation a1s+b1t=1 using the extended Euclidean algorithm.
iii. Particular solution can be found by the following the equation

x0=(c/d)s

$$y0=(c/d)t$$

General Solutions:

After finding the particular solution the general solutions can be find

$$X=x0+k(b/d)$$

$$Y=y0-k(a/d)$$

Where k is the integer

Find the particular Solution & General Solution for the equation $21x + 14y = 35$

gcd(21, 14)

$21x + 14y = 35$

$3x + 2y = 5$

It will be considered as

$3s + 2t = 1$

$A = 21, \quad B = 14 \quad s_1 = 1, \quad s_2 = 0$

$q = r_1 / r_2 \quad t_1 = 0 \quad t_2 = 1$

$= 21/14 = 1$

$r = r_1 - q * r_2$

$= 21 - 1 * 14 = 7$

$r_1 = r_2, \quad r_2 = r$

$r_1 = 14, \quad r_2 = 7$

$s = s_1 - q * s_2$

$= 1 - 1 * 0 = 1$

$s_1 = s_2, \quad s_2 = s$

$s_1 = 0, \quad s_2 = 1$

$t = t_1 - q * t_2$

$= 0 - 1 * (1) = -1$

$t_1 = t_2, \quad t_2 = t$

$t_1 = 1, \quad t_2 = -1$

$x_0 = (c/d)\, s$

$= 5 * 1 = 5$

$y_0 = (c/d)\, t$

$= 5 * (-1) = -5$

$c/d = 35/7 = 5$

$x = x_0 + k(b/d)$

$= 5 + 2(2)$

$x = 9$

when $k = 2$

$y = y_0 - k(a/d)$

$= -5 - 2(3)$

$= -11$

Example if a=161 b=28 find gcd(a,b) and find vales of s & t
A=161, b=28
S1=1, s2=0
T1=0, t2=1
R1=161, r2=28
Round 1
R2>0
Q=r1/r2=161/28=5
R=r1-1*r2= 161-5*28=21
R1=r2, r2=r
R1=28, r2=21
S=s1-q*s2= 1-5*0=1
S1=s2, s2=s
S1=0, s2=1
T=t1-q*t2=0-(5*1)=**-5**
T1=t2, t2=t
T1=1, t2=-5
Round 2
R2>0
Q=r1/r2=28/21=1
R=r1-q*r2=28-(1*21)=7
R1=r2 r2=r
R1=21, r2=7
S=s1-q*s2=0-(1*1)=-1
S1=s2, s2=s
S1=1, s2=-1
T=t1-q*t2=1-(1*-5)=6
T1=t2, t2=t
T1=-5, t2=6
Round 3
R2>0
Q=r1/r2=21/7=3
R=r1-q*r2=21-3*7=0
R1=r2, r2=r
R1=7, r2=0
S=s1-q*s2= 1-(3*-1)=4
S1=s2, s2=s

S1=-1, s2=4
T=t1-q*t2=-5-(3*6)=-23
T1=t2,t2=t
T1=6, t2=-23
R1 =7, r2=0 (according fact 1 of Euclidean gcd(a,0)=a
Gcd(7,0)=7
S=s1, t=t1
S=-1, t=6

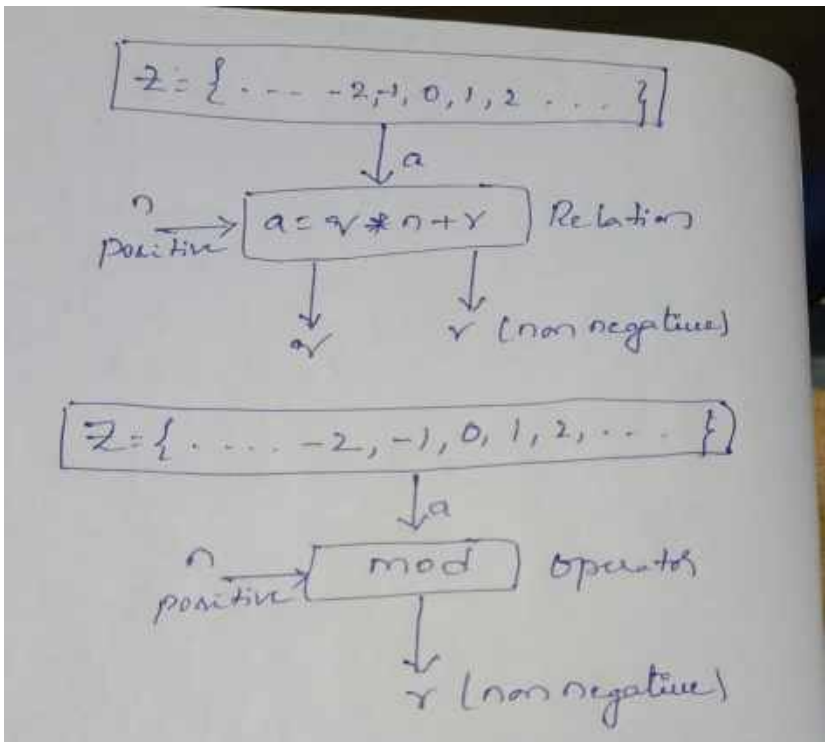
S*a+t*b= -1*161+6*28=7
S*a+t*b=gcd(a,b)
7=7

Modular Arithmetic:

The division relationship a=q*n+r which has 2 inputs a & n & 2 output's in modular arithmetic we will consider only the remainder or by neglecting the quotient Q in simple we use the modular arithmetic to know the remainder value of a & n.

Modulo Operator:

In modular arithmetic we use a binary operator called modulo operator represented as mod for getting the R value the 2nd input N is called modulous the output R is called Residue.



For example find the result of the following operations

a. 27 mod 5
b. 36 mod 12
c. -18 mod 14
d. -7 mod 10

Solution:

We are looking for the residue r, we can divide the a by n and find q and r we can then dieregard q and keep r

a. Dividing 27 by 5 results in r=2. this means that 27 mod 5=2
b. Dividing 36 by 12 results in r=0. this means that 36 mod 12=0
c. Dividing -18 by 14 results in r=-4. how ever we need to add the modulus 14 to make it non negative. We have

r= -4 + 14=10. this means -18 mod 14=10

a. d. Dividing -7 y 10 results in r=-7 how ever we need to add the modulus 10 to make it non negative. We have

r= -7 + 10=3. this means -7 mod 10 =3

Set of Residues:

The result of the modulous operation with modulus n is always an integer between 0 & n-1 in otherwords the result of a mod n is always a non negative integer less than n. the modulo operation creates a set arithmetic is referred to as the set of least residues modulo n or Zn.

For example Z2, Z6, Z11 are

Zn = {0,1,2,3,4,……n-1}

Z2={0,1} Z6={0,1,2,3,4,5} Z11={0,1,2,3,4,5,6,7,8,9,10}

Congruence:

In cryptography we often used the concept of congruence instead of equality mapping from Z to Zn is not 1 to 1 infinite numbers of Z can map to one member of Zn.

For example : 2 mod 10=2, 12 mod 10=2, 22 mod 10=2 here the integers like 2,12,22 are called congruent for the modulo 10 . To show the 2 integers are congruent we use the congruence operator (≅) we add mod n to the right side of the congruence to define the values of modulus that makes the relationship valid.

$2 \cong 12\ (mod\ 10) \quad 13 \cong 23 (mod\ 10)$

Residue Classes:

A residue a or an is the set of integers congruent to modulo n in other words it is the set of all integers such that x=a mod n for example if n=5 we have 5 sets {0}, {1},{2},{3},{4} & each set will be shown as

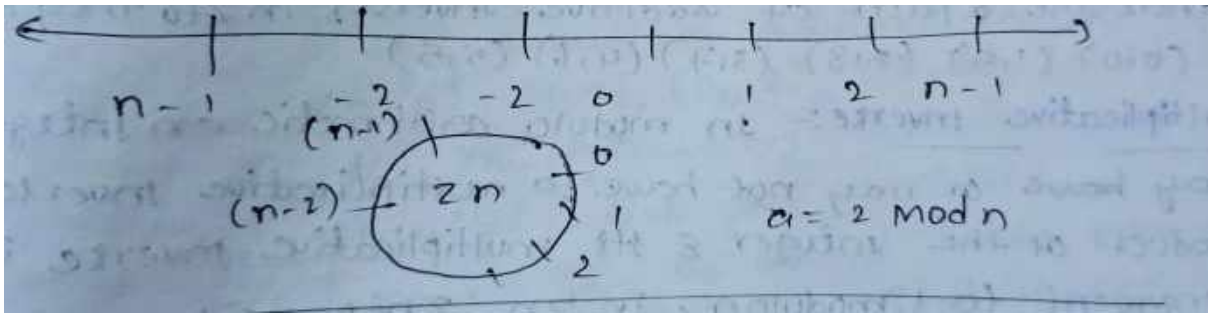
{0}= {….-5,-,4,-3,-2,-,1,0,1,2,3,4,5,….}

{1}= {….-5,-,4,-3,-2,-,1,0,1,2,3,4,5,….}

{2}= {….-5,-,4,-3,-2,-,1,0,1,2,3,4,5,….}

Circular Notation:

The concept of congruence can be explained with the help of circle in this just we show a line for distribution for integers in Z we can even show a line for the distribution of integers for Zn all the integers of congruent modulo n occupy the same point on the circle. And are mapped to the circle in such a way that there is a symmetry between them.



Operations in Zn:

The 3 binary operations Add,Sub,Mul which are defined for the set z can also be explained for the set Zn. The result to be mapped Zn using the modulo operator the properties for the binary operations in modulo arithmetic from z to Zn are

First Property : (a+b)mod n = ((amod n)+(bmodn)) mod n

Second Property : (a-b)mod n=((amodn)-(b mod n))mod n

Third Property : (a*b)mod n=((amod n)*(b mod n)) mod n

Example:

Add 7 to 14 in Z15

(14+7)mod 15= 21 mod 15=6

Sub 11 from 7 in Z13

(7-11) mod 13=(-4) mod 13=9

Multiply 11 by 7 in Z20

(7*11)mod 20 =(77)mod 20 =17

Here are examples of the three properties:

$11 \bmod 8 = 3; 15 \bmod 8 = 7$

$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$

$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$

$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$

$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$

$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$

$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

Exponentiation is performed by repeated multiplication, as in ordinary arithmetic.

To find $11^7 \bmod 13$, we can proceed as follows:

$$11^2 = 121 \equiv 4 \pmod{13}$$
$$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$$
$$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \pmod{13}$$

Table below provides an illustration of modular addition and multiplication modulo 8.

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Additive and multiplicative inverses modulo 8

Properties of Modular Arithmetic for Integers in Z*n*

Property	Expression
Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x + w) \bmod n$
Associative Laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse ($-w$)	For each $w \in Z_n$, there exists a z such that $w + z = 0 \bmod n$

1.3 Matrices

Definitions:

A matrix is a rectangular array of l × m elements, in which l is the number of rows and m is the number of columns. A matrix is normally denoted with a boldface uppercase letter such as A. The element aij is located in the ith row and jth column. Although the elements can be a set of numbers, we discuss only matrices with elements in Z. Figure below shows a matrix of size l x m:

$$\begin{array}{c} \\ l \text{ rows} \end{array} \overbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{bmatrix}}^{m \text{ columns}}$$

If a matrix has only one row (l = 1), it is called a row matrix; if it has only one column (m = 1), it is called a column matrix.

In a square matrix, in which there is the same number of rows and columns (*l = m), the elements a11, a22, . . . , amm make the* main diagonal.

An additive identity matrix, denoted as 0, is a matrix with all rows and columns set to 0's.

An identity matrix, denoted as I, is a square matrix with 1s on the main diagonal and 0s elsewhere.

Figure below: *Example of matrices*

$$\underset{\text{Row matrix}}{\begin{bmatrix} 2 & 1 & 5 & 11 \end{bmatrix}} \quad \underset{\text{Column matrix}}{\begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}} \quad \underset{\text{Square matrix}}{\begin{bmatrix} 23 & 14 & 56 \\ 12 & 21 & 18 \\ 10 & 8 & 31 \end{bmatrix}} \quad \underset{\mathbf{0}}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}} \quad \underset{\mathbf{I}}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}$$

Operations and Relations

In linear algebra, one relation (equality) and four operations (addition, subtraction, multiplication, and scalar multiplication) are defined for matrices.

Equality: Two matrices are equal if they have the same number of rows and columns and the corresponding elements are equal. In other words, **A = B if we have *aij = bij for all i's and j's.***

Addition and Subtraction: Two matrices can be added if they have the same number of columns and rows. This addition is shown as **C = A + B.** Each element of **C is the sum of the two corresponding** elements of **A and B: *cij = aij + bij.***

Subtraction is the same except that each element of **B is subtracted from the corresponding element of A: *dij = aij – bij.***

Figure below *Addition and subtraction of matrices*

$$\begin{bmatrix} 12 & 4 & 4 \\ 11 & 12 & 30 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} + \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}$$

C = A + B

$$\begin{bmatrix} -2 & 0 & -2 \\ -5 & -8 & 10 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} - \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}$$

D = A – B

Multiplication: We can multiply two matrices of different sizes if the number of columns of the first matrix is the same as the number of rows of the second matrix. If A is an *l × m matrix* and B is an *m × p matrix, the product of the two is a matrix C of size l × p. If each element* of matrix A is called *aij, each element of matrix B is call*

$$c_{ik} = \sum a_{ij} \times b_{jk} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + \cdots + a_{im} \times b_{mj}$$

Figure below *Multiplication of a row matrix by a column matrix*

$$\begin{matrix} \mathbf{C} & & \mathbf{A} & & \mathbf{B} \end{matrix}$$
$$\begin{bmatrix} 53 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 8 \\ 2 \end{bmatrix}$$

In which: $53 = 5 \times 7 + 2 \times 8 + 1 \times 2$

Scalar Multiplication: If A is an *l × m matrix and x* is a scalar, C = *xA is a matrix of size l × m, in which cij = x × aij.*

Determinant: The determinant of a square matrix A of size *m × m denoted as det (A) is a scalar calculated* recursively as shown below:

1. If $m = 1$, $\det(\mathbf{A}) = a_{11}$
2. If $m > 1$, $\det(\mathbf{A}) = \sum_{i=1\ldots m} (-1)^{i+j} \times a_{ij} \times \det(\mathbf{A}_{ij})$

   Where $\mathbf{A}_{ij}$ is a matrix obtained from **A** by deleting the *i*th row and *j*th column.

$$\det\begin{bmatrix}5 & 2\\ 3 & 4\end{bmatrix} = (-1)^{1+1}\times 5\times \det[4] + (-1)^{1+2}\times 2\times \det[3] \longrightarrow 5\times 4 - 2\times 3 = 14$$

$$\text{or} \quad \boxed{\det\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix} = a_{11}\times a_{22} - a_{12}\times a_{21}}$$

$$\det\begin{bmatrix}5 & 2 & 1\\ 3 & 0 & -4\\ 2 & 1 & 6\end{bmatrix} = (-1)^{1+1}\times 5\times \det\begin{bmatrix}0 & -4\\ 1 & 6\end{bmatrix} + (-1)^{1+2}\times 2\times \det\begin{bmatrix}3 & -4\\ 2 & 6\end{bmatrix} + (-1)^{1+3}\times 1\times \det\begin{bmatrix}3 & 0\\ 2 & 1\end{bmatrix}$$

$$= (+1)\times 5\times (+4) \quad + \quad (-1)\times 2\times (24) \quad + \quad (+1)\times 1\times (3) = -25$$

Inverses: Matrices have both additive and multiplicative inverses.

Additive Inverse: The additive inverse of matrix A is another matrix B such that A + B = 0. In other words, we have *bij = − aij for all values of i and j. Normally the additive inverse of A is* defined by −A.

Multiplicative Inverse: The multiplicative inverse is defined only for square matrices. The multiplicative inverse of a square matrix A is a square matrix B such that A × B = B × A = I. Normally the multiplicative inverse of A is defined by A−1.

Residue Matrices: Cryptography uses residue matrices: matrices in all elements are in ***Zn.***

All operations on residue matrices are performed the same as for the integer matrices except that the operations are done in modular arithmetic. One interesting result is that a residue matrix has a multiplicative inverse if the determinant of the matrix has a multiplicative inverse in Z*n. In other words, a residue matrix has a multiplicative inverse if gcd* (det(A), *n) = 1.*

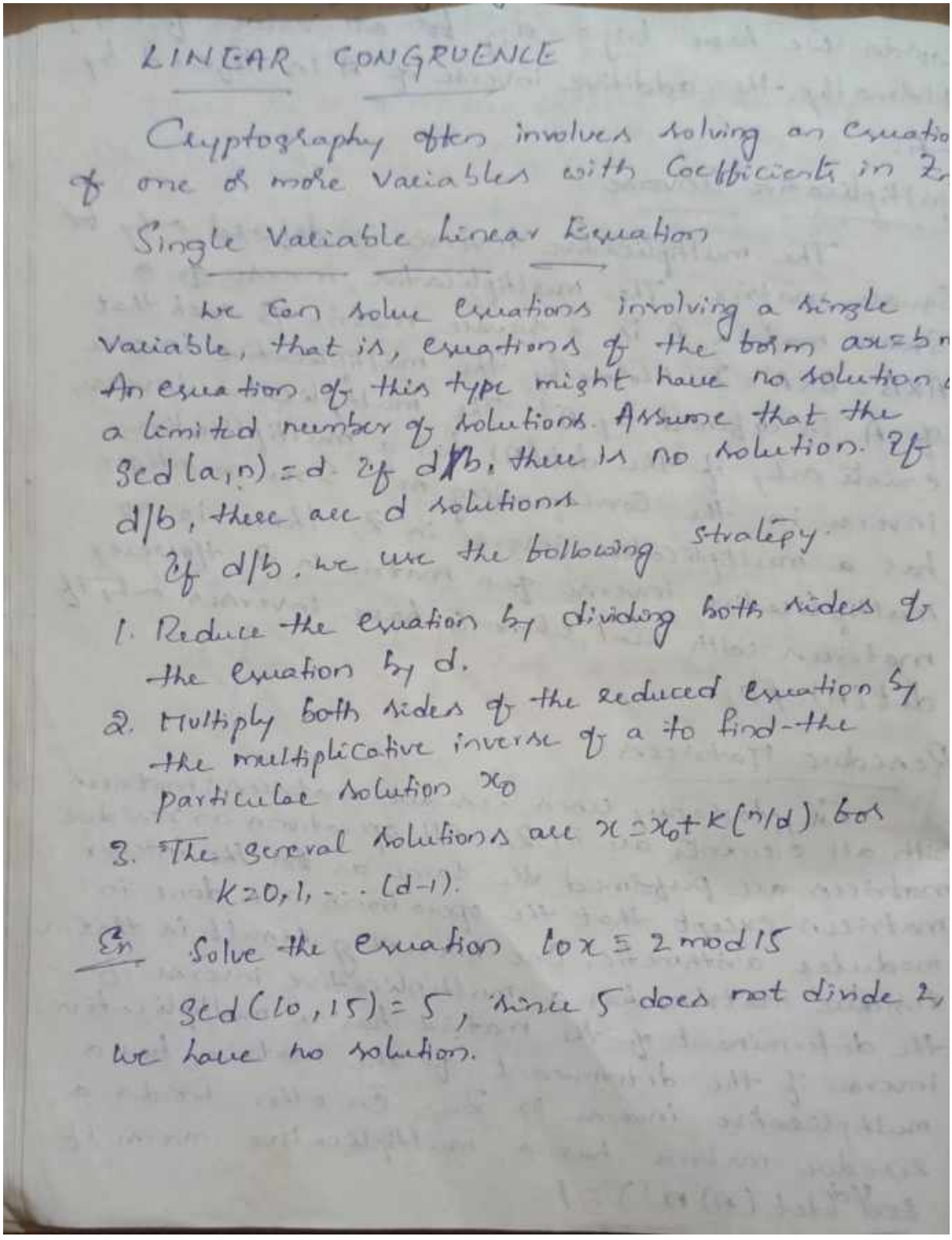
Figure below shows a residue matrix A in Z26 and its multiplicative inverse A−1. We have det(A) = 21 which has the multiplicative inverse 5 in Z26. Note that when we multiply the two matrices, the result is the multiplicative identity matrix in Z26.

$$\mathbf{A} = \begin{bmatrix}3 & 5 & 7 & 2\\ 1 & 4 & 7 & 2\\ 6 & 3 & 9 & 17\\ 13 & 5 & 4 & 16\end{bmatrix} \qquad \mathbf{A}^{-1} = \begin{bmatrix}15 & 21 & 0 & 15\\ 23 & 9 & 0 & 22\\ 15 & 16 & 18 & 3\\ 24 & 7 & 15 & 3\end{bmatrix}$$

$$\det(\mathbf{A}) = 21 \qquad \det(\mathbf{A}^{-1}) = 5$$

Congruence: Two matrices are congruent modulo *n, written as* ***A ≡ B (mod n), if they have the same*** number of rows and columns and all corresponding elements are congruent modulo *n.*

In other words, **A ≡ B (mod *n) if aij ≡ bij (mod n) for all i's and j's.***



LINEAR CONGRUENCE

Cryptography often involves solving an equation of one or more variables with coefficients in Z_n.

Single Variable Linear Equation

We can solve equations involving a single variable, that is, equations of the form $ax \equiv b \pmod{n}$. An equation of this type might have no solution or a limited number of solutions. Assume that the $\gcd(a,n) = d$. If $d \nmid b$, there is no solution. If $d \mid b$, there are d solutions.

If $d \mid b$, we use the following strategy.

1. Reduce the equation by dividing both sides of the equation by d.
2. Multiply both sides of the reduced equation by the multiplicative inverse of a to find the particular solution x_0.
3. The several solutions are $x = x_0 + k(n/d)$ for $k = 0, 1, \ldots (d-1)$.

Ex Solve the equation $10x \equiv 2 \pmod{15}$

$\gcd(10, 15) = 5$, since 5 does not divide 2, we have no solution.

Solve the equation $14x \equiv 12 \bmod 18$

$\gcd(14, 18) = 2$, Since 2 divides 12, we have exactly 2 solutions. but we first we reduce the equation.

$14x \equiv 12 \bmod 18$ ⟵ (1)

divide by 2

$7x \equiv 6 \bmod 9$

$7^{-1} = 4$

has decimal value will be ten value till given $x = 0, 1, 2 \ldots$

$x_0 \equiv 6(7^{-1}) \bmod 9$

$x_0 \equiv 6 \times 4 \bmod 9$

$x * \bmod + 1$

Inverse value

$x_0 \equiv 24 \bmod 9$

$x_0 = 6$

$x = x_0 + k(n/d) \rightarrow k = 0$

$0 * 9 + 1 / 7 = 2$

$= 6 + 0(18/2) = 6$

$1 * 9 + 1 / 7 = 10/7$

$x = x_0 + k(n/d) \rightarrow k = 1$

$2 * 9 + 1 / 7 = 19/10$

$= 6 + 1(18/2)$

$28 * 9 + 1 / 7 = 28/7 = 4$

$= 15$

Algebraic Structures

Cryptography requires sets of integers and specific operations that are defined for those sets. The combination of the set and the operations that are applied to the elements of the set is called an algebraic structure.

Three common algebraic structures: groups, rings, & fields.

Groups:

A group (G) is a set of elements with a binary operation "•" that satisfies four properties. A commutative group, also called an abelian group, is a group in which the operator satisfies the four properties for groups plus an extra property, commutativity. The four properties for groups plus commutativity are defined as

as follows

Closure: If a and b are elements of G, then $c = a \cdot b$ is also an element of G. This means that the result of applying the operation on any two elements in the set is another element in the set.

Associativity: If a, b and c are elements of G, then $(a \cdot b) \cdot c = a \cdot (b \cdot c)$. In other words, it does not matter in which order we apply the operation on more than two elements.

Commutativity: For all a and b in G, we have $a \cdot b = b \cdot a$. This property needs to be satisfied only for a Commutative Group.

Existence of identity : For all a in G, there exists an element e, called the identity element such that $e \cdot a = a \cdot e = a$.

Existence of inverse : For each a in G, there exists an element a', called the inverse of a, such that $a \cdot a' = a' \cdot a = e$.

properties

1. Closure
2. Associativity
3. Commutativity
4. Existence of identity
5. Existence of inverse

Ex The set of residue integers with the addition operator, $G = \langle Z_n, + \rangle$, is a Commutative group. We can perform addition and subtraction on the elements of this set without moving out of the set. Let us check the properties

1. Closure is satisfied. The result of adding two integers in Z_n is another integer in Z_n.
2. Associativity is satisfied. The result of $4+(3+2)$ is the same as $(4+3)+2$.
3. Commutativity is satisfied. We have $3+5=5+3$
4. The identity element is 0. We have $3+0=0+3=3$
5. Every element has an additive inverse. The inverse of an element is its complement. For example, the inverse of 3 is -3 ($n-3$ in Z_n) and the inverse of -3 is 3. The inverse allows us to perform subtraction on the set.

Finite Group

A group is called a finite group if the set has a finite number of elements, otherwise, it is an infinite group.

Order of a Group

The order of a group, |G| is the number of elements in the group. If the group is not finite, its order is infinite; if the group is finite the order is finite.

Sub Groups

A subset H of a group G is a subgroup ~~with respect~~ of G if H itself is a group with respect to the operation on G. In other words, if $G = \langle S, \bullet \rangle$ is a group, $H = \langle T, \bullet \rangle$ is a group under the same operation, and T is a non empty subset of S, then H is a subgroup of G. The above definition implies that

1. If a and b are members of both groups, then $c = a \bullet b$ is also a member of both groups.
2. The groups share the same identity element.
3. If a is a member of both groups, the inverse of a is also a member of both groups.
4. The group made of the identity element of G, $H = \langle \{e\}, \bullet \rangle$, is a subgroup of G
5. Each group is a subgroup of itself

Cyclic Subgroups

If a subgroup of a group can be generated using the power of an element, the sub group is called the cyclic subgroup. The term power here means repeatedly applying the group operation to the element.

$$a^n \rightarrow a \cdot a \cdot a \ldots a \text{ (n times)}$$

The set made from this process is referred to as $\langle a \rangle$. The duplicate elements must be discarded. And also $a^0 = e$.

Example

Four cyclic subgroups can be made from the group $G = \langle Z_6, + \rangle$. They are $H_1 = \langle \{0\}, + \rangle$, $H_2 = \langle \{0,2,4\}, + \rangle$, $H_3 = \langle \{0,3\}, + \rangle$, & $H_4 = G$. Note that when the operation is addition, a^n means multiplying n by a. Note also that in all of these groups, the operation is addition modulo 6. The following show how we find the elements of these cyclic subgroups.

a. The cyclic subgroup generated from 0 is H_1, which has only one element the identity element

$$0^0 \bmod 6 = 0$$

b. The cyclic subgroup generated from 1 is H_4, which G is itself

$$1^0 \bmod 6 = 0$$
$$1^1 \bmod 6 = 1$$
$$1^2 \bmod 6 = (1+1) \bmod 6 = 2$$
$$1^3 \bmod 6 = (1+1+1) \bmod 6 = 3$$
$$1^4 \bmod 6 = (1+1+1+1) \bmod 6 = 4$$
$$1^5 \bmod 6 = (1+1+1+1+1) \bmod 6 = 5$$

c. The Cyclic subgroup generated from 2 is H which has three elements 0, 2 & 4

$2^0 \bmod 6 = 0$

$2^1 \bmod 6 = 2$

$2^2 \bmod 6 = (2+2) \bmod 6 = 4$

d. The Cyclic subgroup generated from 3 is H_3 has two elements 0 & 3.

$3^0 \bmod 6 = 0$

$3^1 \bmod 6 = 3.$

e. The Cyclic subgroup generated from 4 is H_2 is not a new subgroup

$4^0 \bmod 6 = 0$

$4^1 \bmod 6 = 4$

$4^2 \bmod 6 = (4+4) \bmod 6 = 2$

f. The Cyclic subgroup generated from 5 which is G itself

$5^0 \bmod 6 = 0$

$5^1 \bmod 6 = 5$

$5^2 \bmod 6 = 4$

$5^3 \bmod 6 = 3$

$5^4 \bmod 6 = 2$

$5^5 \bmod 6 = 1$

Cyclic Groups

A Cyclic group is a group that is its own Cyclic subgroup. The group G has a Cyclic subgroup $H_5 = G$. This means that the group G is a Cyclic group. In this case, the element that generates the Cyclic subgroup can also generate the group itself. This element is refered to as a generator. If g is a generator, the elements in a finite Cyclic group can be written as

$\{e, g, g^2, \ldots g^{n-1}\}$ where $g^n = e$

Ex The group $G = \langle Z_6, + \rangle$ is a Cyclic group with two generators $g=1$ & $g=5$

The group $G = \langle Z_{10}^*, \times \rangle$ is a Cyclic group with two generators $g=3$ & $g=7$.

Lagrange's Theorem

Lagrange's theorem relates the order of a group to the order of its subgroup. Assume that G is a group and H is a subgroup of G. If the order of G and H are $|G|$ and $|H|$ respectively, then based on this theorem, $|H|$ divides $|G|$. The order of the sub groups are $|H_1| = 1$, $|H_2| = 3$, $|H_3| = 2$ & $|H_4| = 6$. Obviously all of these orders divide 6.

Lagrange's theorem has a very interesting appl: Given a group G of order $|G|$, the orders of the potential subgroups can be easily determined if the divisors of $|G|$ can be found. For ex, the order of the group $G = \langle Z_{17}, + \rangle$ is 17. The only divisors of 17 are 1 and 17. This means that this group can have only two sub groups, H_1 with the identity element & $H_2 = G$.

Order of an Element

The order of an element a in a group, ord(a), is the smallest integer n such that $a^n = e$. The definition can be paraphrased the order of an element is the order of the cyclic group it generates.

ex In the group $G = \langle Z_6, + \rangle$, the orders of the element are ord(0) = 1, ord(1) = 6, ord(2) = 3, ord(3) = 2, ord(4) = 3, ord(5) = 6.

In the group $G = \langle Z_{10}^*, \times \rangle$, the orders of the elements are ord(1) = 1, ord(3) = 4, ord(7) = 4, ord(9) = 2.

Ring

A ring, denoted as $R = \langle \{..\}, \bullet, \square \rangle$, is an algebraic structure with two operations. The first operation must satisfy all five properties required for an abelian group. The second operation must satisfy only the first two. In addition, the second operation must be distributed over the first. Distributivity means that for all a, b and c elements of R, we have $a \square (b \bullet c) = (a \square b) \bullet (a \square c)$ & $(a \bullet b) \square c = (a \square c) \bullet (b \square c)$. A Commutative ring is a ring in which the Commutative property is also satisfied for the second the operation

Distribution of $\square$ over $\bullet$

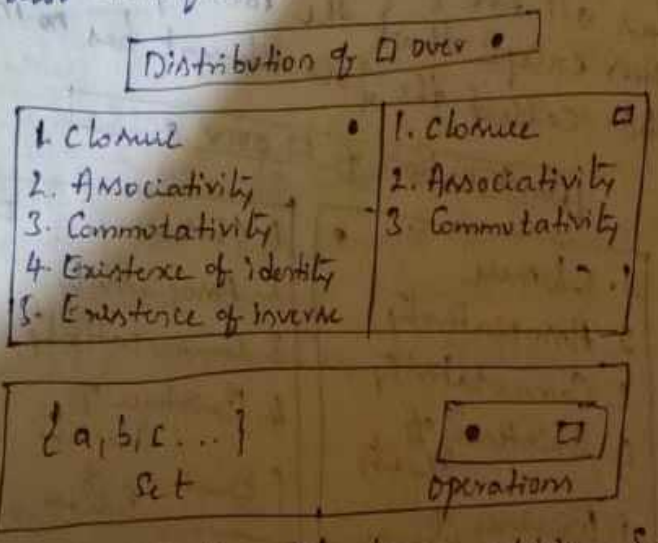
1. Closure $\bullet$	1. closure $\square$
2. Associativity	2. Associativity
3. Commutativity	3. Commutativity
4. Existence of identity	
5. Existence of inverse	

$\{a, b, c \ldots\}$ Set — $\bullet$ $\square$ operations

- The set Z with two operations, addition & multiplication is a Commutative group. we show it by $R = \langle Z, +, \times \rangle$. Addition satisfies all of the five properties, multiplication satisfies only three properties. Multiplication also distributes over addition. for example $5 \times (3+2) = (5 \times 3) + (5 \times 2) = 25$. Although we can perform addition & Substraction

Ring

A ring, denoted as $R = \langle \{..\}, \bullet, \square \rangle$, is an algebraic structure with two operations. The first operation must satisfy all five properties required for an abelian group. The second operation must satisfy only the first two. In addition, the second operation must be distributed over the first. Distributivity means that for all a, b and c elements of R, we have $a \square (b \bullet c) = (a \square b) \bullet (a \square c)$ & $(a \bullet b) \square c = (a \square c) \bullet (b \square c)$. A Commutative ring is a ring in which the Commutative property is also satisfied for the second the operation



- The set Z with two operations, addition & multiplication is a Commutative group. we show it by $R = \langle Z, +, \times \rangle$. Addition satisfies all of the five properties, multiplication satisfies only three properties. Multiplication also distributes over addition. for example $5 \times (3+2) = (5 \times 3) + (5 \times 2) = 25$. Although we can perform addition & substraction

Application

A field is a structure that supports two pairs of operations that we have used in mathematics: addition/subtraction & multiplication/division. There is one exception: division by zero is not allowed

Finite fields

We have fields of infinite order, only finite fields extensively used in Cryptography. A finite field, a field with a finite number of elements, are very important structures in Cryptography. Galois showed that for a field to be finite, the number of elements should be p^n, where p is a prime and n is a positive integer. The finite fields are usually called Galois fields and denoted as $GF(p^n)$.

Def: A Galois field $GF(p^n)$ is a finite field with p^n elements.

GF(p) fields

When n=1, we have GF(p) field. This field can be the set Z_p. $\{0, 1, \ldots p-1\}$, with two arithmetic operations (addition & and multiplication). Recall that in this set each element has an additive inverse and that nonzero elements have a multiplicative inverse (no multiplicative inverse for 0).

Ex A very common field in this category is GF(2) with the set $\{0, 1\}$ and two operations, addition & multiplication as shown

GF(2)

$\{0,1\}$	$+$ $\times$

+	0	1
0	0	1
1	1	0

addition

×	0	1
0	0	0
1	0	1

Multiplication

a	0	0	a	0	1
-a	1	1	a^{-1}	1	1

Inverses

There are several things to notice about this field. First, the set has only two elements, which are binary digits or bits (0 and 1). Second, the addition operation is actually the exclusive-or (XOR) operation we use on two binary digits. Third, the multiplication operation is the AND operation we use on two binary digits. Fourth, addition and subtraction operations are the same XOR operation. Fifth, multiplication and division operations are the same (AND operation).

GF(p^n) Fields

In addition to GF(p) fields, we are also interested in GF(p^n) fields in Cryptography. However the set Z, Z_n, Z_n^* and Z_p, which we have used so far with operations such as addition and multiplication, cannot satisfy the requirements of a field. Some new sets and some new operations on those sets must be defined.

GF(2^n) Fields

In Cryptography, we often need to use four operations (addition, subtraction, multiplication & division). In other words we need to use fields. However, when we work with computers, the positive integers are stored in the computer as n-bit words in which n is usually 8, 16, 32, 64, and so on. This means that the range of integers is 0 to 2^n-1. The modulus is 2^n, so we have two choices if we want to use a field.

1. We can use GF(p) with the set Z_p, where p is the largest prime number less than 2^n. Although this scheme works, it is inefficient because we cannot use the integers from p to 2^n-1.

for example, if $n=4$, the largest prime less than 2^4 is 13. This means that we cannot use integers 13, 14 & 15. If $n=8$, the largest prime less than 2^8 is 251, so we cannot use 251, 252, 253, 254 & 255.

2. We can work in $GF(2^n)$ & uses a set of 2^n elements. The elements in this set are n-bit words. for ex if $n=3$, the set is

{000, 001, 010, 011, 100, 101, 110, 111}

However, we cannot interpret each element as an integer b/w 0 to 7 because the regular four operations cannot be applied (the modulus 2^n is not a prime). We need to define a set of n-bit words & two new operations that satisfies the properties defined for a field.

Ex Let us define a $GF(2^n)$ field in which the set has four 2 bit words {00, 01, 10, 11} we can redefine addition & multiplication for this field in such a way that all properties of these operations are satisfied

$GF(2^2)$ field

Addition

⊕	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

Identity : 00

Multiplication

⊗	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

Identity : 01

Operations

Note that any operation on polynomials actually involves two operations on coefficients and operations on two polynomials. In other words, we need to define two fields: one for the coefficients and one for the polynomials. Coefficients are made of 0 & 1; we use the GF(2) field for the purpose.

Polynomials representing n-bit words use two fields GF(2) & GF(2^n).

Modulus:

Before defining the operations on polynomials, we need to talk about the modulus polynomials. Addition of two polynomials never creates a polynomial out of the set. Multiplication of two polynomials may create a polynomial with degrees more than n-1. This means we need to divide the result by a modulus & keep only the remainder.

For the sets of polynomials in GF(2^n), a group of polynomials of degree n is defined as the modulus. The modulus in this case acts as a prime polynomial, which means that no polynomials in the set can divide this polynomial. A polynomial prime cannot be factored into a polynomial with degree of less than n. Such polynomials are referred to as irreducible polynomials

Addition:

The addition operation for polynomials with coefficients in GF(2). Addition is very easy, we add the coefficients of the corresponding terms in GF(2). Note that adding two polynomials of degree n-1 always create a polynomial with degree n-1, which means that we do not need to reduce the result using the modulus.

for example:

$(x^5+x^2+x) \oplus (x^3+x^2+1)$. in $GF(2^8)$.

we use the symbol $\oplus$ to show that we mean polynomial addition.

$$0x^7+0x^6+1x^5+0x^4+0x^3+1x^2+1x^1+0x^0$$
$$0x^7+0x^6+0x^5+0x^4+1x^3+1x^2+0x^1+1x^0$$
$$0x^7+0x^6+1x^5+0x^4+1x^3+0x^2+1x^1+1x^0$$

$$\Rightarrow x^5+x^3+x+1$$

Multiplication

Multiplication in polynomials is the sum of the multiplication of each term of the first polynomial with each term of the second polynomial.

We need to consider three points

→ First the coefficient multiplication is done in GF(2).

→ Second multiplying x^i by x^j results in x^{i+j}

→ Third the multiplication may create terms with degree more than n-1, which means the results needs to be reduced using a modulus polynomials.

Example

$(x^5+x^2+x) \otimes (x^7+x^4+x^3+x^2+x)$ in $GF(2^8)$ with irreducible polynomial $(x^8+x^4+x^3+x+1)$

Sol $P_1 \otimes P_2 = x^5(x^7+x^4+x^3+x^2+x) +$
$x^2(x^7+x^4+x^3+x^2+x) +$
$x(x^7+x^4+x^3+x^2+x)$

$\Rightarrow x^{12}+x^9+x^8+x^7+x^6+x^9+x^6+x^5+x^4$
$+x^3+x^8+x^5+x^4+x^3+x^2$

Now a pair of terms with equal power of x are deleted. for example x^9+x^9 is totally deleted because the result is a zero polynomial.

$\Rightarrow x^{12}+x^7+x^2$

Now $(x^{12}+x^7+x^2) \bmod (x^8+x^4+x^3+x+1)$

$$x^8+x^4+x^3+x+1 \;\big|\; x^{12}+x^7+x^2 \quad (x^4+1)$$

$$x^{12}+x^8+x^7+x^5+x^4$$

$$x^8+x^5+x^4+x^2$$

$$x^8+x^4+x^3+x+1$$

Remainder $x^5+x^3+x^2+x+1$

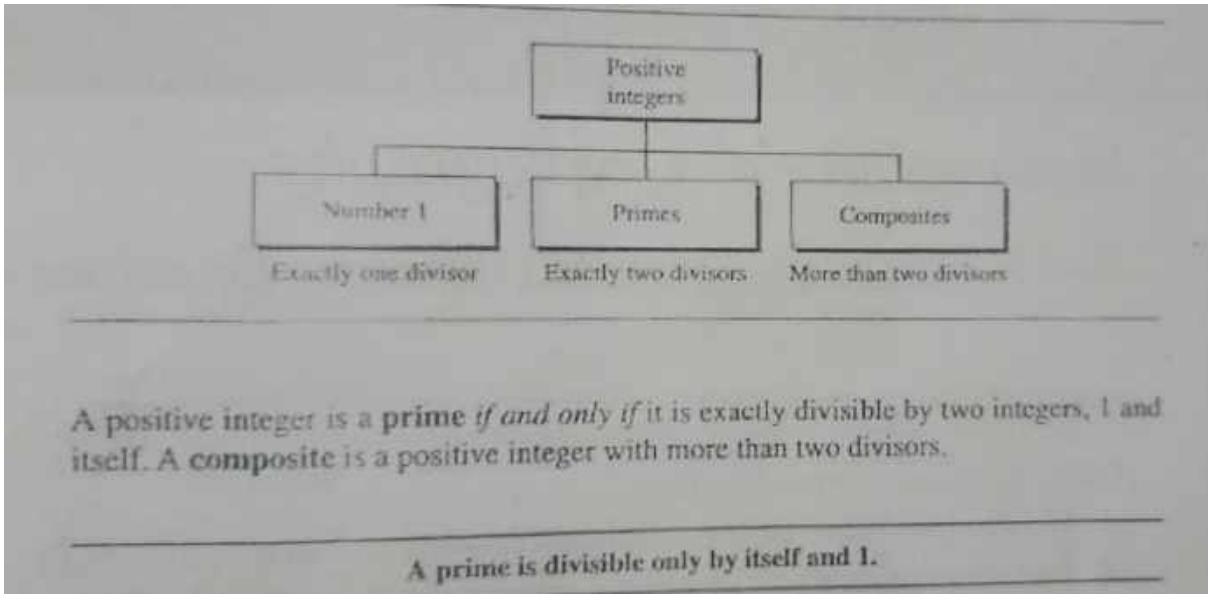
Multiplicative Identity

The multiplicative Identity is always 1. for example in $GF(2^8)$, the multiplicative Inverse is the bit pattern 00000001

PRIMES

The positive integers are classified into three categories:

1. **Number 1-----→ Exactly one Divisor**
2. **Prime -----→ Exactly Two Divisors**
3. **Composite ----→ More than two divisors**



A positive integer is a **prime** *if and only if* it is exactly divisible by two integers, 1 and itself. A **composite** is a positive integer with more than two divisors.

A prime is divisible only by itself and 1.

Example 9.1

What is the smallest prime?

Solution

The smallest prime is 2, which is divisible by 2 (itself) and 1. Note that the integer 1 is not a prime according to the definition, because a prime must be divisible by two different integers, no more, no less. The integer 1 is divisible only by itself, it is not a prime.

Example 9.2

List the primes smaller than 10.

Solution

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.

Coprimes

Two positive integers, a and b, are **relatively prime,** or **coprime,** if gcd $(a, b) = 1$. Note that the number 1 is relatively prime with any integer. If p is a prime, then all integers 1 to $p - 1$ are relatively prime to p. In Chapter 2, we discussed set Z_n^* whose members are all relatively prime to n. Set Z_p^* is the same except that modulus (p) is a prime.

For Example :

Find whether the 10 and 9 relatively prime or not

Check whether 10 and 9 is relatively prime

Solution:

Factors of 10 are 1, 2, 5, 10.

Factors of 9 are 1, 3, 9.

GCD (10 , 9) = 1

So 10 and 9 are relatively prime.

Cardinality of Primes

Cardinality of primes is nothing but to check is there a finite number of primes or the list infinite.

Infinite Number of Primes

The number of primes is infinite. Here is an informal proof: Suppose that the set of primes is finite (limited), with p as the largest prime. Multiply the set of primes and call the result $P = 2 \times 3 \times \cdots \times p$. The integer $(P + 1)$ cannot have a factor $q \leq p$. We know that q divides P. If q also divides $(P + 1)$, then q divides $(P + 1) - P = 1$. The only number that divides 1 is 1, which is not a prime. Therefore, q is larger than p.

There is an infinite number of primes.

Example 9.3

As a trivial example, assume that the only primes are in the set {2, 3, 5, 7, 11, 13, 17}. Here $P = 510510$ and $P + 1 = 510511$. However, $510511 = 19 \times 97 \times 277$; none of these primes were in the original list. Therefore, there are three primes greater than 17.

Number of Primes

To answer the second question, a function called $\pi(n)$ is defined that finds the number of primes smaller than or equal to n. The following shows the values of this function for different n's.

$$\pi(1) = 0 \quad \pi(2) = 1 \quad \pi(3) = 2 \quad \pi(10) = 4 \quad \pi(20) = 8 \quad \pi(50) = 15 \quad \pi(100) = 25$$

But if n is very large, how can we calculate $\pi(n)$? The answer is that we can only use approximation. It has been shown that

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

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

Checking for Primeness

The next question that comes to mind is this: Given a number n, how can we determine if n is a prime? The answer is that we need to see if the number is divisible by all primes less than $\sqrt{n}$. We know that this method is inefficient, but it is a good start.

Example 9.5

Is 97 a prime?

Solution

The floor of $\sqrt{97} = 9$. The primes less than 9 are 2, 3, 5, and 7. We need to see if 97 is divisible by any of these numbers. It is not, so 97 is a prime.

Sieve of Eratosthenes

The Greek mathematician Eratosthenes devised a method to find all primes less than n. The method is called the **sieve of Eratosthenes**. Suppose we want to find all prime less than 100. We write down all the numbers between 2 and 100. Because $\sqrt{100} = 10$, we need to see if any number less than 100 is divisible by 2, 3, 5, and 7. Table 9.1 shows the result.

Table 9.1 *Sieve of Eratosthenes*

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

The following shows the process:

1. Cross out all numbers divisible by 2 (except 2 itself).
2. Cross out all numbers divisible by 3 (except 3 itself).
3. Cross out all numbers divisible by 5 (except 5 itself).
4. Cross out all numbers divisible by 7 (except 7 itself).
5. The numbers left over are primes.

Euler's Phi-Function

Euler's phi-function, $\phi(n)$, 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. Recall from Chapter 2 that the set Z_n^* contains the numbers that are smaller than n and relatively prime to n. The function $\phi(n)$ calculates the number of elements in this set. The following helps to find the value of $\phi(n)$.

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

We can combine the above four rules to find the value of $\phi(n)$. For example, if n can be factored as $n = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_k^{e_k}$, then we combine the third and the fourth rule to find

$$\phi(n) = (p_1^{e_1} - p_1^{e_1-1}) \times (p_2^{e_2} - p_2^{e_2-1}) \times \cdots \times (p_k^{e_k} - p_k^{e_k-1})$$

It is very important to notice that the value of $\phi(n)$ for large composites can be found only if the number n can be factored into primes. In other words, the difficulty of finding $\phi(n)$ depends on the difficulty of finding the factorization of n, which is discussed in the next section.

The difficulty of finding $\phi(n)$ depends on the difficulty of finding the factorization of n.

Example 9.7

What is the value of $\phi(13)$?

Solution
Because 13 is a prime, $\phi(13) = (13 - 1) = 12$.

Example 9.8

What is the value of $\phi(10)$?

Solution
We can use the third rule: $\phi(10) = \phi(2) \times \phi(5) = 1 \times 4 = 4$, because 2 and 5 are primes.

Example 9.9

What is the value of $\phi(240)$?

Solution
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 9.10

Can we say that $\phi(49) = \phi(7) \times \phi(7) = 6 \times 6 = 36$?

Solution
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 9.11

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

Solution
The answer is $\phi(14) = \phi(7) \times \phi(2) = 6 \times 1 = 6$. The members are 1, 3, 5, 9, 11, and 13.

Interesting point: If $n > 2$, the value of $\phi(n)$ is even.

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 $a^{p-1} = 1 \bmod p$.

Second Version

The second version removes the condition on a. It says that if p is a prime and a is an integer, then $a^p = a \bmod p$.

Applications

Although we will see some applications of this theorem later in this chapter, the theorem is very useful for solving some problems.

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

Example 9.12

Find the result of $6^{10} \bmod 11$.

Solution

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

Example 9.13

Find the result of $3^{12} \bmod 11$.

Solution

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

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

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 $a^{-1} \bmod p = a^{p-2} \bmod p$.

This can be easily proved if we multiply both sides of the equality by a and use the first version of Fermat's little theorem:

$$a \times a^{-1} \bmod p = a \times a^{p-2} \bmod p = a^{p-1} \bmod p = 1 \bmod p$$

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

Example 9.14

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

9.2 PRIMALITY TESTING

If schemes for generating primes, like Fermat's or Mersenne's, have failed to produce large primes, how can we create large primes for cryptography? We could just choose a large random number and test it to be sure that it is a prime.

Finding an algorithm to correctly and efficiently test a very large integer and output *a prime* or *a composite* has always been a challenge in number theory, and consequently in cryptography. However, recent developments (one of which we discuss in this section) look very promising.

Algorithms that deal with this issue can be divided into two broad categories: **deterministic algorithms** and **probabilistic algorithms.** Some members of both categories are discussed here. A deterministic algorithm always gives a correct answer; a probabilistic algorithm gives an answer that is correct most of the time, but not all of the time. Although a deterministic algorithm is ideal, it is normally less efficient than the corresponding probabilistic one.

Deterministic Algorithms

A deterministic primality testing algorithm accepts an integer and always outputs *a prime* or *a composite*. Until recently, all deterministic algorithms were so inefficient at finding larger primes that they were considered infeasible. As we will show shortly, a newer algorithm looks more promising.

Divisibility Algorithm

The most elementary deterministic test for primality is the **divisibility test.** We use as divisors all numbers smaller that $\sqrt{n}$. If any of these numbers divides n, then n is composite. Algorithm 9.1 shows the divisibility test in its primitive, very inefficient form.

The algorithm can be improved by testing only odd numbers. It can be further improved by using a table of primes between 2 and $\sqrt{n}$. The number of arithmatic operations in Algorithm 9.1 is $\sqrt{n}$. If we assume that each arithmatic operation uses only one bit operation (unrealistic) then the bit-operation complexity of Algorithm 9.1 is $f(n_b) = \sqrt{2^{n_b}} = 2^{n_b/2}$, where n_b is the number of bits in n. In Big-O notation, the complexity can be shown as $O(2^{n_b})$: *exponential* (see Appendix L). In other words, the divisibility algorithm is infeasible (intractable) if n_b is large.

Deterministic Algorithms

A deterministic primality testing algorithm accepts an integer and always outputs *a prime* or *a composite*. Until recently, all deterministic algorithms were so inefficient at finding larger primes that they were considered infeasible. As we will show shortly, a newer algorithm looks more promising.

Divisibility Algorithm

The most elementary deterministic test for primality is the **divisibility test.** We use as divisors all numbers smaller that $\sqrt{n}$. If any of these numbers divides n, then n is composite. Algorithm 9.1 shows the divisibility test in its primitive, very inefficient form.

The algorithm can be improved by testing only odd numbers. It can be further improved by using a table of primes between 2 and $\sqrt{n}$. The number of arithmatic operations in Algorithm 9.1 is $\sqrt{n}$. If we assume that each arithmatic operation uses only one bit operation (unrealistic) then the bit-operation complexity of Algorithm 9.1 is $f(n_b) = \sqrt{2^{n_b}} = 2^{n_b/2}$, where n_b is the number of bits in n. In Big-O notation, the complexity can be shown as $O(2^{n_b})$: *exponential* (see Appendix L). In other words, the divisibility algorithm is infeasible (intractable) if n_b is large.

Example 9.18

Assume n has 200 bits. What is the number of bit operations needed to run the divisibility-test algorithm?

Solution

The bit-operation complexity of this algorithm is $2^{n_b/2}$. This means that the algorithm needs 2^{100} bit operations. On a computer capable of doing 2^{30} bit operations per second, the algorithm needs 2^{70} seconds to do the testing (forever).

Algorithm 9.1 *Pseudocode for the divisibility test*

```
Divisibility_Test (n)                    // n is the number to test for primality
{
  r ← 2
  while (r < √n )
  {
   if (r | n) return "a composite"
   r ← r + 1
  }
  return "a prime"
}
```

AKS Algorithm

In 2002, Agrawal, Kayal, and Saxena announced that they had found an algorithm for primality testing with polynomial bit-operation time complexity of $O((\log_2 n_b)^{12})$. The algorithm uses the fact that $(x - a)^p \equiv (x^p - a) \bmod p$. It is not surprising to see some future refinements make this algorithm the standard primality test in mathematics and computer science.

Example 9.19

Assume n has 200 bits. What is the number of bit operations needed to run the AKS algorithm?

Solution

The bit-operation complexity of this algorithm is $O((\log_2 n_b)^{12})$. This means that the algorithm needs only $(\log_2 200)^{12} = 39{,}547{,}615{,}483$ bit operations. On a computer capable of doing 1 billion bit operations per second, the algorithm needs only 40 seconds.

Probabilistic Algorithms

Before the AKS algorithm, all efficient methods for primality testing have been probabilistic. These methods may be used for a while until the AKS is formally accepted as the standard. A probabilistic algorithm does not guarantee the correctness of the result. However, we can make the probability of error so small that it is almost certain that the algorithm has returned a correct answer. The bit-operation complexity of the algorithm can become polynomial if we allow a small chance for mistakes. A probabilistic algorithm in this category returns either *a prime* or *a composite* based on the following rules:

a. If the integer to be tested is actually a prime, the algorithm definitely returns *a prime*.

b. If the integer to be tested is actually a composite, it returns *a composite* with probability $1 - \varepsilon$, but it may return *a prime* with the probability ε.

The probability of mistake can be improved if we run the algorithm more than once with different parameters or using different methods. If we run the algorithm m times, the probability of error may reduce to ε^m.

Fermat Test

The first probabilistic method we discuss is the **Fermat primality test.** Recall the *Fermat little theorem*

If n is a prime, then $a^{n-1} \equiv 1 \bmod n$.

Note that this means that if n is a prime, the congruence holds. It does not mean that if the congruence holds, n is a prime. The integer can be a prime or composite. We can define the following as the Fermat test

If n is a prime, $a^n - 1 \equiv 1 \bmod n$
If n is a composite, it is possible that $a^n - 1 \equiv 1 \bmod n$

A prime passes the Fermat test; a composite may pass the Fermat test with probability ε. The bit-operation complexity of Fermat test is the same as the complexity of an algorithm that calculates exponentiation. Later in this chapter, we introduce an algorithm for fast exponentiation with bit-operation complexity of $O(n_b)$, where n_b is the number of bits in n. The probability can be improved by testing with several bases (a_1, a_2, a_3, and so on). Each test increases the probability that the number is a prime.

Example 9.20

Does the number 561 pass the Fermat test?

Solution
Use base 2

$$2^{561-1} = 1 \bmod 561$$

The number passes the Fermat test, but it is not a prime, because $561 = 33 \times 17$.

Square Root Test

In modular arithmetic, if n is a prime, the square root of 1 is either +1 or −1. If n is composite, the square root is +1 or −1, but there may be other roots. This is known as the **square root primality test.** Note that in modular arithmetic, −1 means $n-1$.

If n is a prime, $\sqrt{1} \bmod n = \pm 1$.
If n is a composite, $\sqrt{1} \bmod n = \pm 1$ and possibly other values.

Example 9.21

What are the square roots of 1 mod n if n is 7 (a prime)?

Solution
The only square roots are 1 and −1. We can see that

$1^2 = 1 \bmod 7$	$(-1)^2 = 1 \bmod 7$
$2^2 = 4 \bmod 7$	$(-2)^2 = 4 \bmod 7$
$3^2 = 2 \bmod 7$	$(-3)^2 = 2 \bmod 7$

Example 9.22

What are the square roots of 1 mod n if n is 8 (a composite)?

Solution

There are four solutions: 1, 3, 5, and 7 (which is −1). We can see that

$1^2 = 1 \bmod 8$ $\quad$ $(-1)^2 = 1 \bmod 8$
$3^2 = 1 \bmod 8$ $\quad$ $5^2 = 1 \bmod 8$

Example 9.23

What are the square roots of 1 mod n if n is 17 (a prime)?

Solution

There are only two solutions: 1 and −1

$1^2 = 1 \bmod 17$ $\quad$ $(-1)^2 = 1 \bmod 17$
$2^2 = 4 \bmod 17$ $\quad$ $(-2)^2 = 4 \bmod 17$
$3^2 = 9 \bmod 17$ $\quad$ $(-3)^2 = 9 \bmod 17$
$4^2 = 16 \bmod 17$ $\quad$ $(-4)^2 = 16 \bmod 17$
$5^2 = 8 \bmod 17$ $\quad$ $(-5)^2 = 8 \bmod 17$
$6^2 = 2 \bmod 17$ $\quad$ $(-6)^2 = 2 \bmod 17$
$(7)^2 = 15 \bmod 17$ $\quad$ $(-7)^2 = 15 \bmod 17$
$(8)^2 = 13 \bmod 17$ $\quad$ $(-8)^2 = 13 \bmod 17$

Note that there is no need to check integers larger than 8 because $9 = -8 \bmod 17$, and so on.

Miller-Rabin Test

The **Miller-Rabin primality test** combines the *Fermat test* and the *square root test* in a very elegant way to find a **strong pseudoprime** (a prime with a very high probability). In this test, we write $n - 1$ as the product of an odd number m and a power of 2:

$$n - 1 = m \times 2^k$$

The Fermat test in base a can be written as shown in Figure 9.2.

$$a^{n-1} = a^{m \times 2^k} = [a^m]^{2^k} = [a^m]^{2 \times 2 \times \cdots \times 2}$$

In other words, instead of calculating $a^{n-1} \pmod n$ in one step, we can do it in $k + 1$ steps. What is the benefit of using $k + 1$ steps instead of just one? The benefit is that, in each step, the square root test can be performed. If the square root test fails, we stop and declare n a composite number. In each step, we assure ourself that the Fermat test is passed and the square root test is satisfied between all pairs of adjacent steps, if applicable (if the result is 1).

Initialization:

Choose a base a and calculate $T = a^m$, in which $m = (n - 1) / 2^k$

a. If T is +1 or −1, declare that n is a strong pseudoprime and stop. We say that n has passed two tests, the Fermat test and the square root test. Why? Because if T is ±1, T will become 1 in the next step and remains 1 until it passes the *Fermat test*. In addition T has passed the *square root test*, because T would be 1 in the next step and the square root of 1 (in the next step) is ±1 (in this step).

... are not sure if n is a prime or a composite, so we con-

Step 1:

We square T.

a. If the result is +1, we definitely know that the Fermat test will be passed, because T remains 1 for the succeeding tests. The square root test, however, has not been passed. Because T is 1 in this step and was something other than ±1 in the previous step (the reason why we did not stop in the previous step), we declare n a composite and stop.

b. If the result is −1, we know that n will eventually pass the Fermat test. We also know that it will pass the square root test because T is −1 in this step and becomes 1 in the next step. We declare n a strong pseudoprime and stop.

c. If T is anything else, we are not sure whether we do or do not have a prime. We continue to the next step.

Steps 2 to $k - 1$:

This step and all steps until step $k - 1$ are the same as step 1.

Step k:

This step is not needed. If we have reached this step and we have not made a decision, this step will not help us. If the result of this step is 1, the Fermat test is passed, but because the result of the previous step is not ±1, the square root test is not passed. After step $k - 1$, if we have not already stopped, we declare that n is composite.

The Miller-Rabin test needs from step 0 to step $k - 1$.

Algorithm 9.2 shows the pseudocode for the Miller-Rabin test.

Algorithm 9.2 *Pseudocode for Miller-Rabin test*

```
Miller_Rabin_Test (n, a)                    // n is the number; a is the base.
{
    Find m and k such that n − 1 = m × 2^k
    T ← a^m mod n
    if (T = ± 1) return "a prime"
    for (i ← 1 to k − 1)                    // k − 1 is the maximum number of steps.
    {
        T ← T^2 mod n
        if (T = +1) return "a composite"
        if (T = −1) return "a prime"
    }
    return "a composite"
}
```

Example 9.25

Does the number 561 pass the Miller-Rabin test?

Solution

Using base 2, let $561 - 1 = 35 \times 2^4$, which means $m = 35$, $k = 4$, and $a = 2$

Initialization:	$T = 2^{35} \bmod 561 = 263 \bmod 561$	
$k = 1$:	$T = 263^2 \bmod 561 = 166 \bmod 561$	
$k = 2$:	$T = 166^2 \bmod 561 = 67 \bmod 561$	
$k = 3$:	$T = 67^2 \bmod 561 = +1 \bmod 561$	→ **a composite**

Recommended Primality Test

Today, one of the most popular primality test is a combination of the divisibility test and the Miller-Rabin test. Following are the recommended steps:

1. Choose an odd integer, because all even integers (except 2) are definitely composites.
2. Do some trivial divisibility tests on some known primes such as 3, 5, 7, 11, 13, and so on to be sure that you are not dealing with an obvious composite. If the number passes all of these tests, move to the next step. If the number fails any of these tests, go back to step 1 and choose another odd number.
3. Choose a set of bases for testing. A large set of bases is preferable.
4. Do Miller-Rabin tests on each of the bases. If any of them fails, go back to step 1 and choose another odd number. If the test passes for all bases, declare the number a strong pseudoprime.

Example 9.28

The number 4033 is a composite (37 × 109). Does it pass the recommended primality test?

Solution

1. Perform the divisibility tests first. The numbers 2, 3, 5, 7, 11, 17, and 23 are not divisors of 4033.
2. Perform the Miller-Rabin test with a base of 2, $4033 - 1 = 63 \times 2^6$, which means m is 63 and k is 6.

Initialization: $T \equiv 2^{63} \pmod{4033} \equiv 3521 \pmod{4033}$

$k = 1$ $\quad T \equiv T^2 \equiv 3521^2 \pmod{4033} \equiv -1 \pmod{4033}$ → **Passes**

3. But we are not satisfied. We continue with another base, 3.

Initialization: $T \equiv 3^{63} \pmod{4033} \equiv 3551 \pmod{4033}$

$k = 1$ $\quad T \equiv T^2 \equiv 3551^2 \pmod{4033} \equiv 2443 \pmod{4033}$

$k = 2$ $\quad T \equiv T^2 \equiv 2443^2 \pmod{4033} \equiv 3442 \pmod{4033}$

$k = 3$ $\quad T \equiv T^2 \equiv 3442^2 \pmod{4033} \equiv 2443 \pmod{4033}$

$k = 4$ $\quad T \equiv T^2 \equiv 2443^2 \pmod{4033} \equiv 3442 \pmod{4033}$

$k = 5$ $\quad T \equiv T^2 \equiv 3442^2 \pmod{4033} \equiv 2443 \pmod{4033}$ → Failed (composite)

9.3 FACTORIZATION

Factorization has been the subject of continuous research in the past; such research is likely to continue in the future. Factorization plays a very important role in the security of several public-key cryptosystems (see Chapter 10).

Fundamental Theorem of Arithmetic

According to the *Fundamental Theorem of Arithmetic*, any positive integer greater than one can be written uniquely in the following prime **factorization** form where $p_1, p_2, \ldots, p_k$ are primes and $e_1, e_2, \ldots, e_k$ are positive integers.

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

There are immediate applications of factorization, such as the calculation of the greatest common divisor and the least common multiplier.

Greatest Common Divisor

Chapter 2 discussed the greatest common divisor of two numbers, gcd (a, b). Recall that the Euclidean algorithm gives this value, but this value can also be found if we know the factorization of a and b.

$$a = p_1^{a1} \times p_2^{a2} \times \cdots \times p_k^{ak} \qquad b = p_1^{b1} \times p_2^{b2} \times \cdots \times p_k^{bk}$$

$$\gcd(a, b) = p_1^{\min(a_1, b_1)} \times p_2^{\min(a_2, b_2)} \times \cdots \times p_k^{\min(a_k, b_k)}$$

Least Common Multiplier

The *least common multiplier*, lcm (a, b), is the smallest integer that is a multiple of both a and b. Using factorization, we also find lcm (a, b).

$$a = p_1^{a1} \times p_2^{a2} \times \cdots \times p_k^{ak} \qquad b = p_1^{b1} \times p_2^{b2} \times \cdots \times p_k^{bk}$$

$$\text{lcm}(a, b) = p_1^{\max(a_1, b_1)} \times p_2^{\max(a_2, b_2)} \times \cdots \times p_k^{\max(a_k, b_k)}$$

It can be proved that gcd (a, b) and lcm (a, b) are related to each other as shown below:

$$\text{lcm}(a, b) \times \text{gcd}(a, b) = a \times b$$

Factorization Methods

There has been a long search for efficient algorithms to factor large composite numbers. Unfortunately, no such perfect algorithm has been found. Although there are several algorithms that can factor a number, none are capable of factoring a very large number in a reasonable amount of time. Later we will see that this is good for cryptography because modern cryptosystems rely on this fact. 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.

Trial Division Method

By far, the simplest and least efficient algorithm is the **trial division factorization method.** We simply try all the positive integers, starting with 2, to find one that divides n. From discussion on the *sieve of Eratosthenes*, we know that if n is composite, then it will have a prime $p \leq \sqrt{n}$. Algorithm 9.3 shows the pseudocode for this method. The algorithm has two loops, one outer and one inner. The outer loop finds unique factors; the inner loop finds duplicates of a factor. For example, $24 = 2^3 \times 3$. The outer loop finds the factors 2 and 3. The inner loop finds that 2 is a multiple factor.

Algorithm 9.3 *Pseudocode for trial-division factorization*

```
Trial_Division_Factorization (n)          // n is the number to be factored
{
   a ← 2
   while (a ≤ √n)
   {
      while (n mod a = 0)
      {
         output a                         // Factors are output one by one
         n = n / a
      }
      a ← a + 1
   }
   if (n > 1) output n                    // n has no more factors
}
```

Complexity The trial-division method is normally good if $n < 2^{10}$, but it is very inefficient and infeasible for factoring large integers. The complexity of the algorithm (see Appendix L) is *exponential*.

Example 9.29

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

Solution

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

$$1233 = 3^2 \times 137$$

Example 9.30

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

Solution

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

$$1523357784 = 2^3 \times 3^2 \times 13 \times 37 \times 43987$$

9.4 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:

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

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

Example 9.35

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

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

The solution to this set of equations is given in the next section; for the moment, 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

The solution to the set of equations follows these steps:

1. Find $M = m_1 \times m_2 \times \cdots \times m_k$. This is the 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 $M_1, M_2, \ldots, M_k$ using the corresponding moduli ($m_1, m_2, \ldots, m_k$). Call the inverses $M_1^{-1}, M_2^{-1}, \ldots, M_k^{-1}$.
4. The solution to the simultaneous equations is

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

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.

Find the solution to the simultaneous equations:

$$x \equiv 2 \bmod 3$$
$$x \equiv 3 \bmod 5$$
$$x \equiv 2 \bmod 7$$

Solution

From the previous example, we already know that the answer is $x = 23$. We follow the four steps.

1. $M = 3 \times 5 \times 7 = 105$
2. $M_1 = 105 / 3 = 35$, $M_2 = 105 / 5 = 21$, $M_3 = 105 / 7 = 15$
3. The inverses are $M_1^{-1} = 2$, $M_2^{-1} = 1$, $M_3^{-1} = 1$
4. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 = 23 \bmod 105$

9.5 QUADRATIC CONGRUENCE

Linear congruence was discussed in Chapter 2 and the Chinese remainder theorem was discussed in the previous section. In cryptography, we also need to discuss **quadratic congruence**—that is, equations of the form $a_2x^2 + a_1x + a_0 \equiv 0 \pmod{n}$. We limit our discussion to quadratic equations in which $a_2 = 1$ and $a_1 = 0$, that is equations of the form

$$x^2 \equiv a \pmod{n}.$$

Quadratic Congruence Modulo a Prime

We first consider the case in which the modulus is a prime. In other words, we want to find the solutions for an equation of the form $x^2 \equiv a \pmod{p}$, in which p is a prime, a is an integer such that $p \nmid a$. It can be proved that this type of equation has either no solution or exactly two incongruent solutions.

Example 9.39

The equation $x^2 \equiv 3 \pmod{11}$ has two solutions, $x \equiv 5 \pmod{11}$ and $x \equiv -5 \pmod{11}$. But note that $-5 \equiv 6 \pmod{11}$, so the solutions are actually 5 and 6. Also note that these two solutions are incongruent.

Example 9.40

The equation $x^2 \equiv 2 \pmod{11}$ has no solution. No integer x can be found such that its square is 2 mod 11.

Quadratic Residues and Nonresidue

In the equation $x^2 \equiv a \pmod{p}$, a is called a **quadratic residue (QR)** if the equation has two solutions; a is called **quadratic nonresidue (QNR)** if the equation has no solutions. It can be proved that in $\mathbf{Z}_p^*$, with $p - 1$ elements, exactly $(p - 1)/2$ elements are quadratic residues and $(p - 1)/2$ are quadratic nonresidues.

Example 9.41

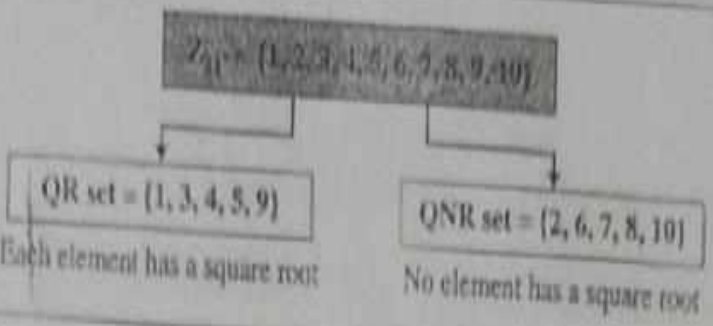
There are 10 elements in $\mathbf{Z}_{11}^*$. Exactly five of them are quadratic residues and five of them are nonresidues. In other words, $\mathbf{Z}_{11}^*$ is divided into two separate sets, QR and QNR, as shown in Figure 9.4.

Euler's Criterion

How can we check to see if an integer is a QR modulo p? Euler's criterion gives a very specific condition:

a. If $a^{(p-1)/2} \equiv 1 \pmod{p}$, a is a quadratic residue modulo p.
b. If $a^{(p-1)/2} \equiv -1 \pmod{p}$, a is a quadratic nonresidue modulo p.

Figure 9.4 *Division of Z_{11}^* elements into QRs and QNRs*



Example 9.42

To find out if 14 or 16 is a QR in Z_{23}^*, we calculate:

$$14^{(23-1)/2} \bmod 23 \rightarrow 14^{11} \bmod 23 \rightarrow 22 \bmod 23 \rightarrow -1 \bmod 23 \quad \text{nonresidue}$$

$$15^{(23-1)/2} \bmod 23 \rightarrow 16^{11} \bmod 23 \rightarrow 1 \bmod 23 \quad \text{residue}$$

Solving Quadratic Equation Modulo a Prime

Although the Euler criterion tells us if an integer a is a QR or QNR in Z_p^*, it cannot find the solution to $x^2 \equiv a \pmod{p}$. To find the solution to this quadratic equation, we notice that a prime can be either $p = 4k + 1$ or $p = 4k + 3$, in which k is a positive integer. The solution to a quadratic equation is very involved in the first case; it is easier in the second. We will discuss only the second case, which we will use in Chapter 10 when we discuss Rabin cryptosystem.

Special Case: $p = 4k + 3$ If p is in the form $4k + 3$ (that is, $p \equiv 3 \bmod 4$) and a is a QR in Z_p^*, then

$$x \equiv a^{(p+1)/4} \pmod{p} \quad \text{and} \quad x \equiv -a^{(p+1)/4} \pmod{p}$$

Solve the following quadratic equations:

a. $x^2 \equiv 3 \pmod{23}$
b. $x^2 \equiv 2 \pmod{11}$
c. $x^2 \equiv 7 \pmod{19}$

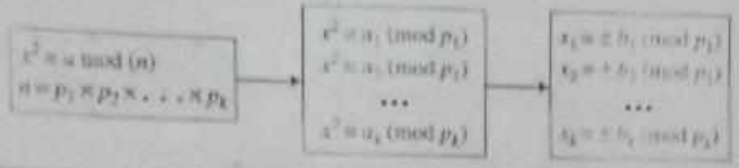
Solutions

a. In the first equation, 3 is a QR in $\mathbf{Z}_{23}$. The solution is $x \equiv \pm 16 \pmod{23}$. In other words, $\sqrt{3} \equiv \pm 16 \pmod{23}$.
b. In the second equation, 2 is a QNR in $\mathbf{Z}_{11}$. There is no solution for $\sqrt{2}$ in $\mathbf{Z}_{11}$.
c. In the third equation, 7 is a QR in $\mathbf{Z}_{19}$. The solution is $x \equiv \pm 11 \pmod{19}$. In other words, $\sqrt{7} \equiv \pm 11 \pmod{19}$.

Quadratic Congruence Modulo a Composite

Quadratic congruence modulo a composite can be done by solving a set of congruence modulo a prime. In other words, we can decompose $x^2 \equiv a \pmod{n}$ if we have the factorization of n. Now we can solve each decomposed equation (if solvable) and find k pairs of answers for x as shown in Figure 9.5.

Figure 9.5 *Decomposition of congruence modulo a composite*



From k pairs of answers, we can make 2^k set of equations that can be solved using the Chinese remainder theorem to find 2^k values for x. In cryptography, normally n is made such that $n = p \times q$, which means $k = 2$ and we have only four total answers.

Example 9.44

Assume that $x^2 \equiv 36 \pmod{77}$. We know that $77 = 7 \times 11$. We can write

$$x^2 \equiv 36 \pmod{7} \equiv 1 \pmod{7} \qquad \text{and} \qquad x^2 \equiv 36 \pmod{11} \equiv 3 \pmod{11}$$

Note that we have chosen 3 and 7 to be of the form $4k + 3$ so that we can solve the equatic based on the previous discussion. Both of these equations have quadratic residues in their o sets. The answers are $x \equiv +1 \pmod{7}$, $x \equiv -1 \pmod{7}$, $x \equiv +5 \pmod{11}$, and $x \equiv -5 \pmod{1}$ Now we can make four sets of equations out of these:

Set 1: $x \equiv +1 \pmod{7}$ $\quad x \equiv +5 \pmod{11}$
Set 2: $x \equiv +1 \pmod{7}$ $\quad x \equiv -5 \pmod{11}$
Set 3: $x \equiv -1 \pmod{7}$ $\quad x \equiv +5 \pmod{11}$
Set 4: $x \equiv -1 \pmod{7}$ $\quad x \equiv -5 \pmod{11}$

The answers are $x = \pm 6$ and ± 27.

9.6 EXPONENTIATION AND LOGARITHM

Exponentiation and logarithm are inverses of each other. The following shows the relationship between them, in which a is called the base of the exponentiation or logarithm.

Exponentiation: $y = a^x$ → **Logarithm:** $x = \log_a y$

Exponentiation

In cryptography, a common modular operation is **exponentiation.** That is, we often need to calculate

$$y = a^x \bmod n$$

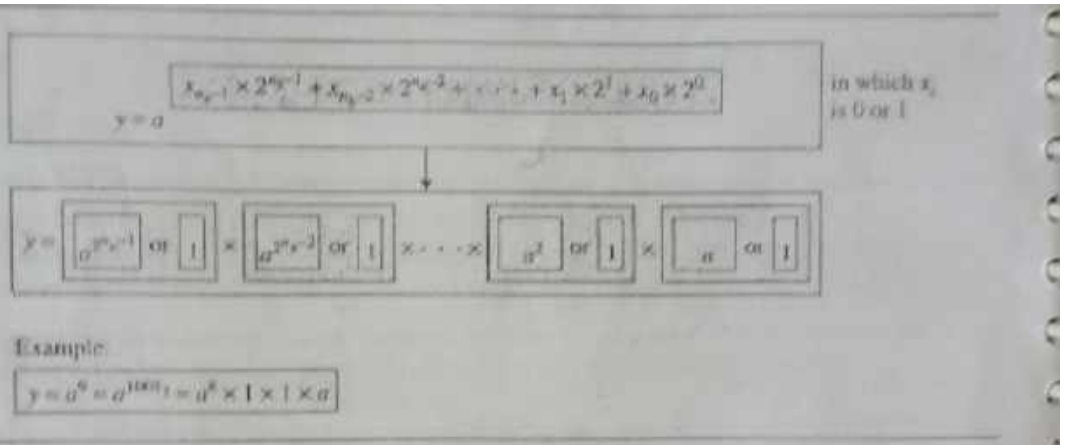
The RSA cryptosystem, which will be discussed in Chapter 10, uses exponentiation for both encryption and decryption with very large exponents. Unfortunately, most computer languages have no operator that can efficiently compute exponentiation, particularly when the exponent is very large. To make this type of calculation more efficient, we need algorithms that are more efficient.

Fast Exponentiation

Fast exponentiation is possible using the **square-and-multiply method.** In traditional algorithms only *multiplication* is used to simulate exponentiation, but the fast exponentiation algorithm uses both *squaring* and *multiplication*. The main idea behind this method is to treat the exponent as a binary number of n_b bits (x_0 to x_{n_b-1}). For example, $x = 22 = (10110)_2$. In general, x can be written as:

$$x = x_{n_b-1} \times 2^{k-1} + x_{n_b-2} \times 2^{k-2} + \cdots + x_2 \times 2^2 + x_1 \times 2^1 + x_0 \times 2^0$$

Now we can write $y = a^x$ as shown in Figure 9.6.



Note that y is the product of n_b terms. Each term is either 1 (if the corresponding bit is 0) or a^{2^i} (if the corresponding bit is 1). In other words, the term a^{2^i} is included in the multiplication if the bit is 1, it is not included if the bit is 0 (multiplication by 1 has no effect). Figure 9.6 gives the general idea how to write the algorithm. We can continuously square the base, $a, a^2, a^4, \ldots, a^{2^{n_b-1}}$. If the corresponding bit is 0, the term is not included in the multiplication process; if the bit is 1, it is. Algorithm 9.7 reflects these two observations.

Algorithm 9.7 *Pseudocode for square-and-multiply algorithm*

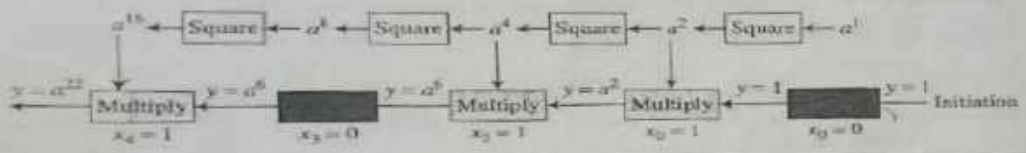
```
Square_and_Multiply (a, x, n)
{
    y ← 1
    for (i ← 0 to n_b − 1)                    // n_b is the number of bits in x
    {
        if (x_i = 1)   y ← a × y  mod n       // multiply only if the bit is 1
        a ← a^2 mod n                         // squaring is not needed in the last iteration
    }
    return y
}
```

Algorithm 9.7 uses n_b iterations. In each iteration, it checks the value of the corresponding bit. If the value of the bit is 1, it multiplies the current base with the previous value of the result. It then squares the base for the next iteration. Note that squaring is not needed in the last step (the result is not used).

Example 9.45

Figure 9.7 shows the process for calculating $y = a^x$ using the Algorithm 9.7 (for simplicity, the modulus is not shown). In this case, $x = 22 = (10110)_2$ in binary. The exponent has five bits.

Figure 9.7 *Demonstration of calculation of a^{22} using square-and-multiply method*



Squaring is done in each step except the last. Multiplication is done only if the corresponding bit is 1. Figure 9.7 shows how the values of y are gradually built until $y = a^{22}$. The solid boxes mean that multiplication is ignored and the previous value of y is carried to the next step. Table 9.3 shows how the value for $y = 17^{22} \bmod 21$ is calculated. The result is $y = 4$.

Table 9.3 *Calculation of $17^{22} \bmod 21$*

i	x_i	*Multiplication (Initialization: $y = 1$)*		*Squaring (Initialization: $a = 17$)*
0	0		→	$a = 17^2 \bmod 21 = 16$
1	1	$y = 1 \times 16 \bmod 21 = 16$	→	$a = 16^2 \bmod 21 = 4$
2	1	$y = 16 \times 4 \bmod 21 = 1$	→	$a = 4^2 \bmod 21 = 16$
3	0		→	$a = 16^2 \bmod 21 = 4$
4	1	$y = 1 \times 4 \bmod 21 = 4$	→	

Complexity Algorithm 9.7 uses a maximum of $2n_b$ arithmetic operations in which n_b is the length of the modulus in bits ($n_b = \log_2 n$), so the bit-operation complexity of the algorithm is $O(n_b)$ or polynomial.

The bit-operation complexity of the fast exponential algorithm is polynomial.

Alternative Algorithm Note that Algorithm 9.7 checks the value of bits in x from the right to the left (least significant to most significant). An algorithm can be written to use the reverse order. We have chosen the above algorithm because the squaring operation is totally independent from the multiplication operation; they can be done in parallel to increase the speed of processing. The alternative algorithm is left as an exercise.

Logarithm

In cryptography, we also need to discuss modular logarithm. If we use exponentiation to encrypt or decrypt, the adversary can use logarithm to attack. We need to know how hard it is to reverse the exponentiation.

Exhaustive Search

The first solution that might come to mind is to solve $x = \log_a y \pmod n$. We can write an algorithm that continuously calculates $y = a^x \bmod n$ until it finds the value of given y. Algorithm 9.8 shows this approach.

Algorithm 9.8 *Exhaustive search for modular logarithm*

```
Modular_Logarithm (a, y, n)
{
    for (x = 1 to n – 1)                          // k is the number of bits in x
    {
        if (y ≡ a^x mod n) return x
    }
    return failure
}
```

Algorithm 9.8 is definitely very inefficient. The bit-operation complexity is $O(2^{n_b})$ or exponential.

Discrete Logarithm

The second approach is to use the concept of **discrete logarithm.** Understanding this concept requires understanding some properties of multiplicative groups.

Finite Multiplicative Group In cryptography, we often use the multiplicative finite group: $G = \langle Z_n^*, \times \rangle$ in which the operation is multiplication. The set Z_n^* contains those integers from 1 to n–1 that are relatively prime to n; the identity element is $e = 1$. Note that when the modulus of the group is a prime, we have $G = \langle Z_p^*, \times \rangle$. This group is the special case of the first group, so we concentrate on the first group in this section.

Order of the Group In Chapter 4, we discussed the order of a finite group, |G|, to be the number of elements in the group G. In $G = \langle Z_n^*, \times \rangle$, it can be proved that the order of group is $\phi(n)$. We have shown how to calculate $\phi(n)$, when n can be factored into primes.

Example 9.46

What is the order of group $G = \langle Z_{21}^*, \times \rangle$? $|G| = \phi(21) = \phi(3) \times \phi(7) = 2 \times 6 = 12$. There are 12 elements in this group: 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, and 20. All are relatively prime with 21.

Order of an Element In Chapter 4, we also discussed the order of an element, ord(a). In $G = \langle Z_n^*, \times \rangle$, we continue with the same definition. The order of an element, a, is the smallest integer i such that $a^i \equiv e \pmod{n}$. The identity element e is 1 in this case.

Example 9.47

Find the order of all elements in $G = \langle Z_{10}^*, \times \rangle$.

Solution

This group has only $\phi(10) = 4$ elements: 1, 3, 7, 9. We can find the order of each element by trial and error. However, recall from Chapter 4 that the order of an element divides the order of the group (Lagrange theorem). The only integers that divide 4 are 1, 2, and 4, which means in each case we need to check only these powers to find the order of the element.

a. $1^1 \equiv 1 \bmod (10) \rightarrow \text{ord}(1) = 1$.

b. $3^1 \equiv 3 \bmod (10)$; $3^2 \equiv 9 \bmod (10)$; $3^4 \equiv 1 \bmod (10) \rightarrow \text{ord}(3) = 4$.

c. $7^1 \equiv 7 \bmod (10)$; $7^2 \equiv 9 \bmod (10)$; $7^4 \equiv 1 \bmod (10) \rightarrow \text{ord}(7) = 4$.

d. $9^1 \equiv 9 \bmod (10)$; $9^2 \equiv 1 \bmod (10) \rightarrow \text{ord}(9) = 2$.

1.4 PRIME NUMBERS

A central concern of number theory is the study of prime numbers. Prime numbers play a critical role in number theory.

An integer p>1 is a prime number if and only if its only divisors are ±1 and ± p.

Any integer a>1 can be factored in a unique way as

Where p1<p2<…<Pt. and where each is ai is a positive integer. This is known as the fundamental theorem of arithmetic;

$$91 = 7 \times 13$$
$$3600 = 2^4 \times 3^2 \times 5^2$$
$$11011 = 7 \times 11^2 \times 13$$

If P is the set of all prime numbers, then any positive integer can be written uniquely in the following form:

$$a = \prod_{p \in P} p^{a_p} \qquad \text{where each } a_p \geq 0$$

The right-hand side is the product over all possible prime numbers ; for any particular value of a, most of the exponents ap will be 0.

The value of any given positive integer can be specified by simply listing all the nonzero exponents in the foregoing formulation.

The integer 12 is represented by $\{a_2 = 2, a_3 = 1\}$.
The integer 18 is represented by $\{a_2 = 1, a_3 = 2\}$.
The integer 91 is represented by $\{a_7 = 1, a_{13} = 1\}$.

Multiplication of two numbers is equivalent to adding the corresponding exponents. Given

$$a = \prod_{p \in \mathrm{P}} p^{a_p}, \; b = \prod_{p \in \mathrm{P}} p^{b_p}$$

Define k = ab. We know that the integer k can be expressed as the product of powers

$$k = \prod_{p \in P} p^{k_p}$$

It follows that $k_p = a_p + b_p$ for all $p \in P$.

$k = 12 \times 18 = (2^2 \times 3) \times (2 \times 3^2) = 216$
$k_2 = 2 + 1 = 3; k_3 = 1 + 2 = 3$
$216 = 2^3 \times 3^3 = 8 \times 27$

Any integer of the form p^n can be divided only by an integer that is of a lesser or equal power of the same prime number, p^j with j<= n. Thus, we can say the following. Given

$$a = \prod_{p \in P} p^{a_p}, b = \prod_{p \in P} p^{b_p}$$

If a|b, then $a_p \leq b_p$ for all p.

$a = 12; b = 36; 12|36$
$12 = 2^2 \times 3; 36 = 2^2 \times 3^2$
$a_2 = 2 = b_2$
$a_3 = 1 \leq 2 = b_3$
Thus, the inequality $a_p \leq b_p$ is satisfied for all prime numbers.

It is easy to determine the greatest common divisor3 of two positive integers if we express each integer as the product of primes.

$$300 = 2^2 \times 3^1 \times 5^2$$
$$18 = 2^1 \times 3^2$$
$$\gcd(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$$

The following relationship always holds:

$$\text{If } k = \gcd(a, b), \text{ then } k_p = \min(a_p, b_p) \text{ for all } p.$$

PRINCIPLES OF PUBLIC-KEY CRYPTOSYSTEMS

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption.

The first problem is that of key distribution, The second problem that Diffie pondered, and one that was apparently unrelated to the first, was that of *digital signatures.*

Public-Key Cryptosystems: Asymmetric algorithms rely on one key for encryption and a different but related key for decryption.

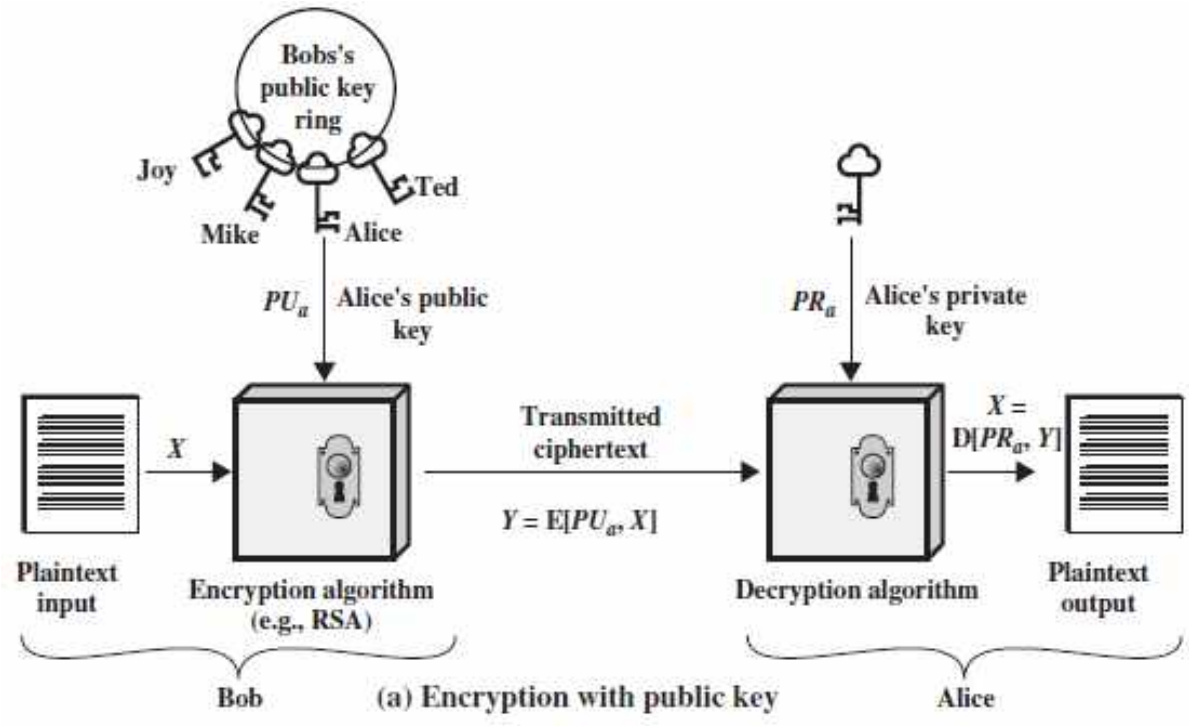
These algorithms have the following important characteristic.

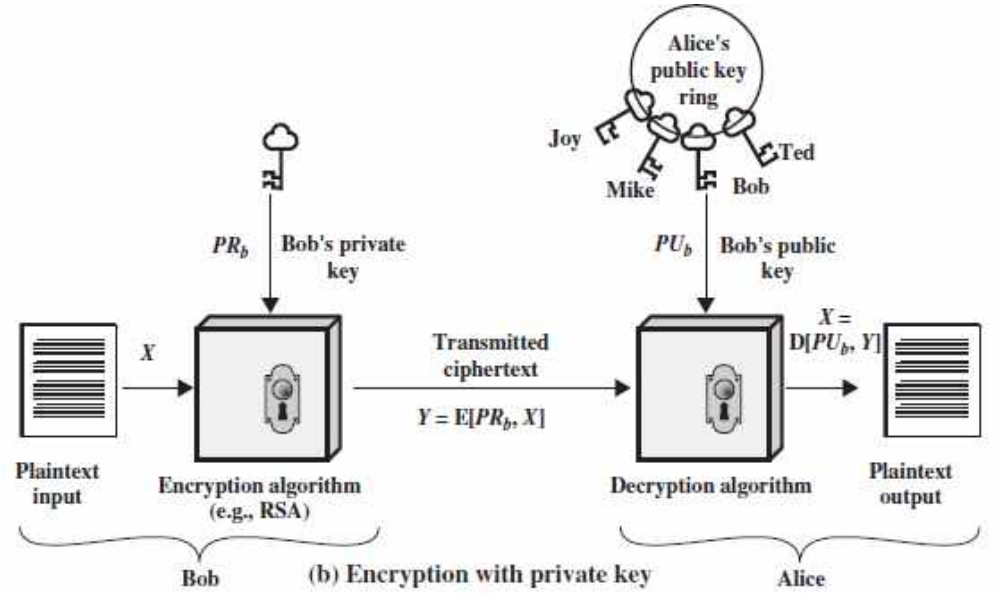
- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

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 shown in Figure a & b below:





- **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 keys:** 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 erformed by the 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 ciphertexts.

• **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

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 a suggests, each user maintains a collection of public keys obtained from others.

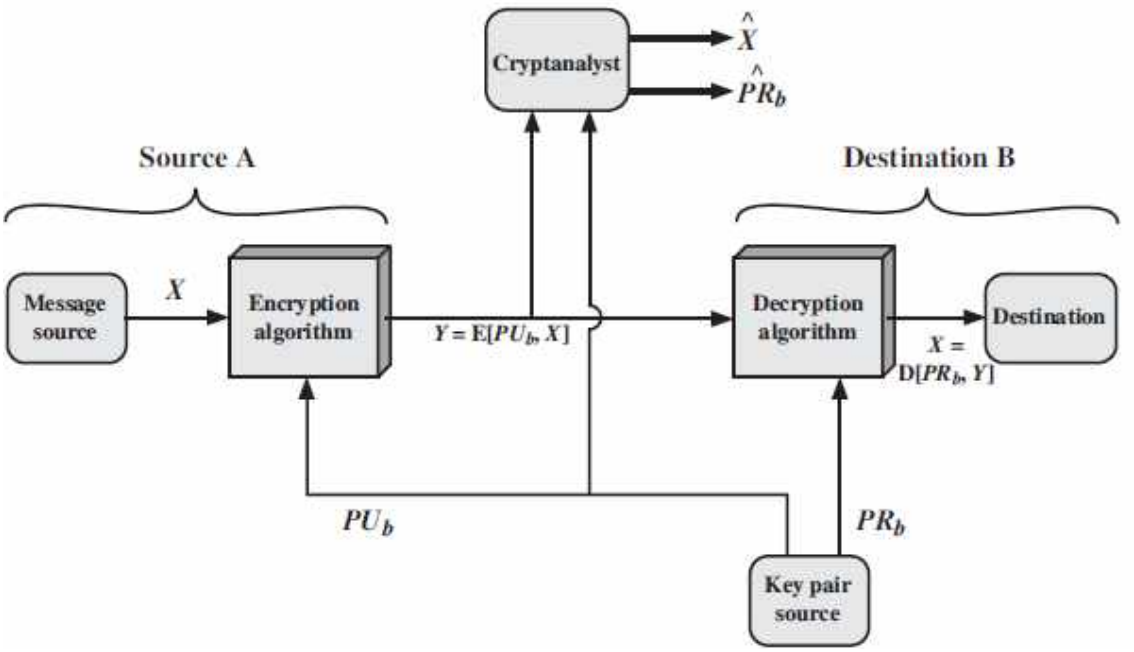
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.

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.

Table below summarizes some of the important aspects of symmetric and public key encryption.

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Let us take a closer look at the essential elements of a public-key encryption scheme, using Figure below:Public-Key Cryptosystem: Secrecy



There is some source A that produces a message in plaintext, X=[x1, x2,…, x_M] . The M elements of X are letters in some finite alphabet. The message is

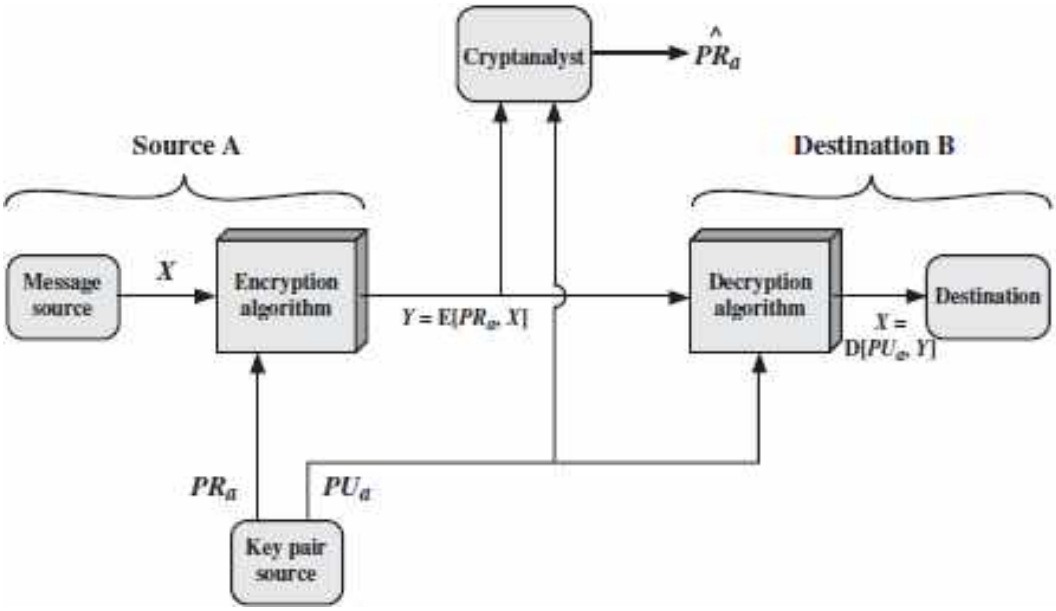
intended for destination B. B generates a related pair of keys: a public key, PU_b and a private key, PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A. With the message *X and the encryption key PUb as input, A forms the* ciphertext *Y* =[y1,y2,…,y_N] :

$$Y = E(PUb, X)$$

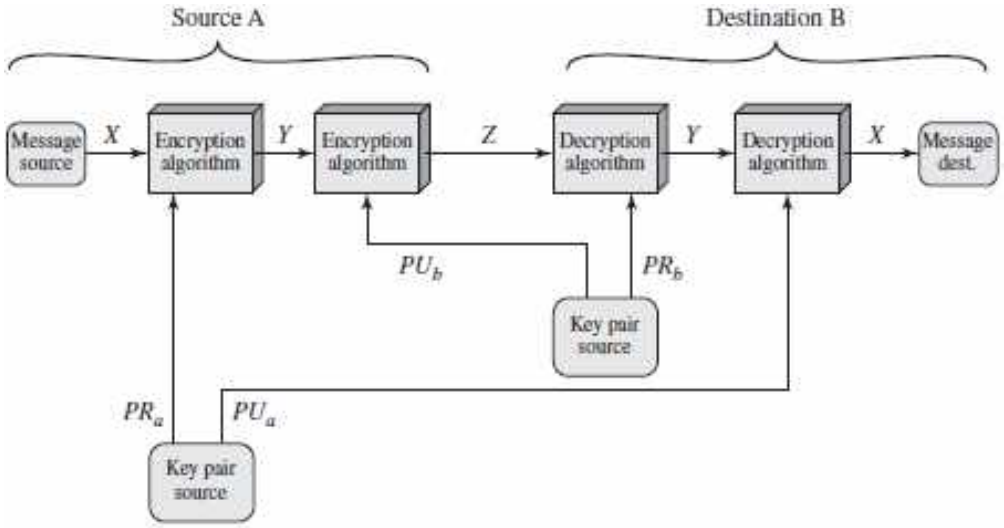
The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = D(PRb, Y)$$

Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Authentication and Secrecy



Applications for Public-Key Cryptosystems

we can classify the use of public-key cryptosystems into three categories

• **Encryption /decryption:** The sender encrypts a message with the recipient's public key.

• **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

• **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

THE RSA ALGORITHM: One of the first successful responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978.

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.

That is, *n is less than* 2^{1024}

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

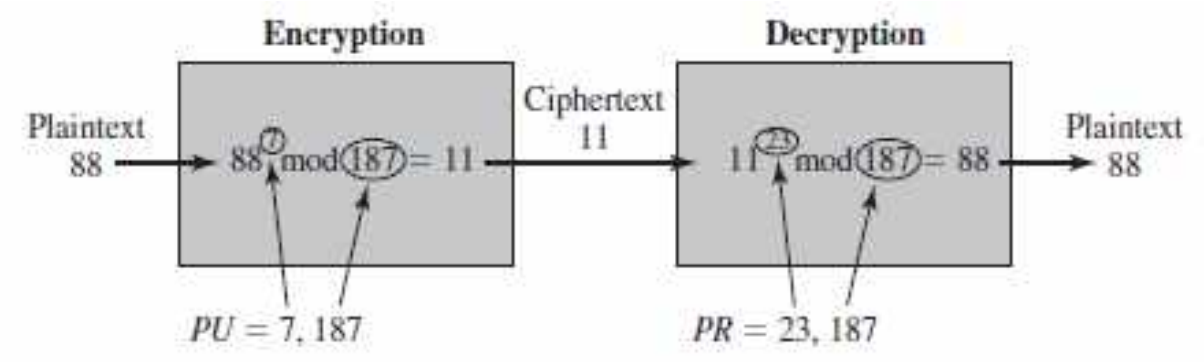
Encryption:

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption:

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Example of RSA Algorithm:



Example: Key Generation Process:

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate n = pq = 17 × 11 = 187.
3. Calculate ϕ (n) = (p - 1)(q - 1) = 16 × 10 = 160.
4. Select e such that e is relatively prime to ϕ(n) = 160 and less than ϕ (n); we choose e = 7.
5. Determine d such that de ≅ 1 (mod 160) and d < 160.The correct value is d = 23, because 23 × 7 = 161 = (1 × 160) + 1;

The resulting keys are public key *PU = {7, 187} and private key PR = {23, 187}.*

Example: Encryption: Plaintext input of *M= 88.* For encryption,
we need to calculate $C = 88^7$ mod 187.Exploiting the properties of modular arithmetic, we can do this as follows.

88^7 mod 187 = [(88^4 mod 187) × (88^2 mod 187)× (88^1 mod 187)] mod 187

88^1 mod 187 = 88

88^2 mod 187 = 7744 mod 187 = 77

DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography and is generally referred to as Diffie-Hellman key exchange. A number of commercial products employ this key exchange technique.

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

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

A primitive root of a prime number as one whose powers modulo generate all the integers from 1 to (p-1). That is, if is a primitive root of the prime number , then the numbers

a mod p, a^2 mod p, ..., a^{p-1} mod p

are distinct and consist of the integers from 1 through (p-1) in some permutation.

For any integer b and a primitive root a of prime number p , we can find a unique exponent i such that

$b = a^i \pmod p$ where $0 \leq i \leq (p-1)$

The exponent is referred to as the discrete logarithm of b for the base a, mod p. We express this value as $\text{dlog}_{a,p}(b)$.

The Algorithm: The following summarizes the Diffie-Hellman key exchange algorithm.

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

For this scheme, there are two publicly known numbers: a prime number and an integer α that is a primitive root of q.

Suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and Calculates Public key Y_A as shown below:

User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

Similarly, user B independently selects a random integer $X_B < q$ and computes Y_B as shown below:

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Each side keeps the X value private and makes the Y value available publicly to the other side.

User A computes the key K as shown below:

Calculation of Secret Key by User A
$K = (Y_B)^{XA} \bmod q$

And user B computes the key K as shown below:

Calculation of Secret Key by User B
$K = (Y_A)^{XB} \bmod q$

The above two calculations produce identical results:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{by the rules of modular arithmetic} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

The result is that the two sides have exchanged a secret value.

ELGAMAL CRYPTOGRAPHIC SYSTEM

In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique.

The ElGamal cryptosystem is used in some form in a number of standards including the digital signature standard (DSS), and the S/MIME e-mail standard.

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q
Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{XA} \bmod q$
Public key	$PU = \{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{XA} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}, as shown in Table 8.3. We choose $\alpha = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^5 \bmod 19 = 3$ (see Table 8.3).
3. Alice's private key is 5; Alice's public key is $\{q, \alpha, Y_A\} = \{19, 10, 3\}$.

Suppose Bob wants to send the message with the value $M = 17$. Then,

1. Bob chooses $k = 6$.
2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$.
3. So
   $C_1 = \alpha^k \bmod q = \alpha^k \bmod 19 = 11$
   $C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext (11, 5).

For decryption:

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2 K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$.

Elliptic Curve Cryptography

- **Definition: Elliptic curve cryptography (ECC)** is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. These are analogy of existing public key cryptosystem in which modular arithmetic is replaced by operations defined over elliptic curve.
- The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in **1985.**
- Elliptic curve cryptography (ECC) is one of the most powerful but least understood types of cryptography in wide use today. An increasing number of websites make extensive use of ECC to secure everything from customers' HTTPS connections to how they pass data between data centers.

An elliptic curve is defined by an equation in two variables with coefficients. For

cryptography, the variables and coefficients are restricted to elements in a finite field,

which results in the definition of a finite abelian group.

Elliptic Curves over Real Numbers

Elliptic curves are not ellipses. They are so named because they are described by cubic equations

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

is similar to equation of calculating circumference of an ellipse.

Where

a,b,c,d and e → real numbers.

X and Y are→taken on values in the real numbers.

For utilization of this in cryptography

$Y^2 = X^3+ax+b$ → EQ1, is sufficient.

Such equations are said to be cubic, or of degree 3, because the highest exponent they contain is a 3. Also included in the definition of an elliptic curve is a single element denoted *O* and called the *point at infinity* or the *zero point. To plot such a curve, we need to compute*

$$y = \sqrt{x^3 + ax + b}$$

For given values of and, the plot consists of positive and negative values of for each value of . Thus, each curve is symmetric about $y = 0$.

Two families of elliptic curves are used in cryptographic applications:

- Prime curves over $\mathbf{Z_p}$ ***[it is** Best for software application**]***
- Binary curves over **GF(2^m)** ***[it is** Best for software application**]***

Prime curves over Z_p

- In Prime curves over $\mathbf{Z_p}$ **,** $\boldsymbol{p}$ → referred to as a modulus.
- ***we use a cubic equation in which the variables and coefficients all take on values in the set of integers from 0** through **p - 1** and in which calculations are performed modulo **p.***
- ***from EQ1, in this case coefficients and variables limited to Z_p.***

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

-- eq2

Now consider the set E*p*(*a*, *b*) consisting of all pairs of integers (*x*, *y*) that satisfy

Equation eq2 together with a point at infinity .The coefficients a and b and the variables x and y are all elements of Zp.

For example, Equation eq2 is satisfied for $a = 1, b = 1, x = 9, y = 7, \alpha = 1$ $p = 23$:

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3 = 3$$

For example, let *p* = 23 and consider the elliptic curve *y*2 = *x*3 + *x* + 1 In this case, *a* = *b* = 1

For the set $E_{23}(1, 1)$, we are only interested in the nonnegative integers in the quadrant from (0, 0) through (*p* - 1, *p* - 1) that satisfy the equation mod *p.*

Elliptic Curves over GF(2^m):

A finite field GF(2^m) consists of 2m elements, together with addition & multiplication operations that can be defined over polynomials.

For elliptic Curves over GF(2^m), we use a cubic equation in which the variables and coefficients all take on values in GF(2^m), for some number m.

By this, the form of cubic equation appropriate for cryptographic application.

The form is $y^2 + xy = x^3 + ax^2 + b$ → EQ3.

To form a cryptographic system using elliptic curves, we need to find a "hard problem" corresponding to factoring the product of two primes or taking the discrete logarithm.

Consider the equation

$$Q = kP \text{ where } Q, P \in E_p(a, b) \text{ and } k < p.$$

It is relatively easy to calculate *Q* given *k and P*

But it is relatively hard to determine given Q and P.

This is called the discrete logarithm problem for elliptic curves.

ECC Diffie-Hellman Key Exchange:

ECC can do key exchange, that is analogous to Diffie Hellman.

Key exchange using elliptic curves can be done in the following manner.

First pick a large integer q , which is either a prime number P or an integer of the form 2^m and elliptic curve parameters a & b for equation

$Y^2 \bmod p = (x^3+ax+b) \bmod p$

or

$$y^2 + xy = x^3 + ax^2 + b$$

This define elliptic group of point $E_q(a,b)$.

Pick a base point G=(x1,y1) in $E_p(a,b)$ whose order is a very large value n.

The order n of a point G on an elliptic curve is the smallest +ve integer n such that $nG=0.E_q(a,b)$

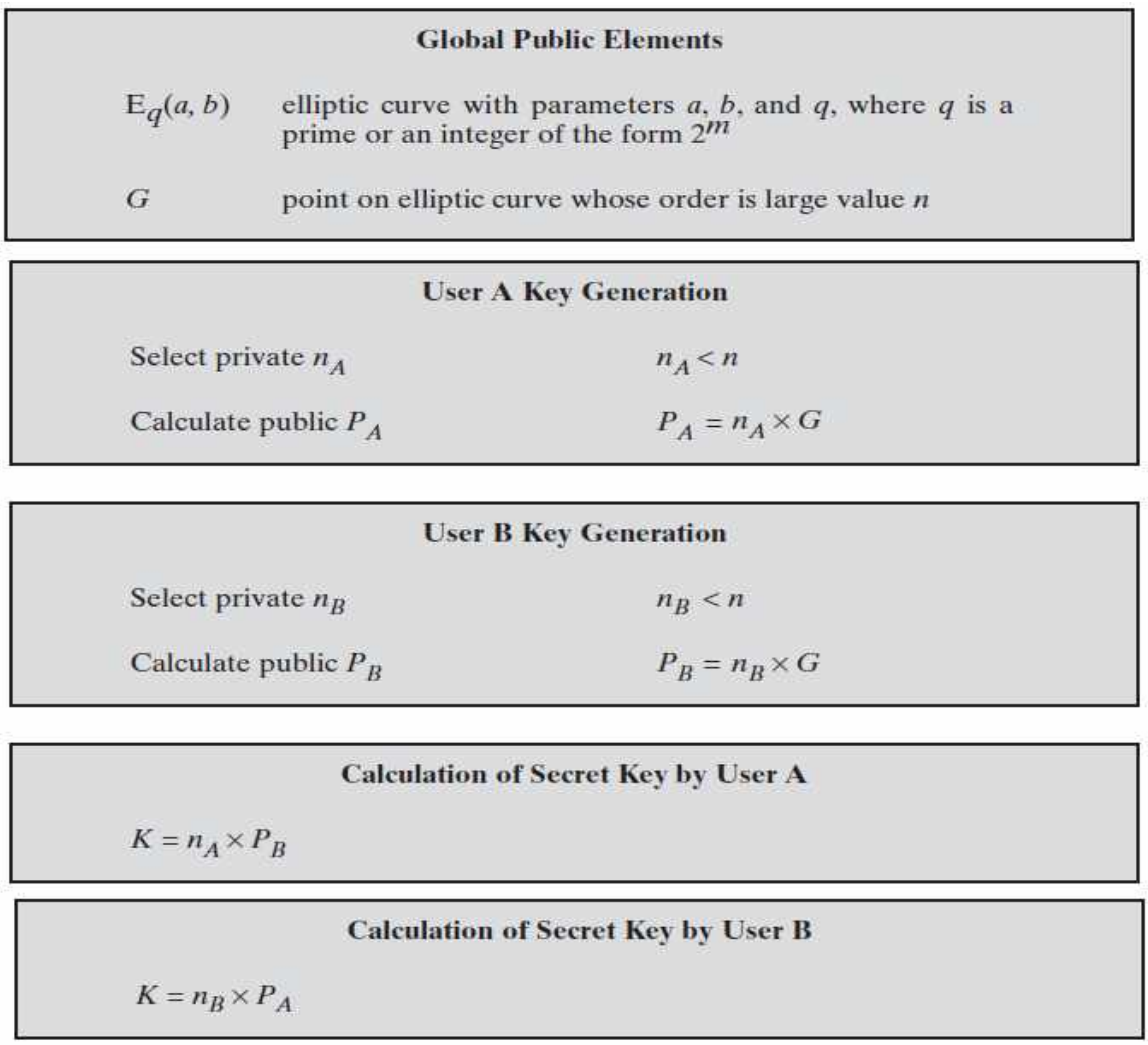


Figure 10.7 ECC Diffie-Hellman Key Exchange

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

Elliptic Curve Encryption/Decryption:

Several approaches to encryption/decryption using elliptic curves have been analyzed in the literature. In this subsection, we look at perhaps the simplest. The first task in this system is to encode the plaintext message m to be sent as an x–y point P_m. It is the point P_m that will be encrypted as a ciphertext and subsequently decrypted. Note that we cannot simply encode the message as the x or y coordinate of a point, because not all such coordinates are in $E_q(a, b)$; for example, see Table 10.1. Again, there are several approaches to this encoding, which we will not address here, but suffice it to say that there are relatively straightforward techniques that can be used.

As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group $E_q(a, b)$ as parameters. Each user A selects a private key n_A and generates a public key $P_A = n_A \times G$.

To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points:

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

Note that A has used B's public key P_B. To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

A has masked the message P_m by adding kP_B to it. Nobody but A knows the value of k, so even though P_b is a public key, nobody can remove the mask kP_B. However, A also includes a "clue," which is enough to remove the mask if one knows the private key n_B. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed to be hard.

As an example of the encryption process (taken from [KOBL94]), take $p = 751$; $E_p(-1, 188)$, which is equivalent to the curve $y^2 = x^3 - x + 188$; and $G = (0, 376)$. Suppose that A wishes to send a message to B that is encoded in the elliptic point $P_m = (562, 201)$ and that A selects the random number $k = 386$. B's public key is $P_{B=(201,5)}$. We have $386(0, 376) = (676, 558)$, and $(562, 201) + 386(201, 5) = (385, 328)$. Thus, A sends the cipher text $\{(676, 558), (385, 328)\}$.

QUESTION BANK

1. Explain public key encryption scheme
2. Perform encryption and decryption using RSA algorithm for p=3,q=11,e=7 and M=5.
3. Explain public key cryptosystem for secrecy and authentication.
4. Explain Deffie-Hellman key exchange.
5. What are the principal elements of a public key cryptosystem? Explain.
6. In a public key system using RSA, you intercept the cipher text C=10 sent to a user whose public key is e=5, n= 35. What is the plain text M?
7. Explain RSA algorithm in detail.
8. Perform encryption and decryption using RSA algorithm for p=5,q=11,e=3 and M=9.

9. Describe in general terms an efficient procedure for picking a prime number
10. What requirements must a public key cryptosystems fulfill to be a secure algorithm?
11. Explain RSA algorithm in detail with an example.
12. Compare and contrast different secure hash functions.