LECTURE NOTES

CRYPTOGRAPHY AND NETWORK SECURITY

Name Of The Programme	B.Tech-CSE
Regulations	R-16
Year and Semester	4th Year, 1st Semester
Name of the Course Coordinator	Mr.Ch.Samsonu
Name of the Module Coordinator	Mr. B. Rama Krishna
Name of the Program Coordinator	Mr.N.Md.Jubair Basha

KHIT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY

NH-5, Chowdavaram Village, Guntur, Andhra Pradesh, India Approved By AICET, New Delhi, Permanently Affiliated to JNTUK Kakinada
Accredited By NBA, Accredited By NAAC with A Grade

ACADEMIC YEAR 2020-21

Index

UNIT -I

Security Goals, Cryptographic Attacks, Services and Mechanisms, Mathematics of Cryptography

WHAT IS NETWORK SECURITY ?

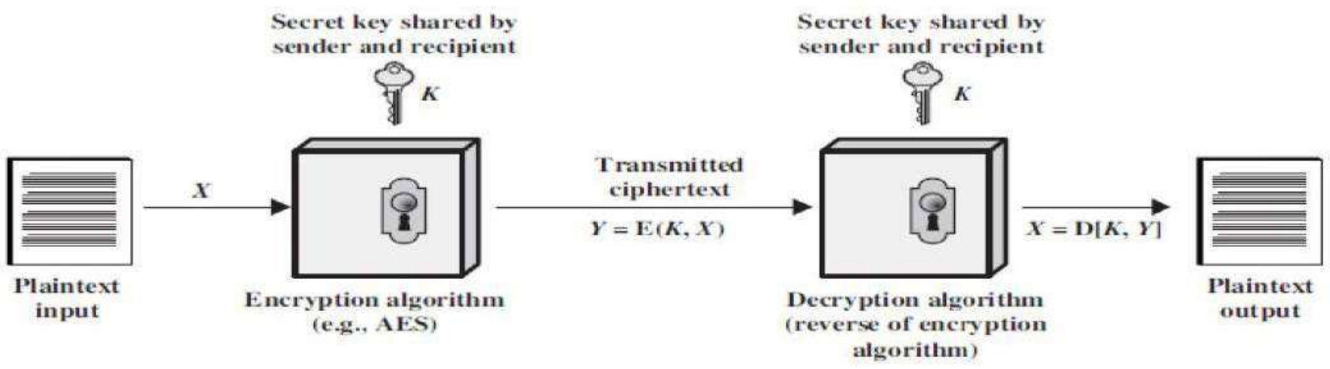
Network Security consists of the *provisions and policies* adapted by network Administrator to *prevent and monitor* unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources.

WHAT IS CRYPTOGRAPHY?

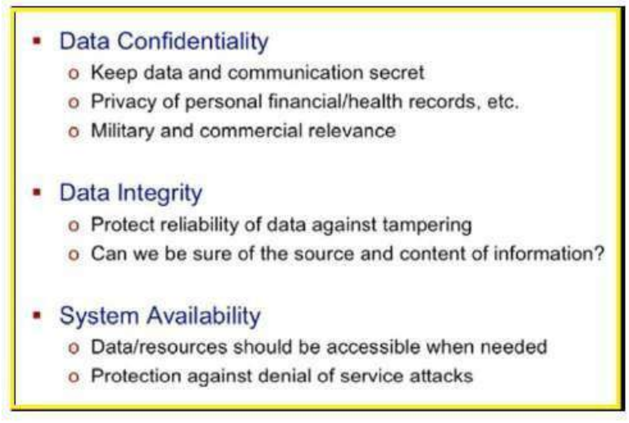
Cryptography is the *study of secure communications techniques* that allow only the sender and intended recipient of a message to view its contents.
The term is derived from the Greek word *kryptos*, which means hidden.

MODEL FOR NETWORK SECURITY - TERMINOLOGY

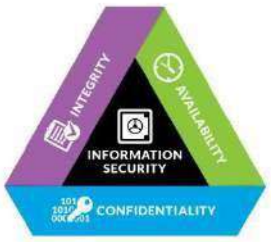
- Plaintext - the original message
- Cipher text - the coded message
- Cipher - algorithm for transforming plaintext to cipher text
- Key - info used in cipher known only to sender/receiver
- Encipher (Encrypt) - converting plaintext to cipher text
- Decipher (Decrypt) - recovering cipher text from plaintext
- Cryptography - study of encryption principles/methods
- Cryptanalysis (code breaking) - the study of principles/ methods of deciphering cipher text *without* knowing key
- Cryptology - the field of both cryptography and cryptanalysis


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D[K, Y]
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

SECURITY GOALS



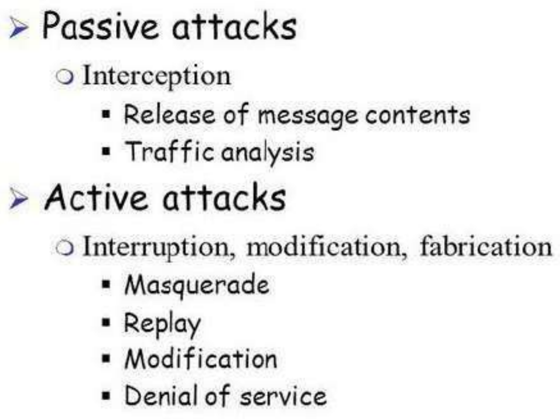
- Data Confidentiality
  - Keep data and communication secret
  - Privacy of personal financial/health records, etc.
  - Military and commercial relevance
- Data Integrity
  - Protect reliability of data against tampering
  - Can we be sure of the source and content of information?
- System Availability
  - Data/resources should be accessible when needed
  - Protection against denial of service attacks



Cryptographic Attacks

Accessing of data by unauthorized entity is called as attack
Passive Attacks
Active Attacks

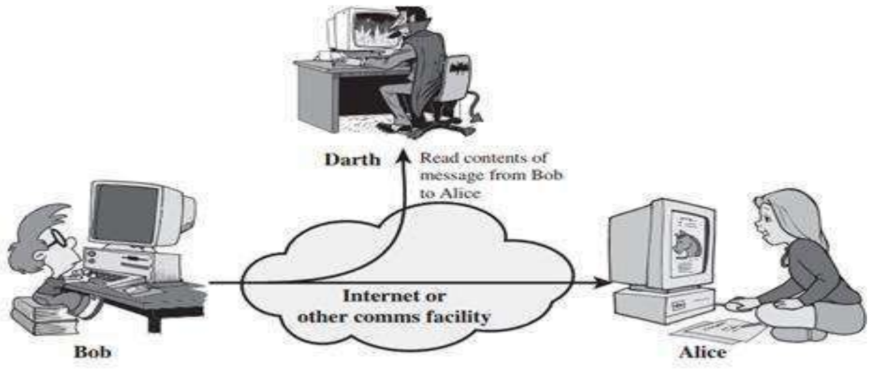
Passive Attacks:
In a passive attack, the attacker's goal is just to obtain information. This means that the attack does not modify data or harm the system.
Active Attacks:
An active attack may change the data or harm the system. Attacks that threaten the integrity and availability are active attacks.



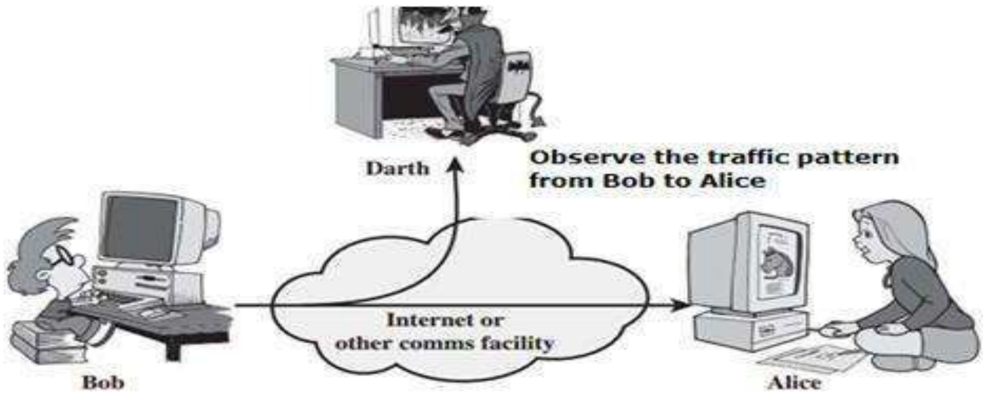
- Passive attacks
  - Interception
    - Release of message contents
    - Traffic analysis
- Active attacks
  - Interruption, modification, fabrication
    - Masquerade
    - Replay
    - Modification
    - Denial of service

Passive Attacks

(a) Release of message content –
Capture and read the content transmissions.
(b) Traffic Analysis–
- can't read the information, but observe the pattern
- determine the location and identity of communicating parties
- observe frequency and length of communication


Darth
Read contents of message from Bob to Alice
Internet or other comms facility
Bob
Alice

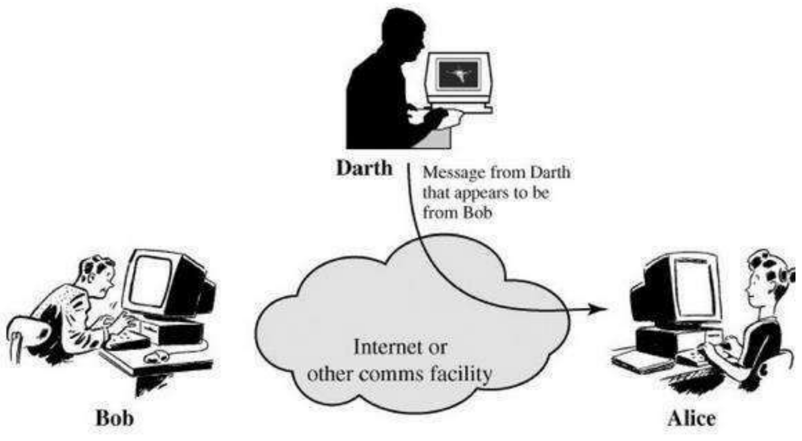
(a) Release of Message content


Darth
Observe the traffic pattern from Bob to Alice
Internet or other comms facility
Bob
Alice

(b) Traffic Analysis

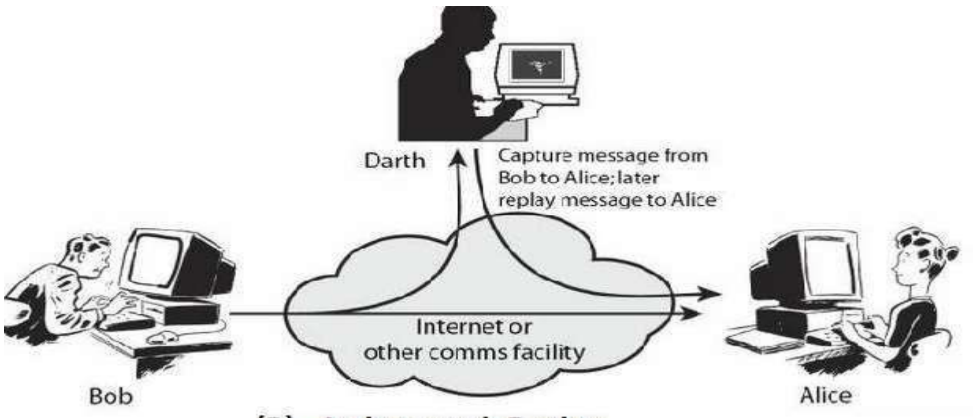
Active Attacks

(a) Masquerading: Masquerading or snooping happens when the attacker impersonates somebody else.

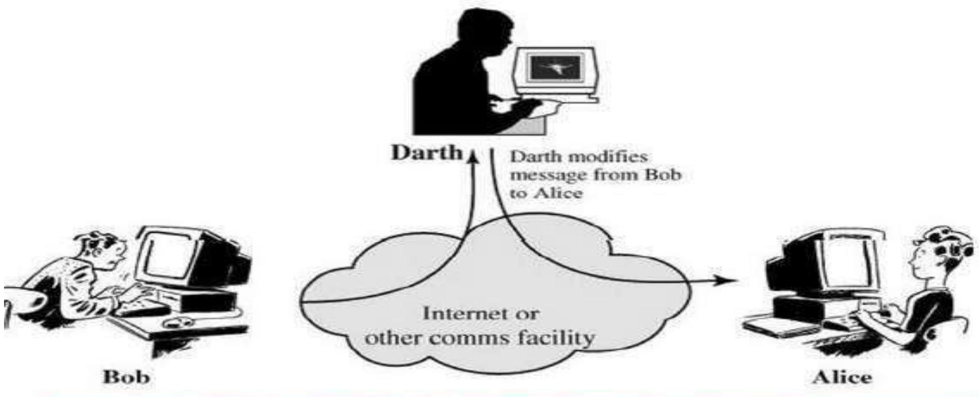

Darth
Message from Darth that appears to be from Bob
Internet or other comms facility
Bob
Alice

Active Attack – Masquerade

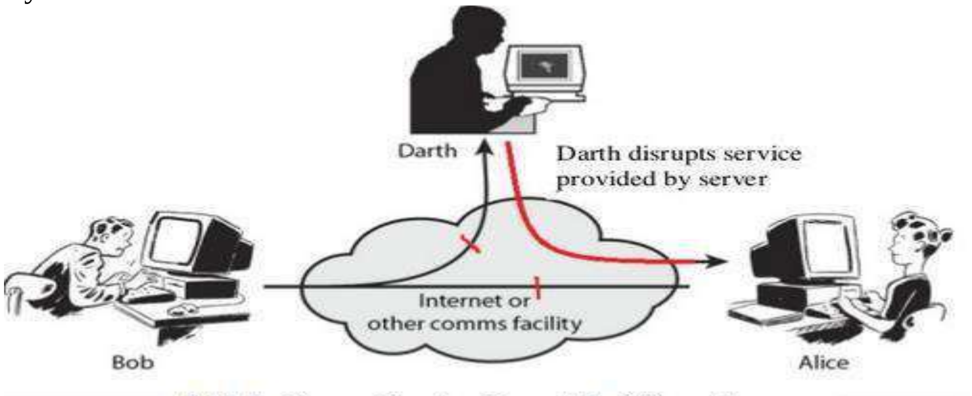
(b) Replay–
The attacker obtains a copy of a message sent by a user and later tries to replay it.


Darth
Capture message from Bob to Alice; later replay message to Alice
Internet or other comms facility
Bob
Alice
(B) Active attack-Replay

(c) Modification: After intercepting or accessing information, the attacker modifies the information then send to receiver.


Darth
Darth modifies message from Bob to Alice
Internet or other comms facility
Bob
Alice
Active Attack – Modification of messages

(d) Denial of service: Denial of service (Dos) is a very common attack.it may slow down or totally interrupt the service of a system.


Darth
Darth disrupts service provided by server
Internet or other comms facility
Bob
Alice
(d) Active attack - Denial of Service

Cryptographic Attacks Categories

Cryptographic attacks can be broadly categorized into two distinct types:

- Cryptanalytic
- Non-Cryptanalytic

Cryptanalytic Attacks:

- These attacks are combinations of statistical and algebraic techniques aimed at discover the secret key of a cipher.

- The attacker thus guesses the key and looks for the distinguishing property. if the property is detected, the guess is correct otherwise the next guess is tried.

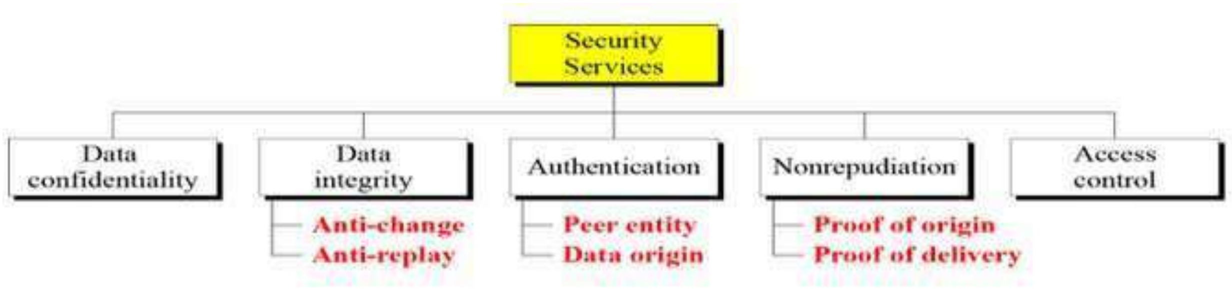
Non-Cryptanalytic Attacks:

- The other types of attacks are non-cryptanalytic attacks, which do not explain the mathematical weakness of the cryptographic algorithm.

SERVICES AND MECHANISM

Security Services

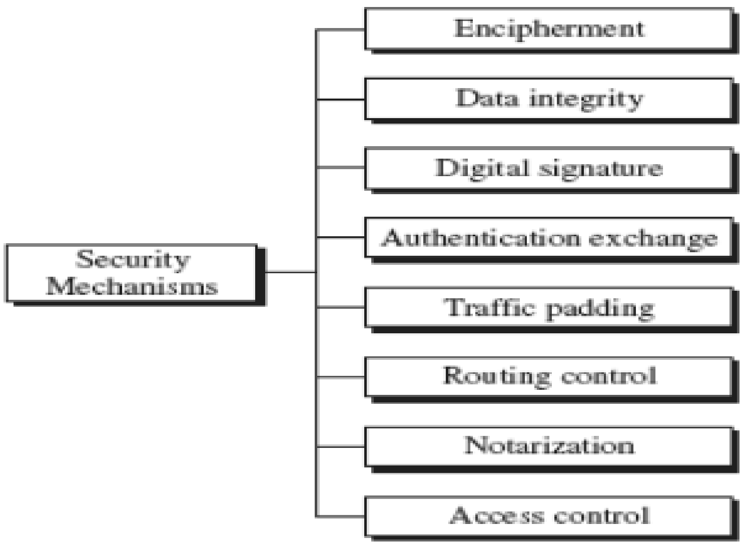
ITU-T (X .800) is provided by protocol layer of transmission that defines security services ensures security of the data transfer



- Data Confidentiality: It is designed to protect data from disclosure attack.. That is, it is designed to prevent snooping and traffic analysis attack.
- Data Integrity: It is designed to protect data from modification, insertion, deletion and replaying by an adversary
- Authentication: It provides the authentication of the party at the other end of the line.
- Non-repudiation: It protects against repudiation by either the sender or the receiver of the data.
- Access Control: It provides protection against unauthorized access to data

Security Mechanism:

ITU-T recommends Security mechanisms to provide the security services



- Encipherment:The use of mathematical algorithms to transform data into a form that is not readily understandable

- Data Integrity:A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
- Digital Signature:A digital signature is a means by which the sender can electronically sign the data and the receiver can electronically verify the signature.
- Authentication Exchange: A mechanism intended to ensure the identity of an entity by means of information exchange.
- Routing Control:Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.
- Traffic Padding: Inserting bogus data to prevent traffic analysis.
- Notarization:The use of a trusted third party to assure certain properties of a data exchange.
- Access Control:A variety of mechanisms that enforce access rights to resources.

Relation Security Services and Mechanisms

➢ Security Mechanism: A mechanism that is designed to detect, prevent, or recover from a security attack.

➢Security Service: A service that enhances the security of data processing systems and information transfers. A security service makes use of one or more security mechanisms.

Services	**Mechanisms**
Confidentiality	**Encryption, routing control**
Integrity	**Digital Signature, Encryption**
Authentication	**Encryption, Digital Signature**
Non-repudiation	**Digital Signature, Notarization**
Access Control	**Interactive Proofs, access control mechanisms and policies.**

MATHEMATICS OF CRYPTOGRAPHY

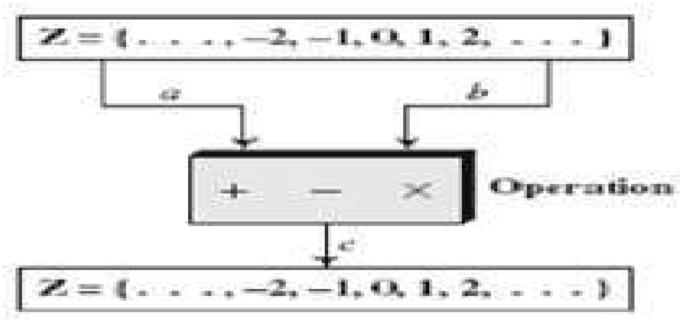
Integer Arithmetic: In Integer arithmetic, we are use a set and a few operations.

➢ Set of Integers: The set of Integers, denoted by z, contains all integral numbers (with no fraction) from negative infinity to positive infinity.

$$\boxed{Z = \{\ldots, -2, -1, 0, 1, 2, \ldots\}}$$

Fig. 2.1 *The set of integers*

➢ Binary Operations: A Binary operation takes two inputs and creates one output. Three common binary operations defined for integers are addition, subtraction and multiplication.


Z = { . . . , −2, −1, 0, 1, 2, . . . }
a
b
+ − ×
Operation
c
Z = { . . . , −2, −1, 0, 1, 2, . . . }

- Examples:

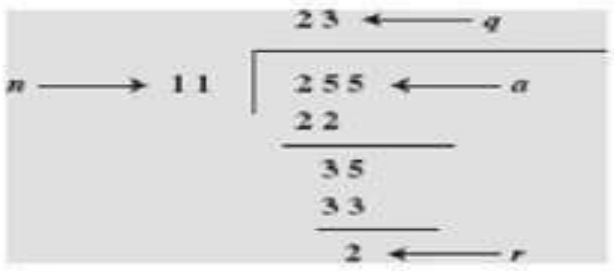
Add:	5+9=14	(-5)+9=4	5+(-9)=-4
Subtract:	5-9=-4	(-5)-9=14	5-(-9)=14
Multiply:	5x9=45	(-5)x9=-45	5x(-9)=45

Integer Division: if we divide a by n, we can get q and r. The relationship between these four integers can be shown as

a=q x n + r

a is dividend, n is the divisor, q is quotient , r is remainder

- Examples: Assume that a = 255 and n = 11. We can find q = 23 and r = 2 using the division algorithm. We have shown in following

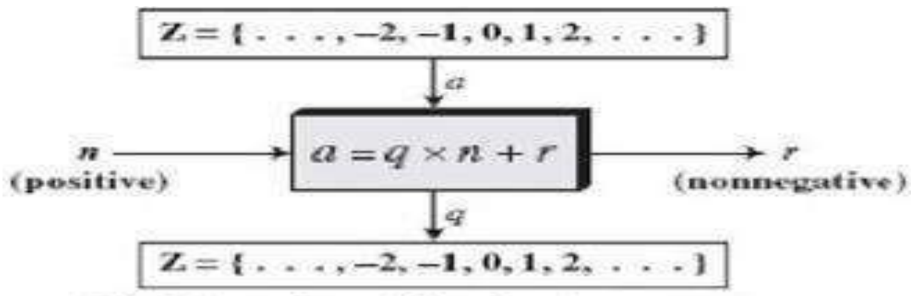


finding the quotient and the remainder

<u>Two Restrictions</u>:

- First, we require that the divisor be a positive integer (n > 0).
- Second, we require that the remainder be a non-negative integer (r ≥ 0).

<u>Integer Division</u>



Z = { . . . , −2, −1, 0, 1, 2, . . . }
a
n (positive)
a = q × n + r
r (nonnegative)
q
Z = { . . . , −2, −1, 0, 1, 2, . . . }

Division algorithm for integers

Examples: Assume r and q are negative when 'a' is negative.

- To make r positive, decrement q by 1 and add value of n tor
- consider -255=(-23x 11) +(-2) ↔ -255=(-24x11)+9
- We have decremented -23 to -24 and added 11 to -2 to make 9.

The relation is still valid

<u>Divisibility:</u>

If a is not zero and we let r = 0 in the division relation, we get

a = q x n

We then say that n divides a (or n is a divisor of a). We can also say that a is divisible by n. The above is n | a .

If the remainder is not zero, then n does not divide a and we can write the relationship as a + n.

- Examples: The integer 4 divides the integer 32 because 32 = 8 x 4.
  We show this is as 4 |32
- The number 8 does not divide the number 42 because 42 = 5 x 8 + 2. There is a remainder, the number 2, in the equation.
  We show this as 8 + 42.

- Examples: The integer 4 divides the integer 32 because 32 = 8 x 4.
  We show this is as 4 |32
- The number 8 does not divide the number 42 because 42 = 5 x 8 + 2. There is a remainder, the number 2, in the equation.

We show this as 8 + 42.

Examples:

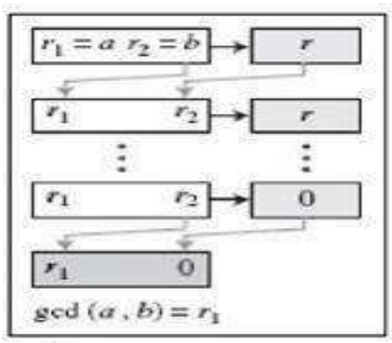
1) Since 3 | 15 and 15 | 45, according to third property, 3 |45
2) Since 3 | 15 and 3 | 9, according to the fourth property, 3 |(15 x 2 + 9 x 4), which means 3 | 66.

Greatest Common Divisor(GCD)

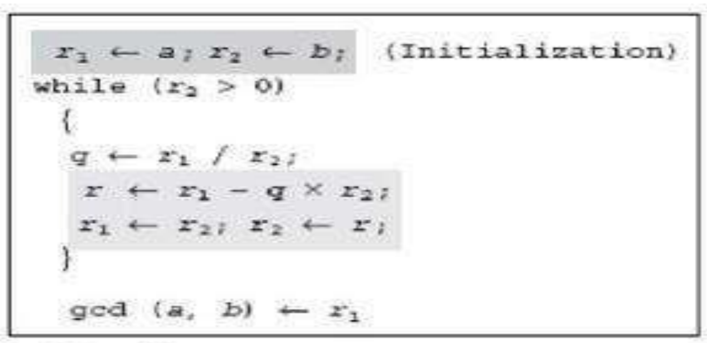
The greatest common divisor of two positive integers is the largest integer that can divide both integers we can write the relationship as a + n.

Examples: GCD of 15 and 20 is 2 because divisors of 15 are 3,5 and divisors of 20 are 2,4,5,10. The GCD is 5

- Euclidean Algorithm:
- Euclidean algorithm is used to finding the greatest common divisor (gcd) of two positive integers. The Euclidean algorithm is based on the following two facts
  - Fact 1: gcd (a, 0) = a
  - Fact 2: gcd (a, b) = gcd (b , r), where r is the remainder of dividing a by b
  - When gcd (a, b) = 1, we say that a and b are relatively prime.



a. Process



```
r1 ← a; r2 ← b;    (Initialization)
while (r2 > 0)
{
  q ← r1 / r2;
    r ← r1 − q × r2;
    r1 ← r2; r2 ← r;
}
  gcd (a, b) ← r1
```

b. Algorithm

Example: gcd (36, 10) = ?

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

Example: gcd (2740,1760) = ?

Solution: we initialize r_1 to 2740 and r_2 to 1760

Answer:

gcd (2740, 1760) = 20

.

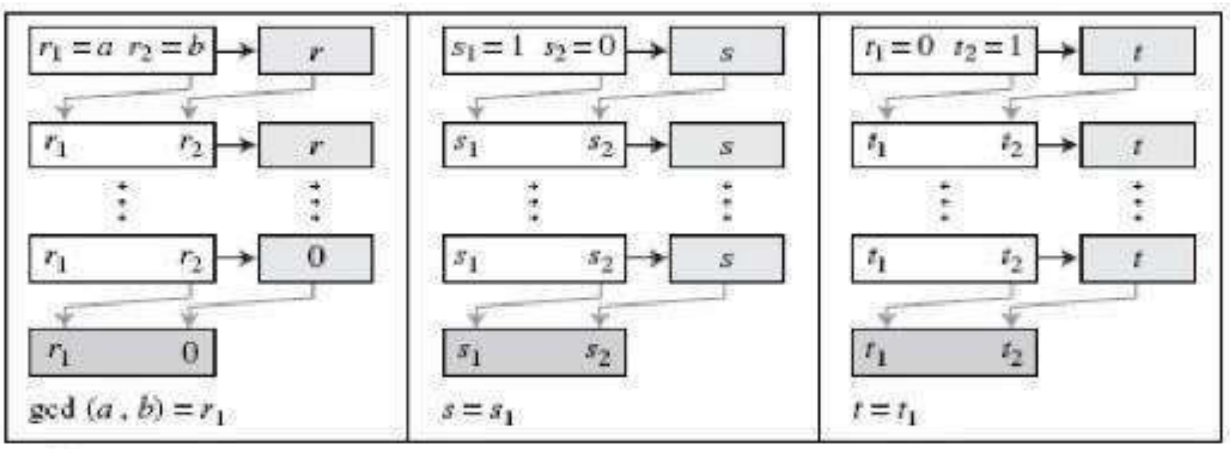
q	r_1	r_2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

Extended Euclidean Algorithm

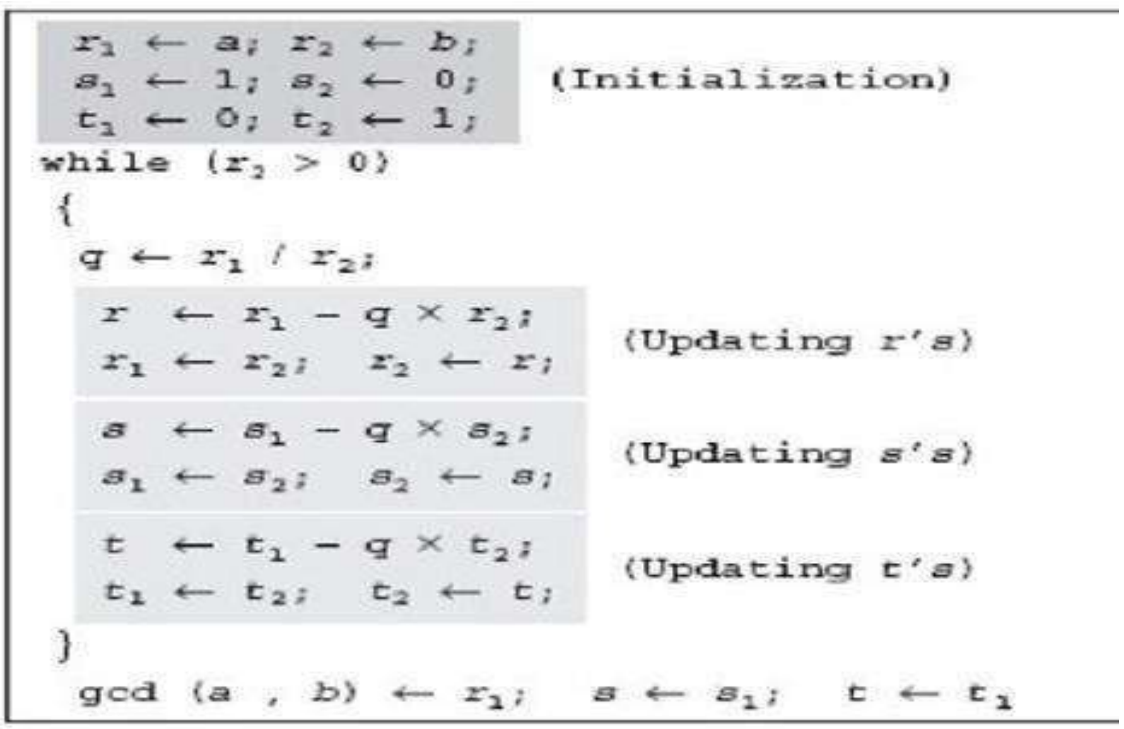
- Given two integers a and b, we often need to find other two integers, s and t, such that

$$\boxed{s \times a + t \times b = \gcd(a,b)}$$

- The Extended Euclidean Algorithm can calculate the gcd (a, b) and at the same time calculate the value if s and t.



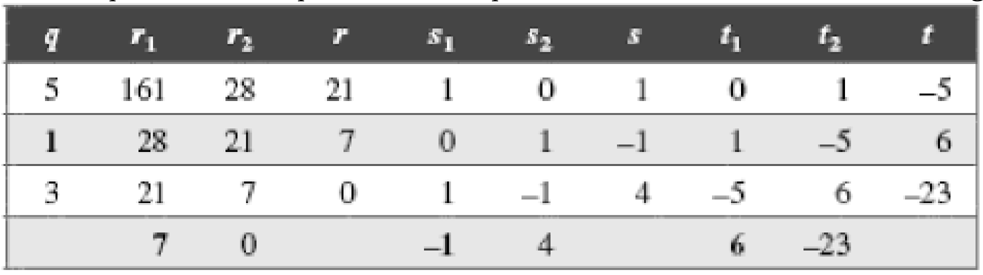
a. Process



```
$r_1 \leftarrow a;$  $r_2 \leftarrow b;$
$s_1 \leftarrow 1;$  $s_2 \leftarrow 0;$      (Initialization)
$t_1 \leftarrow 0;$  $t_2 \leftarrow 1;$
while ($r_2 > 0$)
{
  $q \leftarrow r_1 / r_2;$
    $r \leftarrow r_1 - q \times r_2;$
    $r_1 \leftarrow r_2;$  $r_2 \leftarrow r;$      (Updating r's)
    $s \leftarrow s_1 - q \times s_2;$
    $s_1 \leftarrow s_2;$  $s_2 \leftarrow s;$      (Updating s's)
    $t \leftarrow t_1 - q \times t_2;$
    $t_1 \leftarrow t_2;$  $t_2 \leftarrow t;$      (Updating t's)
}
gcd (a , b) $\leftarrow r_1;$   $s \leftarrow s_1;$   $t \leftarrow t_1$
```

b. Algorithm

Example: Given a = 161 and b = 28,
Find gcd (a,b) and the values of s and t.
Solution:
$r = r_1 - q \times r_2$, $t = t_1 - q \times t_2$, $s = s_1 - q \times s_2$, We use a table to follow the algorithm.



q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	–5
1	28	21	7	0	1	–1	1	–5	6
3	21	7	0	1	–1	4	–5	6	–23
	7	0		–1	4		6	–23	

We get gcd (161,28) = 7, s=-1 and t = 6

Linear Diophantine Equations

An equation of type ax + by = c with variables is called as Linear Diophantine Equation.
The Extended Euclidean algorithm is used to find solutions to the Linear Diophantine Equations
This type of equation has either no solution or an infinite number of solutions. Let d = gcd(a,b).
if d + c, then the equation has no solution.
If d | c, then we have an infinite number of solutions. (one is particular and rest are general soluti ons).
Particular Solution: if d | c, a particular solution to the above equation can be found using the following steps:

- Reduce the equation to $a_1x + b_1y = c_1$ by dividing both sides of the equation by d. This is possible because d divides a, b, and c by the assumption.
- Solve for s and t in the relation $a_1s + b_1t = 1$ using the extended Euclidean algorithm.
- The particular solution: $x_0 = (c/d)s$ and $y_0 = (c/d)t$

General Solutions: after finding the particular solution, the general solutions can be found:
$x = x_0 + k (b/d)$ and
$y = y_0 - k (a/d)$ where k is an integer

Example: Find the particular and general solutions to the equation

$$21x + 14y = 35.$$

Given equation, $21x+14y = 35$ that is written as $ax+by = c$

$a=21, b=14, c=35$

$d = \gcd(a,b) = \gcd(21,14)$ [Apply Euclidean Algorithm]
$= \gcd(14,7)$ 1.gcd(a,0) = a
$= \gcd(7,0)=7$ 2.gcd(a,b) =gcd(b,r)
so, $d=7$ where 'r' remainder

Note: if d | c i.e 7|35 (7 divides 35), so one is Particular solution and infinity General solutions.

Particula Solution :-

$$21x+14y=35 \quad ①$$

Divide both sides by 7 in ①, then

$$3x+2y=5 \quad ②$$

using Extended Euclidean Algorithm , find "s" and "t"
such as $3s+2t = 1$ Ref. (s x a + t x b = gcd (a,b))
Find gcd (3, 2) where r1 is 3 and r2 is 2 using Extended Euclidean Algorithm
$r = r1 - r2 \times q$, $s= s1 - s2 \times q$, $t= t1 - t2 \times q$

q	r1	r2	r	s1	s2	s	t1	t2	t
1	3	2	1	1	0	1	0	1	-1
2	2	1	0	0	1	-2	1	-1	3
x	1	0	x	1	-2	x	-1	3	x

gcd(3,2)=r1=1 s=s1=1 t=t1=-1

as per particular solutions
$x_0 = (c/d)s$ and $y_0 = (c/d)t$
substitute values a=21,b=14 , c=35, d=7 for x_0 and y_0

$x_0 = (35/7)x\ 1= 5$

$y_0 = (35/7)(-1)= - 5$

General Solution:
$x = x_0 + k (b/d)$ and $y = y_0 – k (a/d)$ where k is an integer
$x = 5+k(14/7)$; $y = -5-k(21/7)$
$x = 5+2k$ $y = -5-3k$
here "k" is an integer ; k=0,1,2,3,4… then substitute k in above:
(5,-5), (7,-8),(9,-11), ... are solutions to given equation

Modular Arithmetic

The division relationship (a = q x n + r) has two inputs (a and n) and two outputs (q and r). In modular arithmetic, we are focused in only one of the outputs, the remainder r.

Modulo Operator:

- Modulo operator is shown as *mod*.
- The second input (n) is called the modulus.
- The output r is called the residue.

The below figure shows the division relation compared to the modulo operator

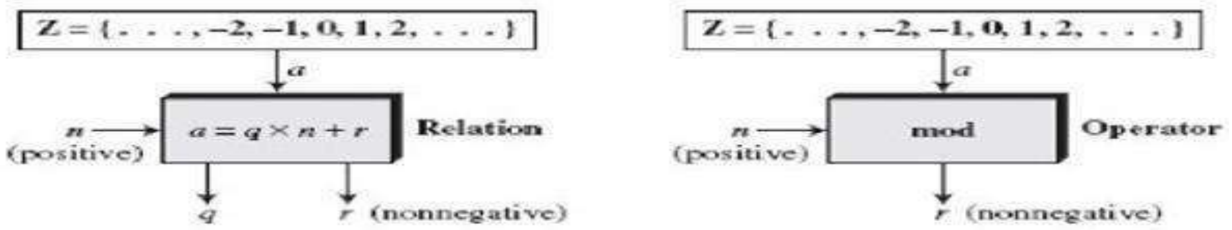


Fig. : *Division relation and modulo operator*

The modulo operator (mod) takes an integer (a) from the set Z and a positive modulus (n). The operator creates a non-negative residue (r).

a mod n = r

- Example

Find the results of the following operations:

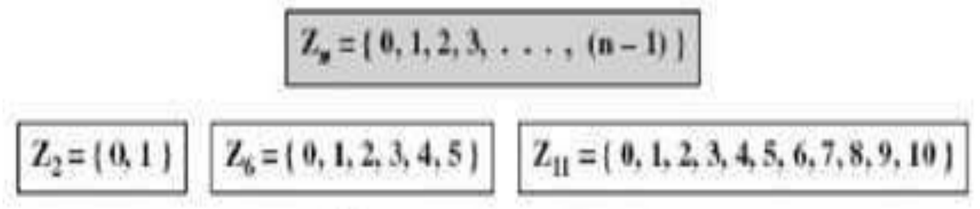
a. 27 mod 5
b. 36 mod 12
c. −18 mod 14
d. −7 mod 10

SOLUTION We are looking for the residue *r*. We can divide the *a* by *n* and find *q* and *r*. We can then disregard *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. However, we need to add the modulus (14) to make it nonnegative. We have *r* = −4 + 14 = 10. This means that −18 mod 14 = 10.
d. Dividing −7 by 10 results in *r* = −7. After adding the modulus to −7, we have *r* = 3. This means that −7 mod 10 = 3.

SET OF RESIDUES: Z_n

The result of the modulo operation with modulus ‘n’ is always an integer between 0 and n-1.
In other words (a mod n) is always a non negative integer less than n
Modulus operation creates a set, that is called set of least residues modulo n or Z_n
We have one set of Z(integers), but we have infinite instances of the set o residues Z_n for each n.



$Z_n = \{0, 1, 2, 3, \ldots, (n-1)\}$
$Z_2 = \{0, 1\}$
$Z_6 = \{0, 1, 2, 3, 4, 5\}$
$Z_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

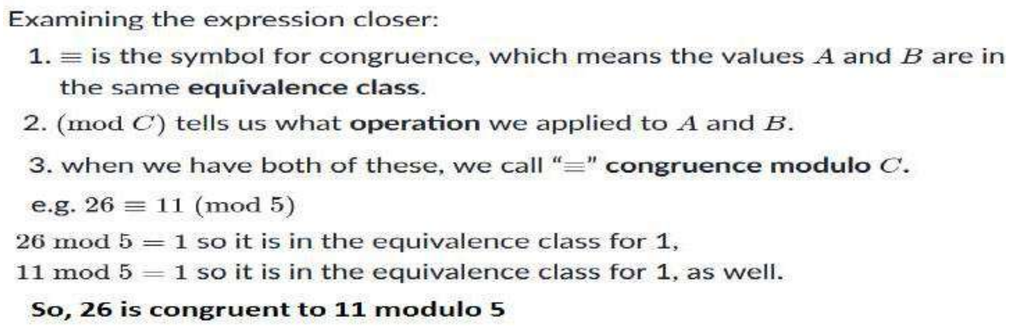
Fig. *Some Z_n sets*

CONGRUENCE (≡)

If two numbers A and B have the property that their difference A-B is integrally divisible by a number C (i.e., (A-B)/C is an integer), then A and B are said to be "congruent modulo C." The number C is called the modulus , and the statement "A is congruent to B (modulo C)" is written mathematically as

$$A \equiv B \pmod{C}$$

This says that " A is congruent to B modulo C".



Examining the expression closer:

1. $\equiv$ is the symbol for congruence, which means the values A and B are in the same **equivalence class**.
2. $(\mathrm{mod}\ C)$ tells us what **operation** we applied to A and B.
3. when we have both of these, we call "$\equiv$" **congruence modulo** C.

e.g. $26 \equiv 11 \pmod{5}$

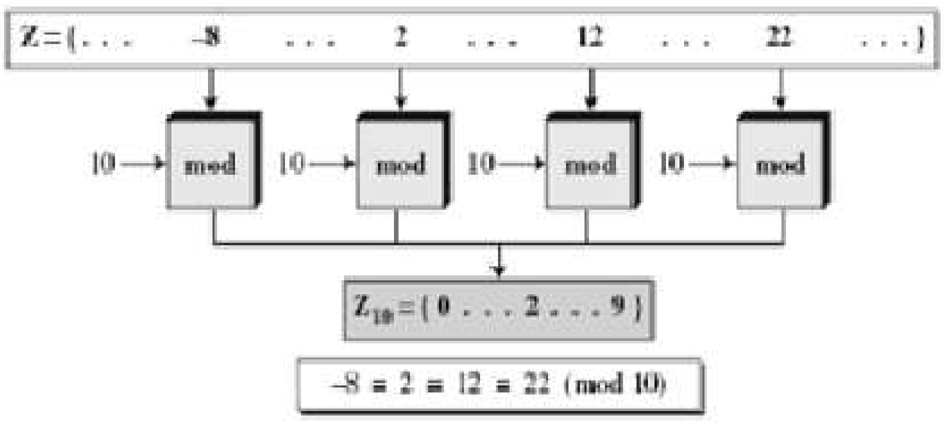
$26 \bmod 5 = 1$ so it is in the equivalence class for 1,

$11 \bmod 5 = 1$ so it is in the equivalence class for 1, as well.

So, 26 is congruent to 11 modulo 5

Example 2:

Assume, -8≡12(mod 10) 2≡12(mod 10) 12≡22(mod 10) 22≡32(mod 10)


Z = { . . . -8 . . . 2 . . . 12 . . . 22 . . . }
10
mod
10
mod
10
mod
10
mod
Z10 = { 0 . . . 2 . . . 9 }
-8 ≡ 2 ≡ 12 ≡ 22 (mod 10)

Congruence Relationship

⌖ <u>RESUDUE CLASSES</u>

A residue class [a] is the set of integers congruent 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 five sets [0], [1], [2], [3], [4] as shown below

[0]= { ...., -15 -10 ,-5, 0, 5, 10,15,...}

[1]= { ...., -16 -11 ,-6, 1, 6, 11,16,...}

[2]= { ...., -17 -12 ,-7, 2, 7, 12,17,...}

[3]= { ...., -18 -13 ,-8, 3, 8, 13,18,...}

[4]= { ...., -19 -14 ,-9, 4, 9, 14,19,...}

From each set there is one lease residue that

0 in [0], 1 in [1], 2 in [2], 3 in[3] and 4 in [4]..

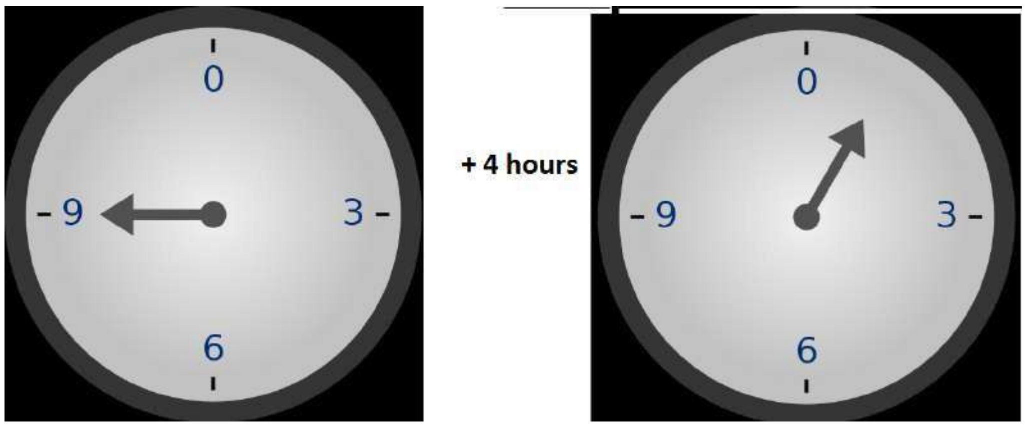
The set of these residues are shown as

Z_5 ={0,1,2,3,4}

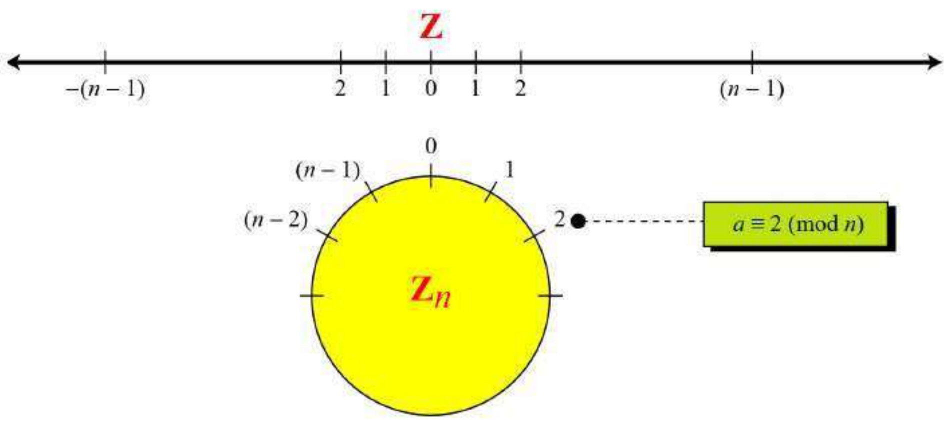
<u>Applications:</u>

We use a clock to measure time. Our clock system uses modulo 12 arithmetic. How ever instead of a 0 we the 12

.



Comparison of Z and Z_n using graphs



⌖ Operations in Z_n

The three Binary operations (addition, subtraction and multiplication) are defined for the set Z $_n$.

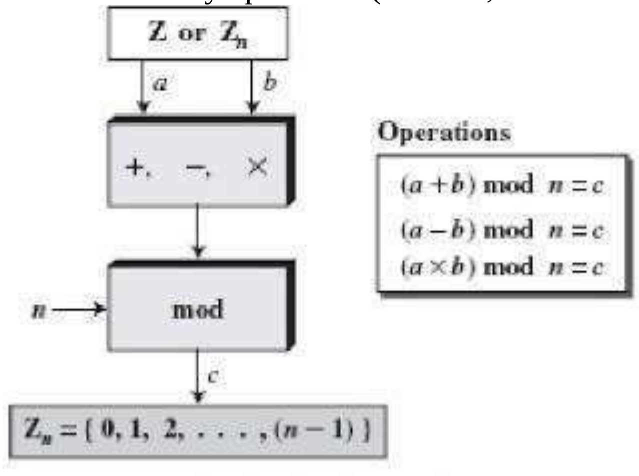


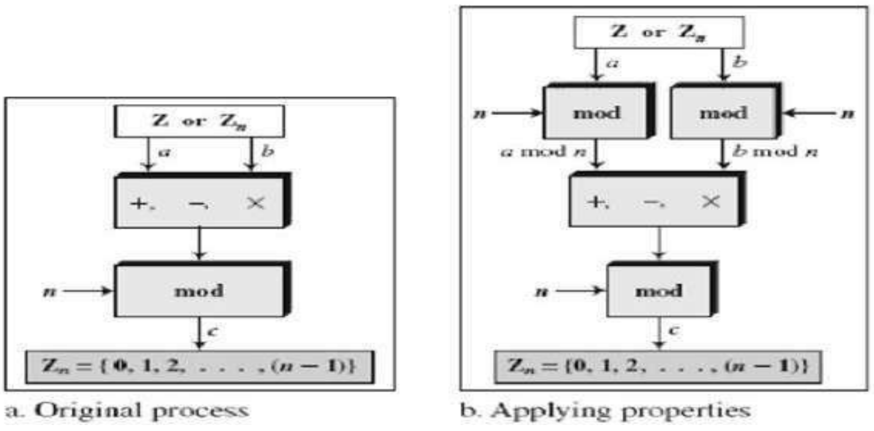
Fig. *Binary operations in Z_n*

Properties:

First Property: $(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$

Second Property: $(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$

Third Property: $(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$



EXAMPLE : Perform the following operations (the inputs come from Z_n):

a. Add 7 to 14 in $\mathbf{Z}_{15}$.
b. Subtract 11 from 7 in $\mathbf{Z}_{13}$.
c. Multiply 11 by 7 in $\mathbf{Z}_{20}$.

Solution:

$(14 + 7) \bmod 15 \rightarrow (21) \bmod 15 = 6$

$(7 - 11) \bmod 13 \rightarrow (-4) \bmod 13 = 9$

$(7 \times 11) \bmod 20 \rightarrow (77) \bmod 20 = 17$

<u>Example 2</u>

Perform the following operation:

a. Add 17 to 27 in Z_{14}
   (17+27) mod 14 = (44) mod 14 = 2
b. Subtract 34 from 12 in Z_{13}
   (12-34) mod 13 = (-22) mod 13 = - 9 = (-9+13) = 4
c. Multiply 123 by -10 in Z_{20}
   (123*(-10)) mod 20 = (-1230) mod 20 = -10 =(-10+20) = 10

Property 1:

(a+b) mod n= [(a mod n) + (b mod n)] mod n

(4+5) mod 2 = [(4 mod 2) + (5 mod 2)] mod 2

9 mod 2 = [0 + 1] mod 2

1 = 1

Property 2:

(a-b) mod n= [(a mod n) - (b mod n)] mod n

(4 - 5) mod 2 = [(4 mod 2) - (5 mod 2)] mod 2

-1 mod 2 = [0 - 1] mod 2

-1 mod 2 = -1 mod 2

Property 3:
(axb) mod n= [(a mod n) x (b mod n)] mod n (4 x 5) mod 2 = [(4 mod 2) x (5 mod 2)] mod 2 20 mod 2 = [0 x 1] mod 2
0 = 0 mod 2
0 = 0

INVERSES

When we are working in modular arithmetic, we need to find inverse of a number relative to an operation. There are two types of inverses are used modular arithmetic.

- Additive inverse (relative to an addition operation).
- Multiplicative inverse (relative to a multiplication operation).

❑ ***Additive Inverse*** In $\mathbf{Z}_n$, two numbers a and b are additive inverses of each other if

$$a + b \equiv 0 \pmod{n}$$

In $\mathbf{Z}_n$, the additive inverse of a can be calculated as $b = n - a$. For example, the additive inverse of 4 in $\mathbf{Z}_{10}$ is $10 - 4 = 6$.

Note:In modular arithmetic, each integer has an additive inverse.

- The sum of an integer and its additive inverse is congruent to 0 modulo n

❑ ***Multiplicative Inverse*** In $\mathbf{Z}_n$, two numbers a and b are the multiplicative inverse of each other if

$$a \times b \equiv 1 \pmod{n}$$

For example, if the modulus is 10, then the multiplicative inverse of 3 is 7. In other words, we have $(3 \times 7) \bmod 10 = 1$.

In modular arithmetic, an integer may or may not have a multiplicative inverse. When it does, the ! product of the integer and its multiplicative inverse is congruent to 1 modulo n.

It can be proved that 'a' has a multiplicative inverse in Zn iff gcd(n,a)=1. (In this case 'a' and n are said to **relatively prime**.

Example 1: Find multiplicative inverse of 8 in Z_{10}.

SOLUTION There is no multiplicative inverse because gcd (10, 8) = 2 ≠ 1. In other words, we cannot find any number between 0 and 9 such that when multiplied by 8, the result is congruent to 1.

Example 2: Find all multiplicative inverses in Z_{10}.

SOLUTION There are only three pairs: (1, 1), (3, 7) and (9, 9). The numbers 0, 2, 4, 5, 6, and 8 do not have a multiplicative inverse. We can see that

$$(1 \times 1) \bmod 10 = 1(3 \times 7) \bmod 10 = 1(9 \times 9) \bmod 10 = 1$$

The extended Euclidean algorithm finds the multiplicative inverses of b in Z_n
when n and b are given and gcd $(n, b) = 1$.
The multiplicative inverse of b is the value of t after being mapped to Z_n.

Example 3: Find all multiplicative inverses 23 in Z_{100}.

SOLUTION We use a table similar to the one we used before with $r_1 = 100$ and $r_2 = 23$. We are interested only in the value of t.

q	r_1	r_2	r	t_1	t_2	t
4	100	23	8	0	1	−4
2	23	8	7	1	−4	9
1	8	7	1	−4	9	−13
7	7	1	0	9	−13	100
	1	0		−13	100	

The gcd (100, 23) is 1, which means the inverse of 23 exists. The extended Euclidean algorithm gives $t_1 = -13$. The inverse is $(-13) \bmod 100 = 87$. In other words, 23 and 87 are multiplicative inverses in $\mathbf{Z}_{100}$. We can see that $(23 \times 87) \bmod 100 = 2001 \bmod 100 = 1$.

Addition and Multiplication Tables:

In addition table, each integer has an additive inverse. The inverse pairs can be found when the result of addition is zero. In Figure 2.16, we have (0,0), (1,9), (2,8), (3,7), (4,6), and (5,5).
In multiplication table, the pairs can be found whenever the result of multiplication is 1. In Figure, we have (1,1), (3,7) and (9,9).

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

Addition Table in $\mathbf{Z}_{10}$

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	0	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

Multiplication Table in $\mathbf{Z}_{10}$

Fig: Addition and multiplication tables in Z_{10}
Note: We need to use Z_n when additive inverses are needed; we need to use Z*n when multiplicative inverses are needed.

Two more Sets:
Cryptography often uses two more sets: Z_p and Z*p.

The set $\mathbf{Z}_p$ is the same as $\mathbf{Z}_n$ except that n is a prime. $\mathbf{Z}_p$ contains all integers from 0 to $p - 1$. Each member in $\mathbf{Z}_p$ has an additive inverse; each member except 0 has a multiplicative inverse.

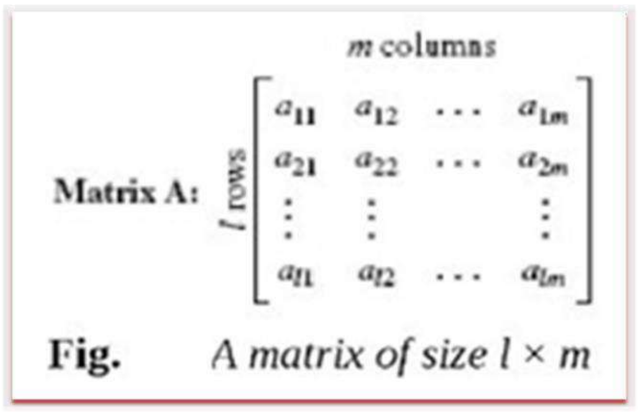
The set $\mathbf{Z}_{p*}$ is the same as $\mathbf{Z}_{n*}$ except that n is a prime. $\mathbf{Z}_{p*}$ contains all integers from 1 to $p - 1$. Each member in $\mathbf{Z}_{p*}$ has an additive and a multiplicative inverse. $\mathbf{Z}_{p*}$ is a very good candidate when we need a set that supports both additive and multiplicative inverse.

The following shows these two sets when $p = 13$.

$$Z_{13} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$
$$Z_{13}{}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

MATRICES

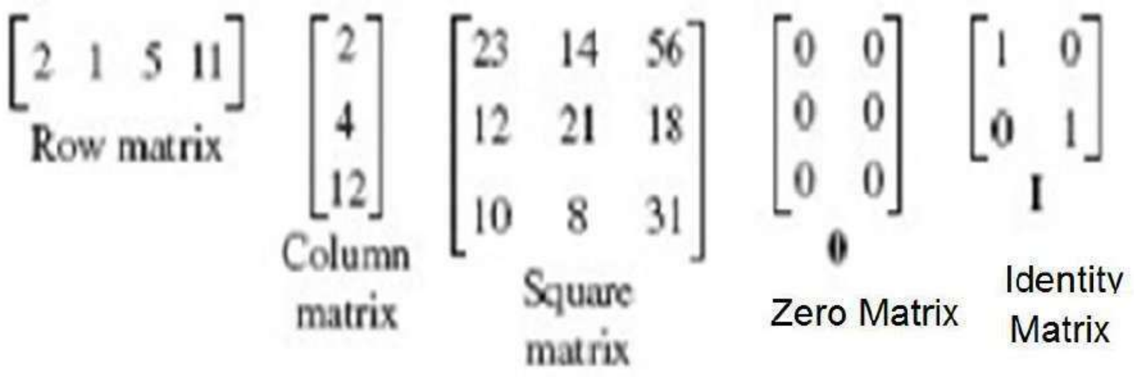
A matrix is a rectangular array of *l x m* elements; in which
l is the number of rows and
m is the number of columns.
A matrix is normally denoted with an Uppercase Letter such as A.
The element a_{ij} is located in the ith row and jth column.



$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{bmatrix}$$

Fig. *A matrix of size l × m*

DIFFERENT TYPES OF MATRICES



$$\begin{bmatrix} 2 & 1 & 5 & 11 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} \quad \begin{bmatrix} 23 & 14 & 56 \\ 12 & 21 & 18 \\ 10 & 8 & 31 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

OPERATIONS AND RELATIONS

Relation operation:

Equality:

If two matrices are equal sizde and content is same then they have equality

Four operations:

1. Addition
2. Subtraction
3. Multiplication
4. Scalar multiplication

Examples:

Addition : $C_{IJ}=A_{IJ}+ B_{IJ}$

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

$$\mathbf{C = A + B}$$

Subtraction: : $C_{IJ}=A_{IJ} - B_{IJ}$

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

$$\mathbf{D = A - B}$$

Multiplication

If each element of matrix **A** is called a_{ij}, each element of matrix **B** is called bj_k, then each element of matrix **C**, c_{ik}, can be calculated as

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

Examples:

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

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

$$\begin{array}{ccccc} \mathbf{C} & & \mathbf{A} & & \mathbf{B} \\ \begin{bmatrix} 52 & 18 & 14 & 9 \\ 41 & 21 & 22 & 7 \end{bmatrix} & = & \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 4 \end{bmatrix} & \times & \begin{bmatrix} 7 & 3 & 2 & 1 \\ 8 & 0 & 0 & 2 \\ 1 & 3 & 4 & 0 \end{bmatrix} \end{array}$$

❑ ***Scalar Multiplication*** We can also multiply a matrix by a number (called a **scalar**). If **A** is an $l \times m$ matrix and x is a scalar, $C = xA$ is a matrix of size $l \times m$, in which $c_{ij} = x \times a_{ij}$.

$$\begin{array}{c}\mathbf{B}\\ \begin{bmatrix}15 & 6 & 3\\ 9 & 6 & 12\end{bmatrix}\end{array} = 3 \times \begin{array}{c}\mathbf{A}\\ \begin{bmatrix}5 & 2 & 1\\ 3 & 2 & 4\end{bmatrix}\end{array}$$

Multiplication unit matrix with normal matrix gives the same matrix

A X I = I X A = A

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} X \begin{bmatrix}2 & 4\\ 3 & 1\end{bmatrix} = \begin{bmatrix}1X2+0X3 & 1X4+0X1\\ 0X2+1X3 & 0X4+1X1\end{bmatrix} = \begin{bmatrix}2 & 4\\ 3 & 1\end{bmatrix}$$

DETERMINANT

If A is square matrix of mxm then determinant of A is det(A)

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

Where A_{ij} is a matrix obtained from A by deleting the ith row and jth column.
Determinant is obtained for only square matrices

Det(2x2) matrix

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

$$=5x4-2x3=14$$

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

Example : det(3x3) matrix

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

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

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

$$+ \quad (+1) \times 1 \times (3)$$

$$= -25$$

MATRICES-Inverses

Additive Inverse

The additive inverse of the matrix A is another matrix B such that A+B=0.

In other words $b_{ij}=-a_{ij}$

Generally additive inverse is of A=-A

Multiplicative Inverse:

The multiplicative Inverse of a square matrix A is a B such that A X B = I.

Normally Multiplicative inverse of A is defined by A^{-1}

<u>Multiplicative inverse is defined for only square matrices</u>

<u>Residue Matrices</u>

Cryptography uses residue matrices: matrices with all elements are in $\mathbf{Z}_n$. 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 $\mathbf{Z}_n$. In other words, a residue matrix has a multiplicative inverse if gcd (det($\mathbf{A}$), n) = 1.

EXAMPLE Figure shows a residue matrix A in Z_{26} and its multiplicative inverse A^{-1}. We have det(A) = 21 which has the multiplicative inverse 5 in Z_{26}. Note that when we multiply the two matrices, the result is the multiplicative identity matrix in Z_{26}.

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

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

Fig. *A residue matrix and its multiplicative inverse*

Example : Find A^{-1} modulo value -

Problem:

$$\text{if } A = \begin{pmatrix} 3 & 2 \\ 4 & 7 \end{pmatrix} \bmod 36; \text{ then } A^{-1} \bmod 36 = ?$$

Solution:

$$A = \begin{pmatrix} 3 & 2 \\ 4 & 7 \end{pmatrix} \Rightarrow \det(A) = 3\times7 - 2\times4 = 13$$

$[\det(A)]^{-1} \bmod 36 = 13^{-1} \bmod 36 = ?$

$(13)^{-1} \bmod 36 = ? \Rightarrow$ use extended Euclidian Alg. to find $13^{-1} \bmod 36 = ?$

Calculate gcd (36, 13), 13 multiplicative inverse

$r = r_1 - q \times r_2 \qquad t = t_1 - q \times t_2$

$r_1 = 36;\ r_2 = 13,\ t_1 = 0;\ t_2 = 1$

q	r_1	r_2	r	t_1	t_2	t
2	36	13	10	0	1	-2
1	13	10	3	1	-2	3
3	10	3	1	-2	3	-11
3	3	1	0	3	-11	36
X	1	0	X	-11	36	X

gcd(13,36)=1 "13" multiplicative inverse

$13^{-1} \bmod 36 = -11 \bmod 36$

$= (-11+36) \bmod 36 = 25 \bmod 36$

'13' multiplicative inverse in Z_{36} $\Big\}$ $= \underline{25}$

$\Rightarrow [\det(A)]^{-1} \bmod 36 = 13^{-1} \bmod 36 = 25$

$$A^{-1} = 25\begin{bmatrix} 7 & -2 \\ -4 & 3 \end{bmatrix} \pmod{36}$$

$$= \begin{bmatrix} 175 & -50 \\ -100 & 75 \end{bmatrix} \pmod{36}$$

$$= \begin{bmatrix} 175 \bmod 36 & -50 \bmod 36 \\ -100 \bmod 36 & 75 \bmod 36 \end{bmatrix}$$

175 mod 36= 31
-50 mod 36=(72-50 mod 36) = 22
-100 mod 36=(108-100 mod 36)=8
75 mod 36=3

$$= \begin{bmatrix} 31 & 22 \\ 8 & 3 \end{bmatrix} \quad \text{so, } A^{-1} \bmod 36 = \begin{bmatrix} 31 & 22 \\ 8 & 3 \end{bmatrix}$$

Linear Congruence

Single variable Linear Equations:
Equations of the form ax ≡ b (mod n) might have no solution or a limited number of solutions
Assume that the gcd(a,n) = d.
If d + b (d not divides b), there is no solution.
If d | b (d divides b), there are d solutions.
If d | b, we use the following strategy to find the solutions:

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

Congruence-Example

Example 1: Solve the equation

10 x = 2(mod 15).

Solution :-
Given Linear equation 10x≡ 2(mod 15)
In basic form ax ≡ b(mod n)
a = 10 ; b = 2; n= 15
Now, find d = ?
d = gcd(a,n)= gcd (10,15)
= gcd (15,10) = gcd (10,5)
= gcd (5,0)
= 5
check if d+b (d not divides b), then no solution
5+2 means ‘5’ not divides ‘2’, so, The given equation has No solution.

Example 2: Solve the equation

14 x= 12 (mod 18)

Solution :- Given Linear equation
14x ≡ 12(mod 18)
In basic form ax ≡ b(mod n)
a = 14 ; b = 12; n= 18
d = gcd(a,n)= gcd (14,18) = gcd (18,14)
= gcd (14,4) = gcd (4,2)=gcd(2,0)=2
check, d b or d+ b
d | b → 2 | 12 means “ 2 divides 12”, so the given equation have “2 solutions”.

Given equation 14 x 12 (mod 18)
divides ‘d’ on both sides of equation
7x 6 (mod 9)
multiply 7^{-1} on both sides of above to get particular solution ‘x_0’.
$7^{-1} \times 7 * x_0 \equiv 6 * 7^{-1}$ (mod 9)
$x_0 \equiv 6x\ 7^{-1}$ (mod 9) i.e 7^{-1} mod 9 ≡ 4
$x_0 \equiv 6 \times 4$ (mod 9)
$x_0 \equiv 24$ mod 9
$x_0 \equiv 6$
solutions are $x = x_0 + k\,(n/d)$ where k = 0,1
(d = 2)
if k = 0 $x = x_0 + 0\,(n/d)$
x = 6+ 0 (18/2) = 6
x = 6
if k = 1 $x = x_0 + 1\,((n/d) = 6+1\,(18/2)$
x = 15
‘6’ and ‘15’ are solution to 14 x 12 (mod 18)

Set of Linear Equations:

Solve the set of linear equations with same modulus by forming three matrices using coefficients.
Matrix 1: square matrix made from coefficients
Matrix 2: Column matrix made from variables
Matrix 3: Column matrix made from values at right side of equations
Consider the matrix as

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array}$$

a. Equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

b. Interpretation

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

c. Solution

Example

Solve the following sets of Linear equations?
3x + 2y ≡ 5 (mod 7)
4x + 6y ≡ 4 (mod 7)

Solution:

Matrix format of above equations is

$\begin{bmatrix}3 & 2\\4 & 6\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}5\\4\end{bmatrix}$(mod 7)

$\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}3 & 2\\4 & 6\end{bmatrix}^{-1}\begin{bmatrix}5\\4\end{bmatrix}$(mod 7)

Let A= $\begin{bmatrix}3 & 2\\4 & 6\end{bmatrix}$ then A^{-1}= $\begin{bmatrix}3 & 2\\4 & 6\end{bmatrix}^{-1} = \frac{1}{10}\begin{bmatrix}6 & -2\\-4 & 3\end{bmatrix}$

$\begin{bmatrix}x\\y\end{bmatrix}=\frac{1}{10}\begin{bmatrix}6 & -2\\-4 & 3\end{bmatrix}\begin{bmatrix}5\\4\end{bmatrix}$(mod 7) = $\frac{1}{10}\begin{bmatrix}30-8\\-20+12\end{bmatrix}$(mod 7)

= $\frac{1}{10}\begin{bmatrix}22\\-8\end{bmatrix}$(mod 7) = $\begin{bmatrix}22/10\\-8/10\end{bmatrix}$(mod 7)

x= 22/10 (mod 7)

y= -8/10 (mod 7)

Check the answer by inserting values:

3x + 2y = 5 (mod 7)

3(22/10) + 2(-8/10)=(66/10)-(16/10)=5 (mod 7)

4x + 6y = 4 (mod 7)

4(22/10) + 6(-8/10)=(88/10)-(48/10)=4 (mod 7)

LINEAR DIOPHANTINE EQUATION

The equation of the form ax +by = c is called as **Linear Diophantine Equation.**

Example: 19x +13y = 20

We can solve the above equation using following steps:

Step1: check whether it has solution or not.

Perform gcd(a,b) and if gcd(a,b) divides c then it has solution

Step 2: Use Euclidian Algorithm and reverse Euclidian algorithms to find the Particular solution.. x_0, y_0

Step3: Find general solution

$x=x_0+b.n$ where n is any integer

$y=y_0-a.n$

Example 1:-

Find particular and General solutions to the following **Linear Diophantine Equation:**

25x+10y=15

Solution:

a=25, b=10 and c=15

Check whether we have solution or not by calculating GCD(25,10) using Euclidian Algorithm:

gcd(25,10)=gcd(10,5)=gcd(5,0)=5

since gcd(25,10)=5 that divides the 15, we have solutions.

Reverse Euclidian Algorithm is:

25 = 2 x 10 + **5** ---------Eq1

10 = 2 x 5 + 0

Since gcd is 5 , rewrite the Eq1 from write to left:

5= 25 – 2 x10

5= 1x25 – 2x10 (1x25 – 2x10 is similar to 25x + 10y)

Multiply both sides by 3 since in the given equation right hand side is 15

3x5=3(1x25)-3(2x10)

15 = 3 x 25 – 6 x 10 then it can be rewrite as

25x3 – 10x6 = 15

The above is similar to

25x + 10y = 15

So, the **particular soluti**on is

x_0 = 3 and y_0 = -6

Substitute the x0 and y0 in the above $25x_0 + 10y_0$ = 25x3+10(-6)=75-60=15

Now, find **General solution**:

$x=x_0+b.n$ (where n is any integer and a=25, b=10)
$y=y_0-a.n$

- **if n=1, then**
  x=3+10x1=13
  y=-6-25x1=-31

Substitute the x and y in the given equation to check result:
25x + 10y = 25x13+10(-31)=325-310=15

- **if n= -1, then**
  x=3+10x(-1)=3-10= -7
  y=-6-25x(-1)= -6 +25 = 19

Substitute the x and y in the given equation to check result:
25x + 10y = 25x(-7)+10(19)=-175+190=15

Finally, the x,y pairs are (-7,19), (3,-6), (13,-31),….

Example 2:

Find particular and General solutions to the following **Linear Diophantine Equation:**
19x+13y=20

Solution:
A=19, b=13 and c=20
Check whether we have solution or not by calculating GCD(19,13) using Euclidian Algorithm:
gcd(19,13)=gcd(13,6)=gcd(6,1)=gcd(1,0)=1
since gcd(19,13)=1 that divides the 20, we have solutions.
Reverse of Euclidian Algorithm is :
19= 1 x13+ 6--------------**(Eq 1)**
13= 2 x 6 + **<u>1</u>** ------------- **(Eq 2)**
6 = 6x1 + 0
So, gcd(19,13) is 1.

- Rewrite the Eq 2 from write to left:
  1 = 13 – 2 x 6 -------------**(Eq 3)**
- Rewrite the Eq 1 from write to left:
  6 = 19 – 1 x13 -------------**(Eq 4)**

Substitute the 6 equivalent in Eq 4 that is 19-1x13 in Eq 3
1= 13- 2 x (19-1x13)
1= 13-2x19 + 2 x13
1= 1x13-2x19 + 2 x13
1= 3x13-2x19
Multiply both sides by 20 since in the given equation right hand side is 20
20x1=60x13-40x19)
20 = 13x60 – 19 x 40 then it can be rewrite as
This can be rewrite as similar to the given equation 19x+13y=20
13x60-19x40 = 20
-19x40 + 13x 60 =20
19x(-40)+13x60=20
So, the **particular soluti**on is
x_0 = -40 and y_0 = 60
Substitute the x_0 and y_0 in the given equation to check the result
19x+13y = 19(-40) + 13x60 = -760+780 = 20
Now, find **General solution**:
$x=x_0+b.n$ (where n is any integer and a=19, b=13)
$y=y_0-a.n$

- **if n=1, then**

x=-40+13x1=-40+13= -27
y=60 –19x1=41

Substitute the x and y in the given equation to check result:
19x + 13y = 19(-27)+13(41)=-513+533=20

- **if n= -1, then**
  x=-40+13x(-1)=-40-13= -53
  y=60-19x(-1)= 60+19 = 79

Substitute the x and y in the given equation to check result:
19x + 13y = 19x(-53)+13(79)=-1007+1027=20

Finally, the x,y pairs are (-40,60), (-27,41), (-53,79),….

UNIT -II
Symmetric Encryption
Mathematics of Symmetric Key Cryptography, Introduction to Modern Symmetric Key Ciphers,
Data Encryption Standard, Advanced Encryption Standard.

Mathematics of Symmetric Key Cryptography:

Cryptography ?

Cryptography is a technique of securing information and communications through use of codes. Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”.

Cryptography Types

1) **Symmetric Key Cryptography:**
The sender and receiver of message use a single common key to encrypt and decrypt messages.
2) **Asymmetric Key Cryptography:**
A pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key.
3) **Hash Functions:**
There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered..

Mathematics of Symmetric Key Cryptography

Algebraic Structures:

Cryptography requires set 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**.

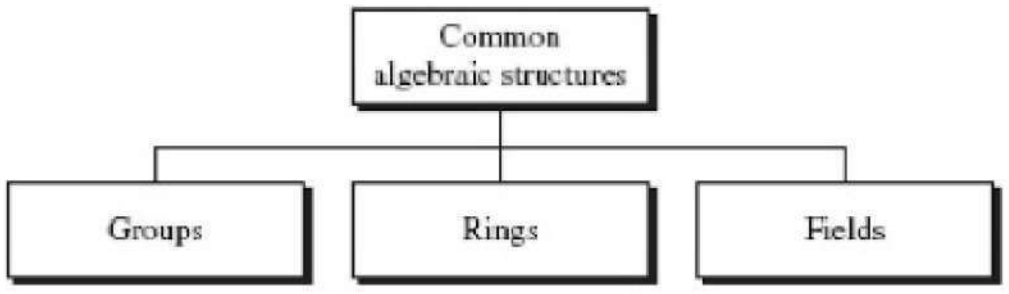


Fig. *Common algebraic structures*

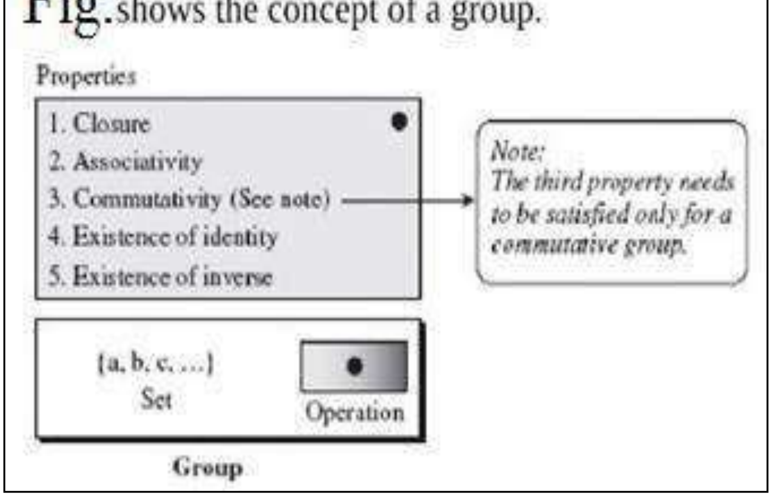
1. Groups

A **Group (G)** is a set elements with a binary operation “•” usually Addition or multiplication that satisfies four properties(Axioms).
• 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.

- Closure Property: if a and b are elements of G, then c = a • b is also an element of G.
- Associatively Property: if a, b, and c are elements of “G, then (a • b) • c = a • (b • c).
- Existence of Identity Property: For all a in G, there exists an element e, called the identity element, such that e • a = a • e = a

- Existence of Inverse Property: For each a in G, there exists an element a^1, called the inverse of a, such that $a \bullet a^1 = a^1 \bullet a = e$
- Commutativity Property: For all a and b in G, **we have a • b = b • a.**

Fig.shows the concept of a group.



EXAMPLE 1
The set of residue integers with the addition operator, G=< Z_n , + >, is a commutative group
1. Closure is satisfied. The result of adding two integers in Z_n is another integer in Zn
2.Associativity is satisfied. The result of 4+(3+2) is same as (4+3) + 2
3. Commutative is satisfied. we have 3+4=4+3
4.The identity element is 0. we have 3+0=0+3=3
5. Every element has an additive inverse. The inverse of 3 is 7 (3+7 mod 10 =0 mod 10 in Z_{10}) and inverse of 7 is 3(7+3 mod 10 =0 mod 10 in Z_{10}), so inverse property satisfied
EXAMPLE 2
The set Z_n* with multiplication operator, G=<Z_n*, x >, is also an abelian group. We can perform multiplication and divisions on the elements. We an identity element as 1.
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 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.

Subgroups: A subset **H** of a group **G** is a subgroup of G if H itself is a group with respect to the operation on **G**. In other words, if **G = <S, • >** is a group, **H = <T, • >** 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 • b is also a member of both groups
2. The group 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 with the identity element of G,H=<{e}, • >, is a sub group of G
5. Each group is a subgroup of itself

Cyclic Subgroup: If a subgroup of a group can be generated using the power of an element, the subgroup is called the cyclic subgroup.
The term power means repeatedly applying the group operation to the element:
a^n -> a.a.a.a.....a (n times)
Example: The group G=< Z3, + > contains cyclic subgroups for 0,1 and 2:
If generated using 0:
0^0 mod 3 = 0, 0^1 mod 3 = 0, 0^2 mod 3 = 0. so, H1=<{0}, +>

If generated using 1:
1^0 mod 3 = 0, 1^1 mod 3 = 1, 1^2 mod 3 = (1+1) mod 3=2. so, H2=G
If generated using 2:
2^0 mod 3 = 0, 2^1 mod 3 = 2, 2^2 mod 3 = (2+2) mod 3=1. so, H3=G
Cyclic Group: A Cyclic group is a group that is its own cyclic subgroup. The element that generates cyclic subgroup can also generates group itself.This element is referred as generator 'g'.
Example: In the previous example, The group G=<Z3, +> is a cyclic grop with two generators g=1 and g=2

Lagrange's Theorem:

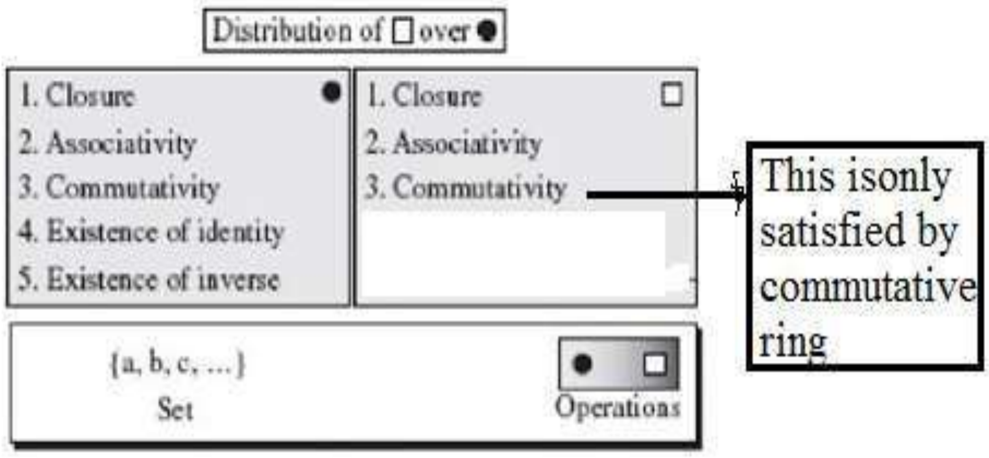
It related the order of a group to the order of its sub group. Assme that G is group and H is its subgroup. If order of G and H are |G| and |H|, respectively, based on this theorem |H| divides |G|.
EXAMPLE: As per the previous cyclic subgroup example, |H1|=1, |H2|=3, |H3|=3,
Obviously, all of these orders divide the order of |G|.

❑ ***Order of an Element*** The **order of an element** a in a group, ord(a), is the smallest integer n such that $an = e$. The definition can be paraphrased: the order of an element is the order of the cyclic group it generates.
Example:
In the group G=<Z3, +>, ord(0)=1, ord(1)=3, ord(2)=3

2. RING

A **Ring**, denoted as **R = < {....}, •** , □ >, is an algebraic structure with two operations(addition and multiplication).
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 operation.
Distributivity means that for all a, b and c elements of **R**, we have
a □ (b • c) = (a □ b) • (a □ c) and **(a • b) □ c = (a □ c) • (b □ c)**

Commutative Ring: If a ring satisfies commutative property, then we say the ring is a *commutative ring*.
- **Rings do not need to have a multiplicative inverse.**



Example: Z an Integer set is a Ring structure.
Explain why Z (set of Integer numbers) is a ring?
Suppose that 2,3,4∈Z.
- Both addition and multiplication are associative since

2+(3+4)=(2+3)+4, and 2(3x4)=(2x3)*4*.

- It follows that

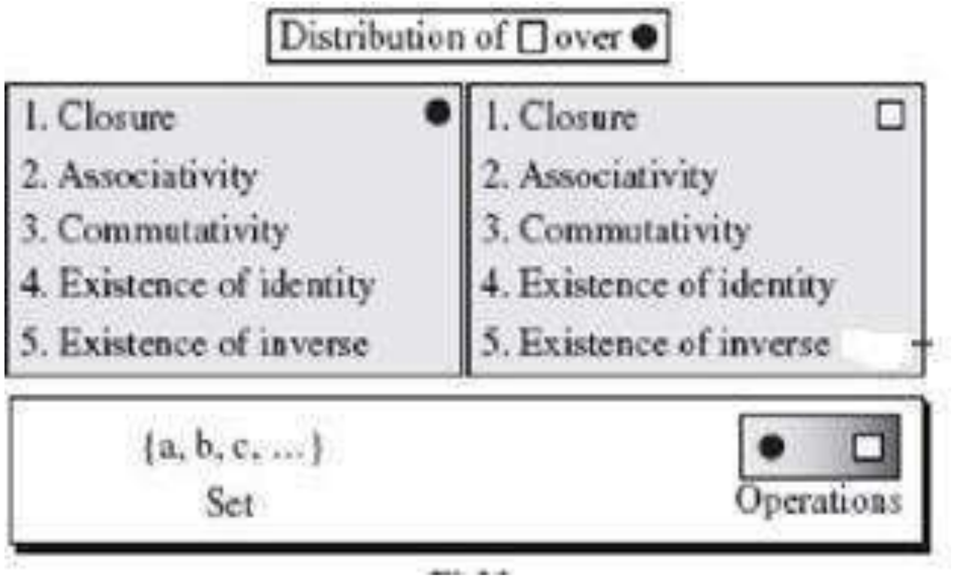
The identity element for addition is 0 since, 2+0=2.
The identity element for multiplication is 1 since 1x2=2.

- Addition is commutative too since 2+3=3+2

Multiplication is also commutative since 2x3=3x2,
so, Z can be called a *commutative ring*).
Addition has the inverse of -2 since 2+(−2)=0
(Note that multiplication does not need to have a multiplicative inverse. Because multiplicative inverse of 2 is ½. It is not an integer.
Lastly, multiplication also distributes over addition, that is 2(3+4)=2x3+2x4.
Rings do not need to have a multiplicative inverse.

3.FIELDS

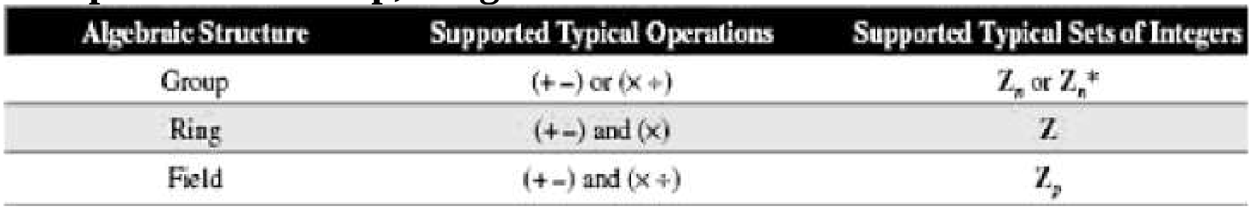
A field, denoted by **F = < {…}, • , □ >,** is a commutative ring in which first and second operations satisfies all five properties.
In other words:
A field is a set with the two binary operations of addition and multiplication, both of which operations are commutative, associative, contain identity elements, and contain inverse elements.
The identity element for addition is 0, and the identity element for multiplication is 1.
Application: A field is a structure that supports two pairs of operations: addition/subtraction and multiplication/division



FIELDS-Example

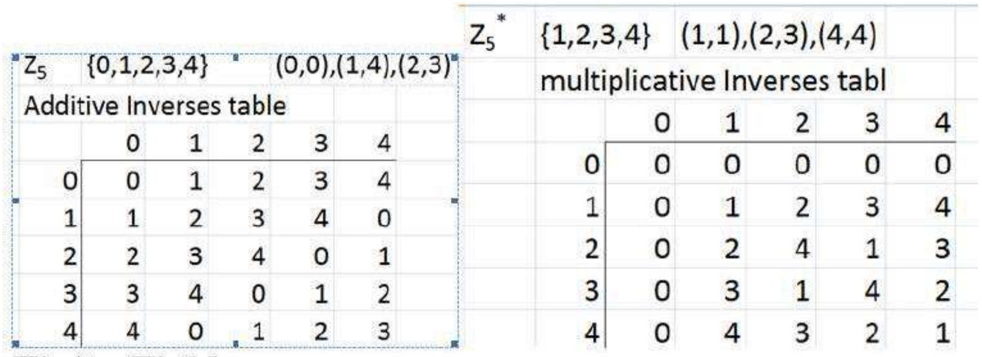
Explain why R is a field.(R is set of real numbers) :Suppose that *a*,*b*,*c*,*d*∈R. We know that R has addition and multiplication as binary operations since (*a*+*b*)=*c* for some c, and *ab*=*d* for some d. Furthermore, we know that addition and multiplication defined on real numbers is both commutative and associative.
Additionally, the identity element for addition is 0, *x*+0=*x*, and the identity element for multiplication is 1, since 1*x*=*x*.
Lastly, the inverse element for addition is -x, since *x*+(−*x*)=0 (0 being the identity for addition), and the inverse element for multiplication 1/x since *x*⋅1/*x*=1 when x ≠ 0.

Comparison of Group, Ring and Field:



Algebraic Structure	Supported Typical Operations	Supported Typical Sets of Integers
Group	(+ −) or (× ÷)	Z_n or Z_n^*
Ring	(+ −) and (×)	Z
Field	(+ −) and (× ÷)	Z_p

Check whether Z_p is field structure or not?



Z_5	{0,1,2,3,4}	(0,0),(1,4),(2,3)			
Additive Inverses table					
	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Z_5^*	{1,2,3,4}	(1,1),(2,3),(4,4)			
multiplicative Inverses tabl					
	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

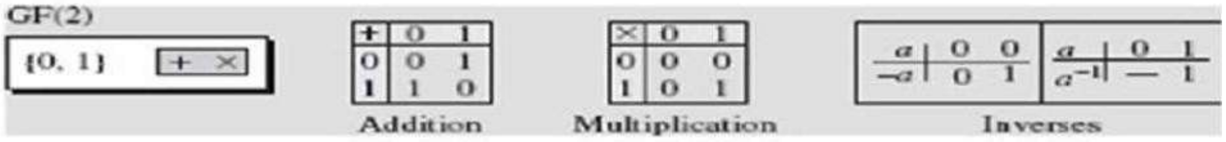
Finite Fields:

A finite field, a field with a finite number of elements. The finite fields are usually called **Galois fields** and denoted as **GF(p^n)**.

Note: A Galois field, **GF(p^n)**, is a finite field with p^n elements where p is prime.

GF(p) Fields: When n=1, we have GF(p) field. Tis field can be the set Z, (0,1,2,p-1), with two operations addition and multiplication. Each element has an additive inverse and that nonzero elements have a multiplicative inverse for prime p.

Example for GF(p) Field: A very common field in this category is GF(2) with the set {0,1} and two operations addition and multiplication a shown below:



+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

a	0	1
$-a$	0	1

a	0	1
a^{-1}	—	1

Fig. *GF(2) field*

NOTE: Addition is same as to XOR and multiplication is AND operation

GF(2^n) Fields:-

GF(2^n) is a Finite Field with 2^n elements. The elements in this set are n-bit words. For example, if n = 3, the set is: { 000, 001, 010, 011, 100, 101, 110, 111 }

Example: if n = 2, then GF(2^2) field in which the set has four 2-bit words:
{ 00, 01, 10, 11 }.

Why GF(2^n)?

Generally computer stores positive integers as n-bit words, can be 8-bit, 16-bit, 32-bit, 64-bit.
This means that range of words(integers) is 0 to 2^n-1. So, the modulus is 2^n
We have two choices if we use a field structure 1) using GF(p) or GF(2 ^{n})

1) If we use GF(p) with the set Z_p, where p is the largest prime number less than 2^n. This is ineffiecient, if we use integers from p to 2^n-1.

If n=3, the largest prime less than 2^3 is 7. This means that we can not use integer 7,8.

2) If we GF(2^n) with the set 2^n elements. The elements in this set are n-bit words. Example: If n=3, , the set is {000,001,010,011,100,101,110,111}

Polynomials

The data is shown as n-bit words in the computers that satisfy the properties in GF(2^n) . These n-bit words are easily represented by Polynomial of degree n-1.
A polynomial of degree n-1 is an expression of the form: Where x^i is called the i_{th} term and a_i is called coefficient of the i_{th} term.

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x^1 + a_0x^0$$

EXAMPLE Figure shows how we can represent the 8-bit word (10011001) using a polynomials.

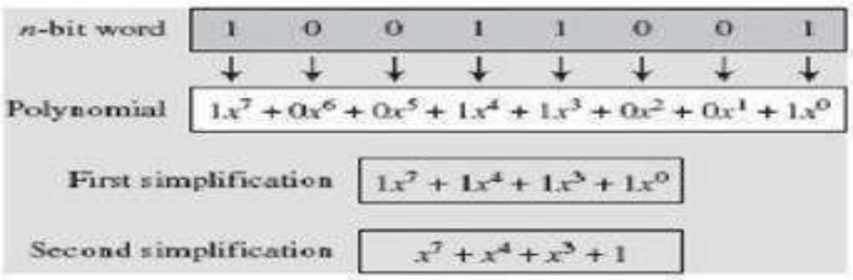


Fig. · *Representation of an 8-bit word by a polynomial*

EXAMPLE To find the 8-bit word related to the polynomial $x^5 + x^3 + x$, we first supply the omitted terms. Since n = 8, it means the polynomial is of degree 7. The expanded polynomial is

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

This is related to the 8-bit word 00100110.

Note: Polynomials representing n-bit words use two fields: GF(2) for Coefficients and GF(2^n) for terms.

Modulus:

Addition of two polynomials never creates a polynomial out of the set. However, multiplication of two polynomials may create a polynomial with degrees more than **n-1**. This means that we need to divide the result by a modulus and keep only the remainder.
A ***Prime Polynomial*** cannot be factored into a polynomial with degree of less than **n**. Such polynomials are referred to as ***Irreducible polynomials***.

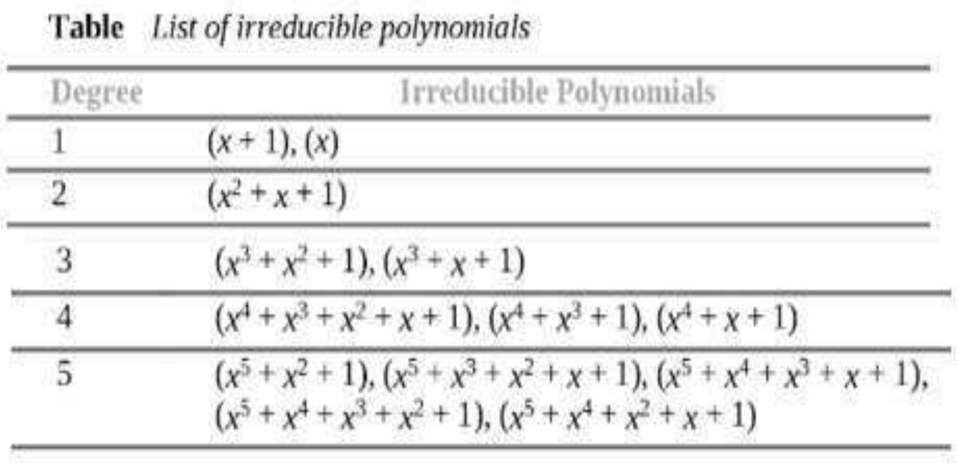


Table *List of irreducible polynomials*

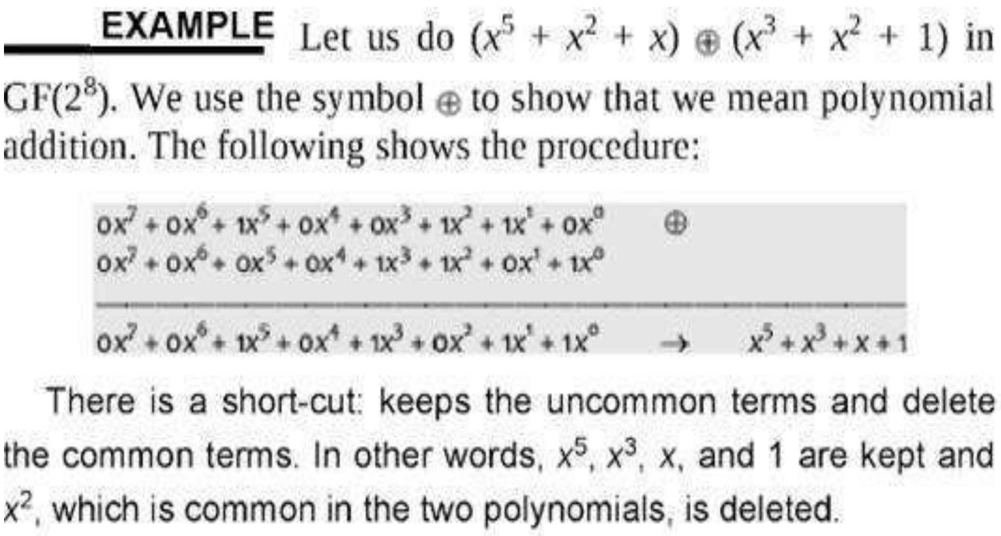
Degree	Irreducible Polynomials
1	$(x+1), (x)$
2	(x^2+x+1)
3	$(x^3+x^2+1), (x^3+x+1)$
4	$(x^4+x^3+x^2+x+1), (x^4+x^3+1), (x^4+x+1)$
5	$(x^5+x^2+1), (x^5+x^3+x^2+x+1), (x^5+x^4+x^3+x+1),$ $(x^5+x^4+x^3+x^2+1), (x^5+x^4+x^2+x+1)$

Operations on Polynomials: Addition:
Addition and Subtraction operations on polynomials are the same operation.
The addition operation for polynomials with coefficient in GF(2) is add the coefficients of the corresponding term in GF(2).
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.

Additive Identity: The additive identity in a polynomial is a zero polynomial (a polynomial with all coefficients set to zero).

Additive inverse: The additive inverse of a polynomial with coefficients in GF(2) is the polynomial itself. This means that the subtraction operation is the same as the addition operation.

Polynomials-Addition



EXAMPLE Let us do $(x^5 + x^2 + x) \oplus (x^3 + x^2 + 1)$ in GF(2^8). We use the symbol $\oplus$ to show that we mean polynomial addition. The following shows the procedure:

$$\begin{array}{l}
0x^7 + 0x^6 + 1x^5 + 0x^4 + 0x^3 + 1x^2 + 1x^1 + 0x^0 \quad \oplus \\
0x^7 + 0x^6 + 0x^5 + 0x^4 + 1x^3 + 1x^2 + 0x^1 + 1x^0 \\
\hline
0x^7 + 0x^6 + 1x^5 + 0x^4 + 1x^3 + 0x^2 + 1x^1 + 1x^0 \quad \rightarrow \quad x^5 + x^3 + x + 1
\end{array}$$

There is a short-cut: keeps the uncommon terms and delete the common terms. In other words, x^5, x^3, x, and 1 are kept and x^2, which is common in the two polynomials, is deleted.

Polynomials- Multiplication

- •Multiplication in polynomials is the sum of the multiplication of each term of the first polynomial with each term of the second polynomial.
- • The multiplication may create terms with degree more than **n-1**, which means the result needs to be reduced using a modulus polynomial

EXAMPLE Find the result of $(x^5 + x^2 + x) \otimes (x^7 + x^4 + x^3 + x^2 + x)$ in $GF(2^8)$ with irreducible polynomial $(x^8 + x^4 + x^3 + x + 1)$. Note that we use the symbol $\otimes$ to show the multiplication of two polynomials.

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

$$P_1 \otimes P_2 = x^{12}+x^9+x^8+x^7+x^6+x^9+x^6+x^5+x^4+x^3+x^8+x^5+x^4+x^3+x^2$$

$$P_1 \otimes P_2 = (x^{12}+x^7+x^2) \bmod (x^8+x^4+x^3+x+1) = x^5+x^3+x^2+x+1$$

Division by modulus to reduce Polynomial

$$\begin{array}{r|l} & x^4+1 \\ \hline x^8+x^4+x^3+x+1 & x^{12}+x^7+x^2 \\ & x^{12}+x^8+x^7+x^5+x^4 \\ \hline & x^8+x^5+x^4+x^2 \\ & x^8+x^4+x^3+x+1 \\ \hline \text{Remainder} & \boxed{x^5+x^3+x^2+x+1} \end{array}$$

Fig. *Polynomial division with coefficients in $GF(2^8)$*

Multiplicative identity & Multiplicative inverse

Multiplicative identity:- The multiplicative identity is always 1. For example, in $GF(2^8)$, the multiplicative inverse is the bit pattern 00000001

Multiplicative inverse:- Use extended Euclidean Algorithm to find the multiplicative inverse of a polynomial. This process is exactly same as for integers.

Example: In GF(24), find the inverse of (x2+1) modulo (x4+x+1)

Solution:- Use the extended Euclidean algorithm as in Table:

q	r_1	r_2	r	t_1	t_2	t
(x^2+1)	(x^4+x+1)	(x^2+1)	(x)	(0)	(1)	(x^2+1)
(x)	(x^2+1)	(x)	(1)	(1)	(x^2+1)	(x^3+x+1)
(x)	(x)	(1)	(0)	(x^2+1)	(x^3+x+1)	(0)
	(1)	(0)		(x^3+x+1)	(0)	

This means that $(x^2 + 1)^{-1}$ modulo $(x^4 + x + 1)$ is $(x^3 + x + 1)$. The answer can be easily proved by multiplying the two polynomials and finding the remainder when the result is divided by the modulus.

$$[(x^2 + 1) \otimes (x^3 + x + 1)] \bmod (x^4 + x + 1) = 1$$

Polynomials- Multiplication using a computer

If we multiply two polynomials, we also need to perform division operation that reduces an efficiency. Computer uses an algorithm for multiply the polynomials that should not use division operation, instead repeatedly multiplying a reduced polynomial by x.

Example: Instead of finding the result of $x^2 \otimes P_2$, it can done like
$x \otimes (x \otimes P_2)$

Example:

Power	Operation	New Result	Reduction
$x^0 \otimes P_2$		$x^7+x^4+x^3+x^2+x$	No
$x^1 \otimes P_2$	$x\otimes (x^7+x^4+x^3+x^2+x)$	x^5+x^2+x+1	Yes
$x^2 \otimes P_2$	$x\otimes (x^5+x^2+x+1)$	$x^6+x^3+x^2+x$	No
$x^3 \otimes P_2$	$x\otimes (X^6+x^3+x^2+x)$	$x^7+x^4+x^3+x^2$	No
$x^4 \otimes P_2$	$x\otimes (x^7+x^4+x^3+x^2)$	x^5+x+1	Yes
$x^5\otimes P_2$	$x\otimes (x^5+x+1)$	x^6+x^2+x	No

Result is P1xP2= $(x^6+x^2+x) + (x^6+x^3+x^2+x)+ (x^5+x^2+x+1)=x^5+x^3+x^2+x+1$

Simple algorithm

1. If the most significant bit of the previous result is 0, just shift the previous result one bit to the left.
2. If the most significant bit of the previous result is 1.

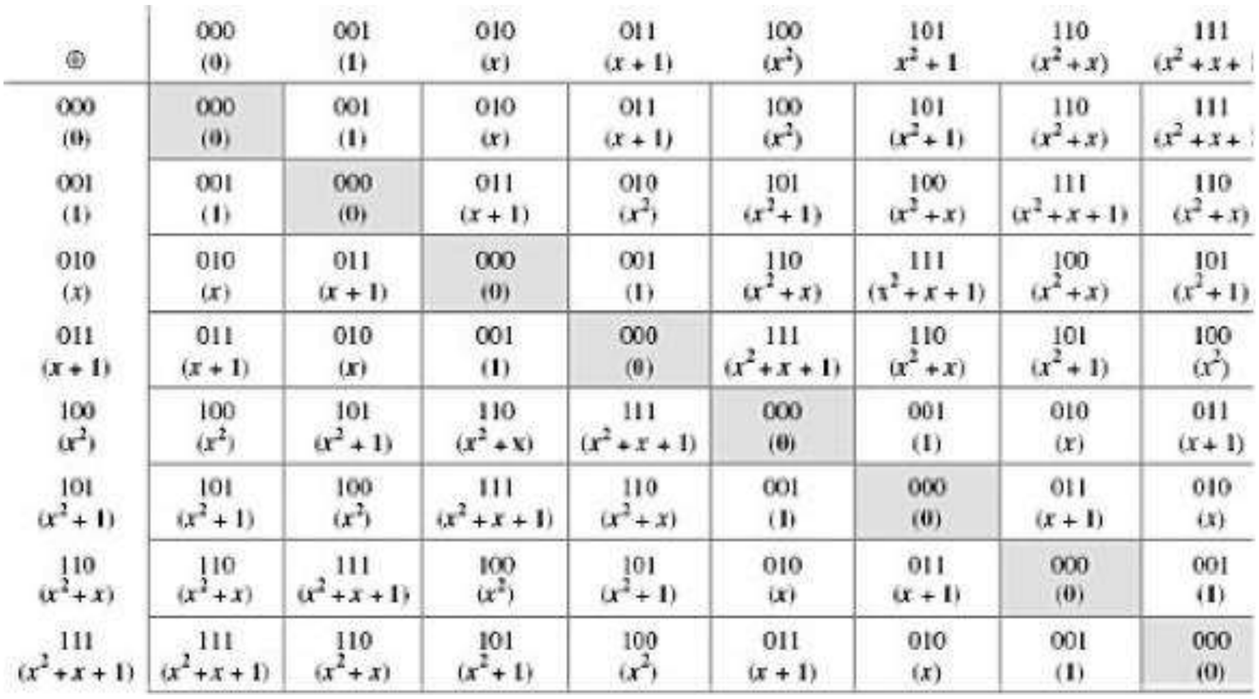
a)Shift it one bit to the left, and
b)Exclusive-OR it with the modulus without the most significant bit.

Example:Multiply P_1= (x^5+x^2+x) by $P_2=(x^7+x^4+x^3+x^2+x)$ in $GF(2^8)$ with irreducible $(x^8+x^4+x^3+x+1)$
Binary representation of P2=10011110,
Irreducible polynomial = 100011011(9bits)

Power	Shift-left Operation	Exclusive-OR
$x^0 \otimes P_2$		10011110
$X^1 \otimes P_2$	111100	(00111100)+(00011011)=00100111
$X^2 \otimes P_2$	1001110	1001110
$X^3 \otimes P_2$	10011100	10011100
$X^4 \otimes P_2$	111000	(00111000)+(00011011)= 00100011
$X^5\otimes P_2$	01000110	01000110
$P_1\otimes P_2$ = (00100111) + 01001110+01000110= 00101111		

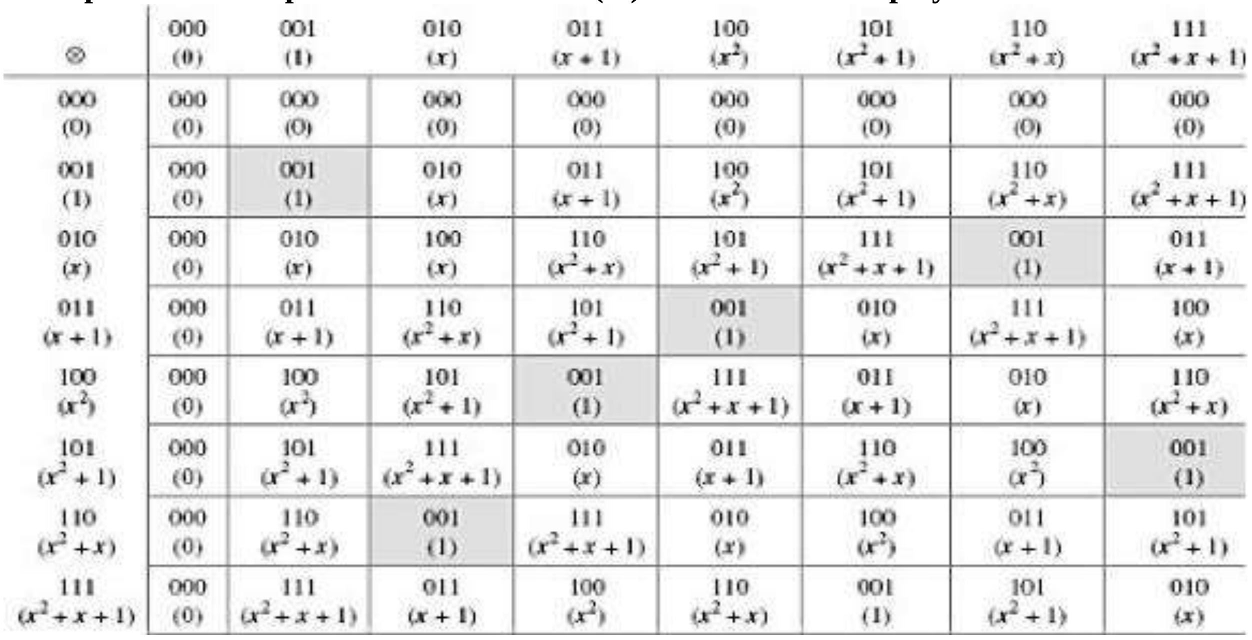
Multiplication of polynomials in $GF(2^n)$ can be achieved using shift-left and exclusive-or operations

Example:Find Addition Table for $GF(2^3)$ -



⊕	000 (0)	001 (1)	010 (x)	011 $(x+1)$	100 (x^2)	101 x^2+1	110 (x^2+x)	111 $(x^2+x+$
000 (0)	000 (0)	001 (1)	010 (x)	011 $(x+1)$	100 (x^2)	101 (x^2+1)	110 (x^2+x)	111 $(x^2+x+$
001 (1)	001 (1)	000 (0)	011 $(x+1)$	010 (x^2)	101 (x^2+1)	100 (x^2+x)	111 (x^2+x+1)	110 (x^2+x)
010 (x)	010 (x)	011 $(x+1)$	000 (0)	001 (1)	110 (x^2+x)	111 (x^2+x+1)	100 (x^2+x)	101 (x^2+1)
011 $(x+1)$	011 $(x+1)$	010 (x)	001 (1)	000 (0)	111 (x^2+x+1)	110 (x^2+x)	101 (x^2+1)	100 (x^2)
100 (x^2)	100 (x^2)	101 (x^2+1)	110 (x^2+x)	111 (x^2+x+1)	000 (0)	001 (1)	010 (x)	011 $(x+1)$
101 (x^2+1)	101 (x^2+1)	100 (x^2)	111 (x^2+x+1)	110 (x^2+x)	001 (1)	000 (0)	011 $(x+1)$	010 (x)
110 (x^2+x)	110 (x^2+x)	111 (x^2+x+1)	100 (x^2)	101 (x^2+1)	010 (x)	011 $(x+1)$	000 (0)	001 (1)
111 (x^2+x+1)	111 (x^2+x+1)	110 (x^2+x)	101 (x^2+1)	100 (x^2)	011 $(x+1)$	010 (x)	001 (1)	000 (0)

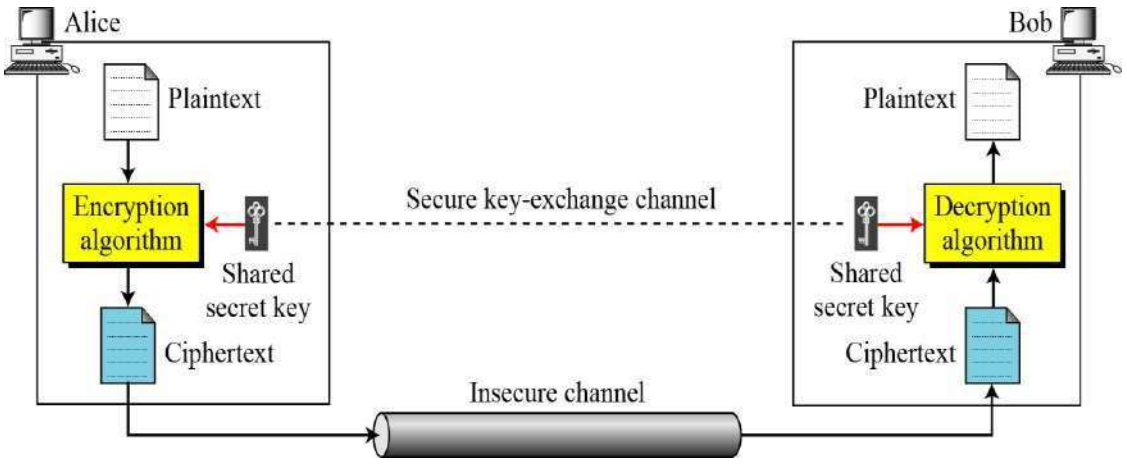
Example :find Multiplication Table for GF(2^3) -with irreducible polynomial is x^3+x^2+1



⊗	000 (0)	001 (1)	010 (x)	011 $(x+1)$	100 (x^2)	101 (x^2+1)	110 (x^2+x)	111 (x^2+x+1)
000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)
001 (1)	000 (0)	001 (1)	010 (x)	011 $(x+1)$	100 (x^2)	101 (x^2+1)	110 (x^2+x)	111 (x^2+x+1)
010 (x)	000 (0)	010 (x)	100 (x)	110 (x^2+x)	101 (x^2+1)	111 (x^2+x+1)	001 (1)	011 $(x+1)$
011 $(x+1)$	000 (0)	011 $(x+1)$	110 (x^2+x)	101 (x^2+1)	001 (1)	010 (x)	111 (x^2+x+1)	100 (x)
100 (x^2)	000 (0)	100 (x^2)	101 (x^2+1)	001 (1)	111 (x^2+x+1)	011 $(x+1)$	010 (x)	110 (x^2+x)
101 (x^2+1)	000 (0)	101 (x^2+1)	111 (x^2+x+1)	010 (x)	011 $(x+1)$	110 (x^2+x)	100 (x^2)	001 (1)
110 (x^2+x)	000 (0)	110 (x^2+x)	001 (1)	111 (x^2+x+1)	010 (x)	100 (x^2)	011 $(x+1)$	101 (x^2+1)
111 (x^2+x+1)	000 (0)	111 (x^2+x+1)	011 $(x+1)$	100 (x^2)	110 (x^2+x)	001 (1)	101 (x^2+1)	010 (x)

Symmetric Key Cipher

The sender and receiver of message use a single common key to encrypt and decrypt messages.



If P is the plaintext, C is the ciphertext, and K is the key,

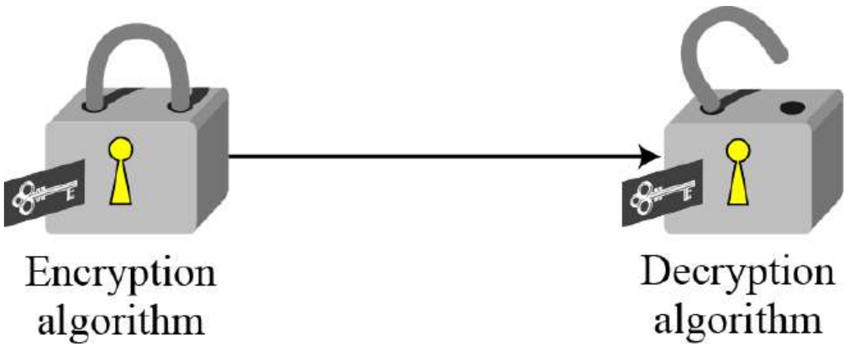
$$\text{Encryption: } C = E_k(P) \qquad \text{Decryption: } P = D_k(C)$$

$$\text{In which, } D_k(E_k(x)) = E_k(D_k(x)) = x$$

We assume that Bob creates P_1; we prove that $P_1 = P$:

$$\textbf{Alice: } C = E_k(P) \qquad \textbf{Bob: } P_1 = D_k(C) = D_k(E_k(P)) = P$$

Figure Locking and unlocking with the same key

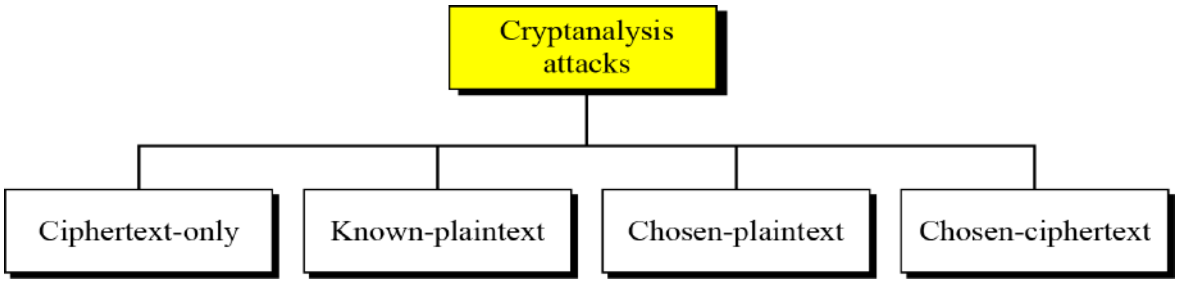


Kerckhoff’s Principle

Based on Kerckhoff’s principle, one should always assume that the adversary, Eve, knows the encryption/decryption algorithm. The resistance of the cipher to attack must be based only on the secrecy of the key.

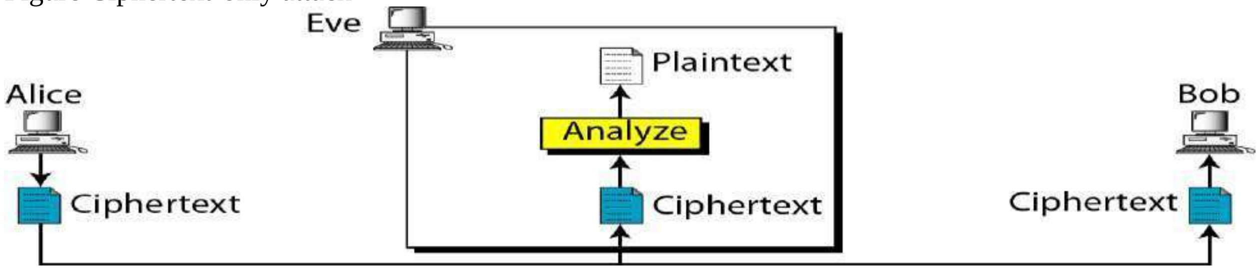
Cryptanalysis

As cryptography is the science and art of creating secret codes, cryptanalysis is the science and art of breaking those codes.



Ciphertext-Only Attack

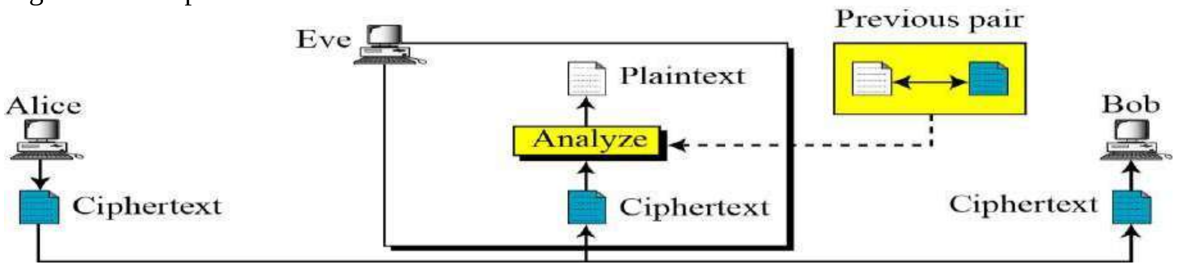
Figure Ciphertext-only attack



In Ciphertext-Only Attack , the attacker knows only some cipher text. He try to find corresponding key and plain text using various methods.
Brute-Force attack: Attacker tries all possible keys. We assume that he knows key domain
Statistical attack: The cryptanalyst can benefit from some inherent charactersistics of the plain text language to perform statistical attack. Example: Letter E is most frequently used character in English.

Known-Plaintext Attack

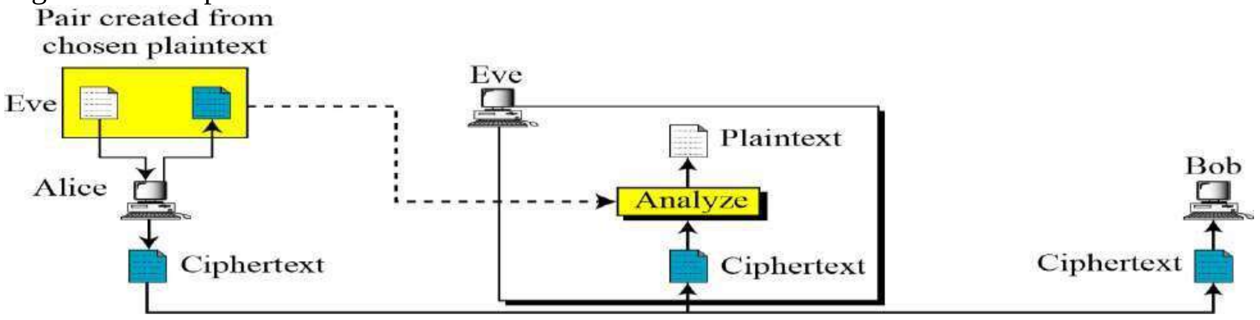
Figure Known-plaintext attack



In this attack, he know some cipher text and plain text pairs that were sent previously by Alice to Bob. Attacker has kept both cipher text and plain text to use them to break the next secrete message.

Chosen-Plaintext Attack

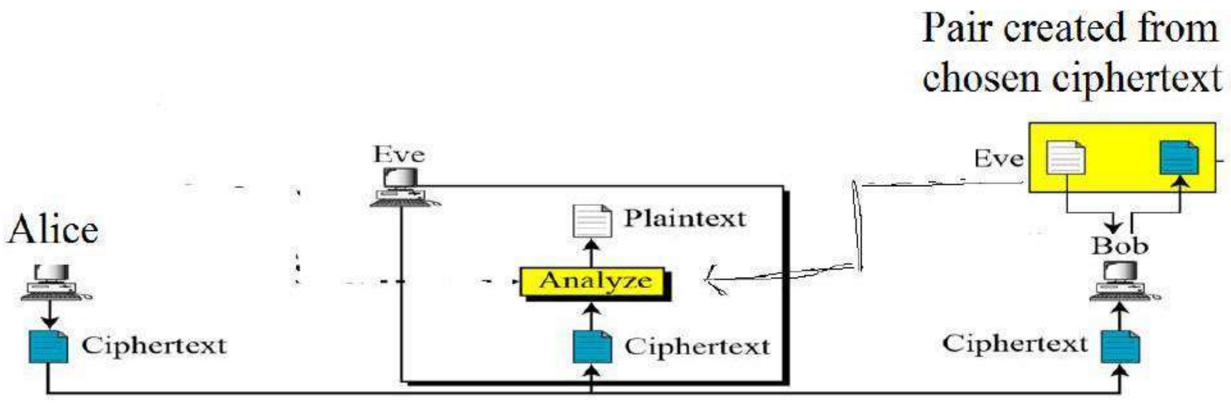
Figure Chosen-plaintext attack



This is similar to known-plaintext attack, but plaintext/cipher text pairs have been choosen by the attacker . This can happen when attacker has access to Alice computer. She can choose some plaintext and interpret ciphertext.

Chosen-Ciphertext Attack

Figure Chosen-Ciphertext attack



This is similar to Chosen Plaintext attack except eve chooses some ciphertext and decrypt it to from a cipher/plain text pairs. This can happen when Eve has access to Bob computer.

Categories of Traditional Ciphers

1. SUBSTITUTION CIPHERS
   A substitution cipher replaces one character with another
2. TRANSPOSITION CIPHERS
   A Transposition cipher reorders symbols

1. SUBSTITUTION CIPHERS

A substitution cipher replaces one symbol with another. Substitution ciphers can be categorized as either monoalphabetic ciphers or polyalphabetic ciphers.

Note:

A substitution cipher replaces one symbol with another.

Monoalphabetic Ciphers:

In monoalphabetic substitution, the relationship between a symbol in the plaintext to a symbol in the ciphertext is always one-to-one.

Example 1

The following shows a plaintext and its corresponding ciphertext. The cipher is probably monoalphabetic because both l's (els) are encrypted as O's.

Plaintext: hello **Ciphertext:** KHOOR

Example 2

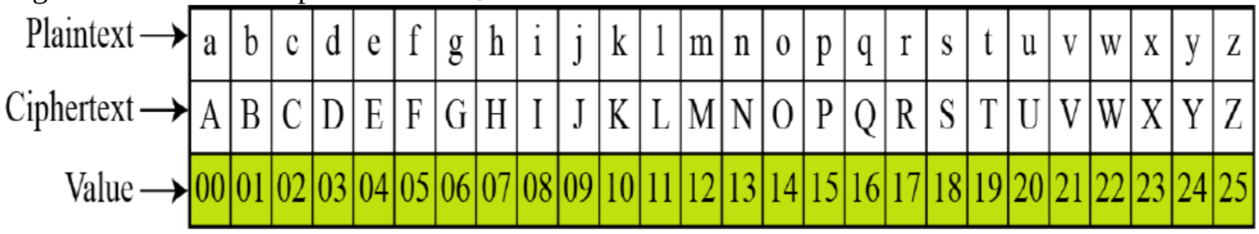
The following shows a plaintext and its corresponding ciphertext. The cipher is not monoalphabetic because each l (el) is encrypted by a different character. The first l (el) is encrypted with N;the second as Z

Plaintext: hello **Ciphertext:** ABNZF

Additive Cipher

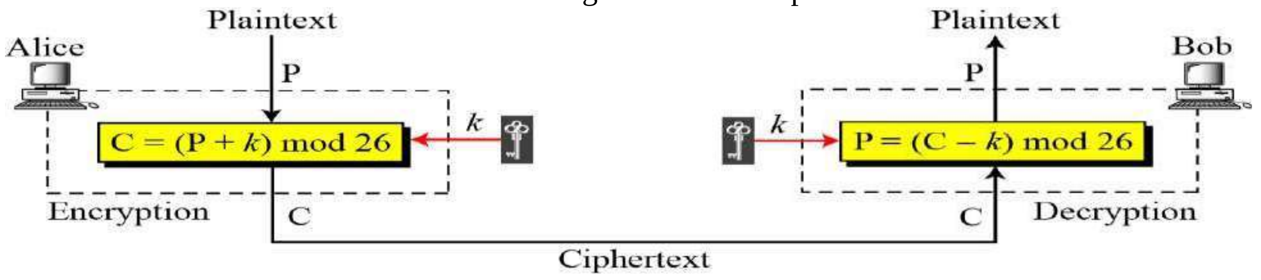
The simplest monoalphabetic cipher is the additive cipher. This cipher is sometimes called a shift cipher and sometimes a Caesar cipher, but the term additive cipher better reveals its mathematical nature.

Figure Plaintext and ciphertext in Z_{26}



Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Figure Additive cipher



Note:

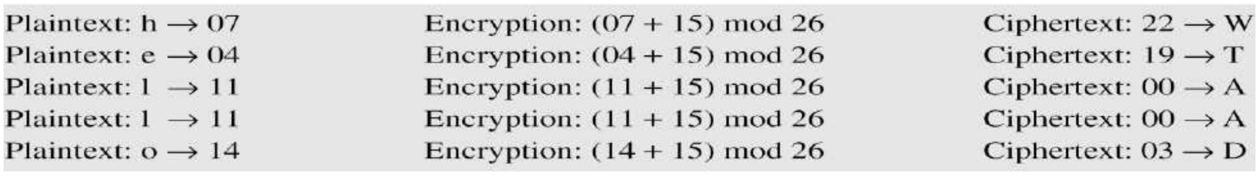
When the cipher is additive, the plaintext, ciphertext, and key are integers in Z_{26}.

Example:

Use the additive cipher with key = 15 to encrypt the message "hello".

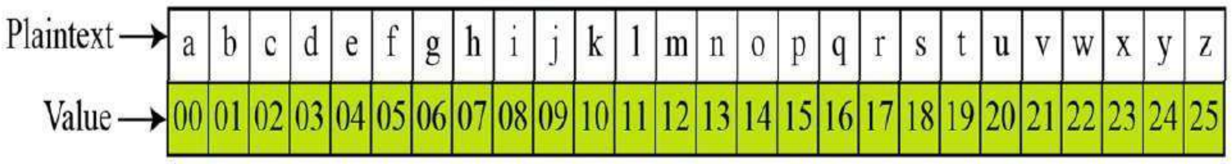
Solution

We apply the encryption algorithm to the plaintext, character by character:



Plaintext: h → 07	Encryption: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: e → 04	Encryption: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: o → 14	Encryption: (14 + 15) mod 26	Ciphertext: 03 → D

Reference



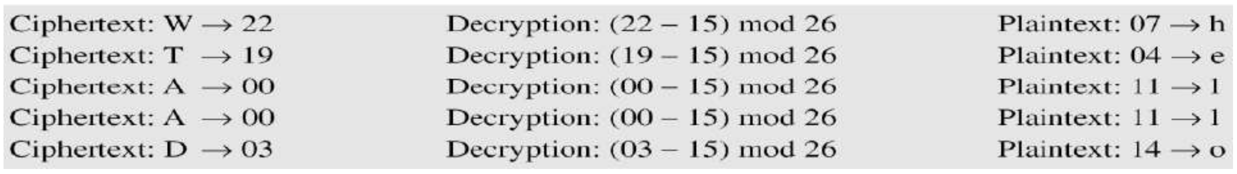
Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Example:

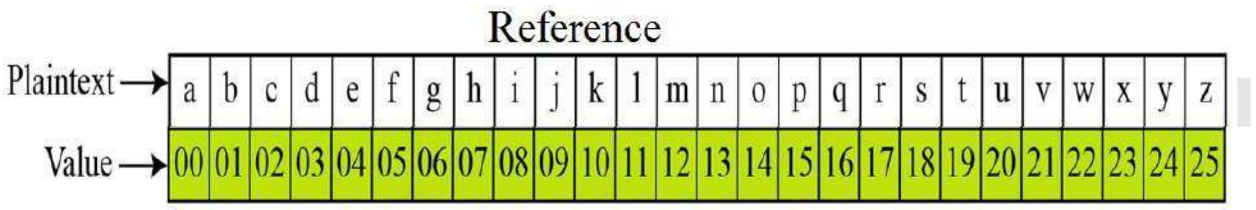
Use the additive cipher with key = 15 to decrypt the message "WTAAD".

Solution

We apply the decryption algorithm to the plaintext character by character:



Ciphertext: W → 22	Decryption: (22 − 15) mod 26	Plaintext: 07 → h
Ciphertext: T → 19	Decryption: (19 − 15) mod 26	Plaintext: 04 → e
Ciphertext: A → 00	Decryption: (00 − 15) mod 26	Plaintext: 11 → l
Ciphertext: A → 00	Decryption: (00 − 15) mod 26	Plaintext: 11 → l
Ciphertext: D → 03	Decryption: (03 − 15) mod 26	Plaintext: 14 → o



Reference

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Shift Cipher and Caesar Cipher

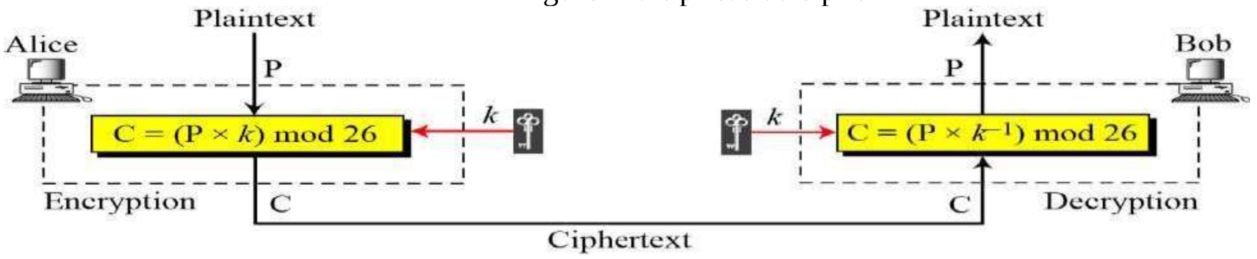
Historically, additive ciphers are called shift ciphers. Julius Caesar used an additive cipher to communicate with his officers. For this reason, additive ciphers are sometimes referred to as the Caesar cipher. Caesar used a key of 3 for his communications.

Note:

Additive ciphers are sometimes referred to as shift ciphers or Caesar cipher

Multiplicative Ciphers

Figure Multiplicative cipher



Note:

In a multiplicative cipher, the plaintext and ciphertext are integers in Z_{26}; the key is an integer in Z_{26}^*.

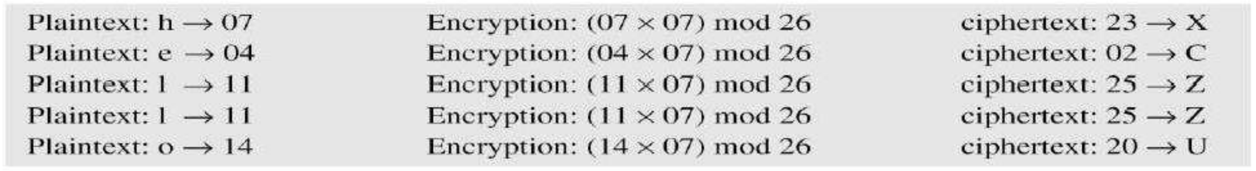
Example1:

What is the key domain for any multiplicative cipher?

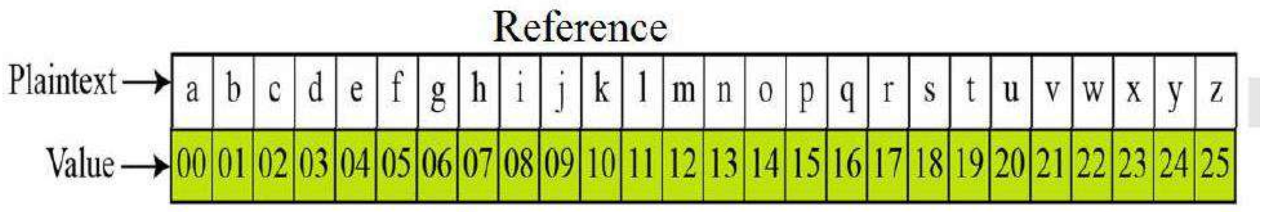
Solution: The key needs to be in Z_{26}^*. This set has only 12 members: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25.

Example2:

We use a multiplicative cipher to encrypt the message “hello” with a key of 7. The ciphertext is “XCZZU”.



Plaintext: h → 07	Encryption: (07 × 07) mod 26	ciphertext: 23 → X
Plaintext: e → 04	Encryption: (04 × 07) mod 26	ciphertext: 02 → C
Plaintext: l → 11	Encryption: (11 × 07) mod 26	ciphertext: 25 → Z
Plaintext: l → 11	Encryption: (11 × 07) mod 26	ciphertext: 25 → Z
Plaintext: o → 14	Encryption: (14 × 07) mod 26	ciphertext: 20 → U

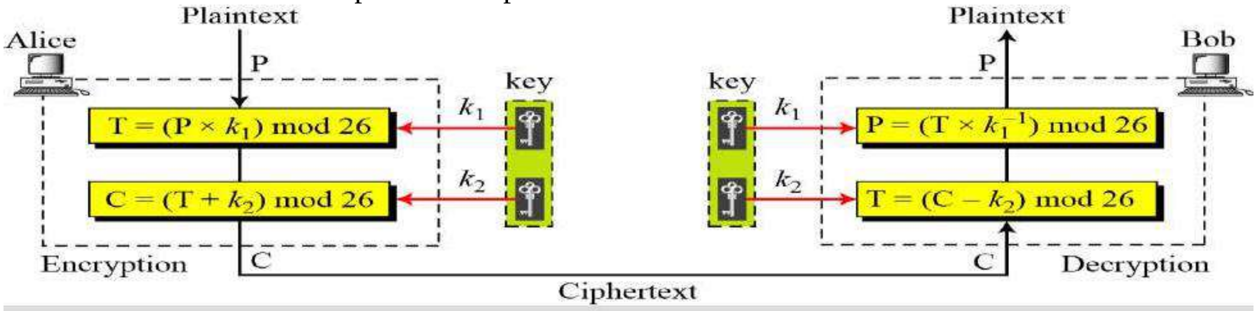


Reference

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Affine Ciphers:

Combine additve and multiplicative Ciphers



$C = (P \times k_1 + k_2) \bmod 26$ $P = ((C - k_2) \times k_1^{-1}) \bmod 26$

where k_1^{-1} is the multiplicative inverse of k_1 and $-k_2$ is the additive inverse of k_2

Example1:

The affine cipher uses a pair of keys in which the first key is from $Z_{26}*$ and the second is from Z_{26}. The size of the key domain is

26 × 12 = 312.

Example2:

Use an affine cipher to encrypt the message "hello" with the key pair (7, 2).

P: h → 07	Encryption: (07 × 7 + 2) mod 26	C: 25 → Z
P: e → 04	Encryption: (04 × 7 + 2) mod 26	C: 04 → E
P: l → 11	Encryption: (11 × 7 + 2) mod 26	C: 01 → B
P: l → 11	Encryption: (11 × 7 + 2) mod 26	C: 01 → B
P: o → 14	Encryption: (14 × 7 + 2) mod 26	C: 22 → W

Monoalphabetic Substitution Cipher

Because additive, multiplicative, and affine ciphers have small key domains, they are very vulnerable to brute-force attack.

A better solution is to create a mapping between each plaintext character and the corresponding ciphertext character. Alice and Bob can agree on a table showing the mapping for each character.

Figure An example key for monoalphabetic substitution cipher

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

Example:

We can use the key in Figure to encrypt the message

this message is easy to encrypt but hard to find the key

The ciphertext is

ICFVQRVVNEFVRNVSIYRGAHSLIOJICNHTIYBFGTICRXRS

Reference

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

Polyalphabetic Ciphers

In polyalphabetic substitution, each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is one-to-many. Example 'a' can be enciphered as 'D' in the beginning of the text, but as 'N' at the middle.

Polyalphabetic has advantage of hiding the letter frequency.
Example: Autokey Cipher

$$\mathrm{P} = \mathrm{P}_1\mathrm{P}_2\mathrm{P}_3 \ldots \qquad \mathrm{C} = \mathrm{C}_1\mathrm{C}_2\mathrm{C}_3\ldots \qquad k = (k_1, \mathrm{P}_1, \mathrm{P}_2, \ldots)$$

$$\text{Encryption: } \mathrm{C}_i = (\mathrm{P}_i + k_i) \bmod 26 \qquad \text{Decryption: } \mathrm{P}_i = (\mathrm{C}_i - k_i) \bmod 26$$

Example:

Assume that Alice and Bob agreed to use an autokey cipher with initial key value k1 = 12. Now Alice wants to send Bob the message "Attack is today". Enciphering is done character by character.

Plaintext:	a	t	t	a	c	k	i	s	t	o	d	a	y
P's Values:	00	19	19	00	02	10	08	18	19	14	03	00	24
Key stream:	*12*	*00*	*19*	*19*	*00*	*02*	*10*	*08*	*18*	*19*	*14*	*03*	*00*
C's Values:	12	19	12	19	02	12	18	00	11	7	17	03	24
Ciphertext:	M	T	M	T	C	M	S	A	L	H	R	D	Y

TRANSPOSITION CIPHERS

A transposition cipher does not substitute one symbol for another, instead it changes the location of the symbols. A symbol in the first position may appaer in the tenth position of the cipher. A symbol in the eighth position may appear in the first osition of the cipher.

Note: A transposition cipher reorders symbols

Keyless Transposition Ciphers

Simple transposition ciphers, which were used in the past, are keyless.

Example 1:

A good example of a keyless cipher using the first method is the rail fence cipher. The ciphertext is created reading the pattern row by row. For example, to send the message "***Meet me at the park***" to Bob, Alice writes

```
m     e     m     a     t     e     a     k
 ↘   ↗ ↘   ↗ ↘   ↗ ↘   ↗ ↘   ↗ ↘   ↗ ↘   ↗
   e     t     e     t     h     p     r
```

She then creates the ciphertext "MEMATEAKETETHPR".

Example 2:

Alice and Bob can agree on the number of columns and use the second method. Alice writes the same plaintext, row by row, in a table of four columns.

m	e	e	t
m	e	a	t
t	h	e	p
a	r	k	

She then creates the ciphertext "MMTAEEHREAEKTTP" by transmitting the characters column by column. Bob receives the cipher text and follows the reverse process to get plain text .

Example:

m	e	e	t
m	e	a	t
t	h	e	p
a	r	k	

The cipher in previous example is actually a transposition cipher. **The following shows the permutation of each character in the plaintext into the ciphertext based on the positions.**

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
01	05	09	13	02	06	10	13	03	07	11	15	04	08	12

The second character in the plaintext has moved to the fifth position in the ciphertext; the third character has moved to the ninth position; and so on.Although the characters are permuted, there is a **pattern** in the permutation: (01, 05, 09, 13), (02, 06, 10, 13), (03, 07, 11, 15), and (4, 8, 12). In each section, the difference between the two adjacent numbers is 4.

Keyed Transposition Ciphers

The keyless ciphers permute the characters by using writing plaintext in one way and reading it in another way The permutation is done on the whole plaintext to create the whole ciphertext.

Another method is to divide the plaintext into groups of predetermined size, called blocks, and then **use a key** to permute the characters in each block separately.

Example

Alice needs to send the message "**Enemy attacks tonight**" to Bob..

e n e m y a t t a c k s t o n i g h t z

The key used for encryption and decryption is a permutation key, which shows how the character are permuted.

Encryption ↓

3	1	4	5	2
1	2	3	4	5

↑ Decryption

The permutation yields

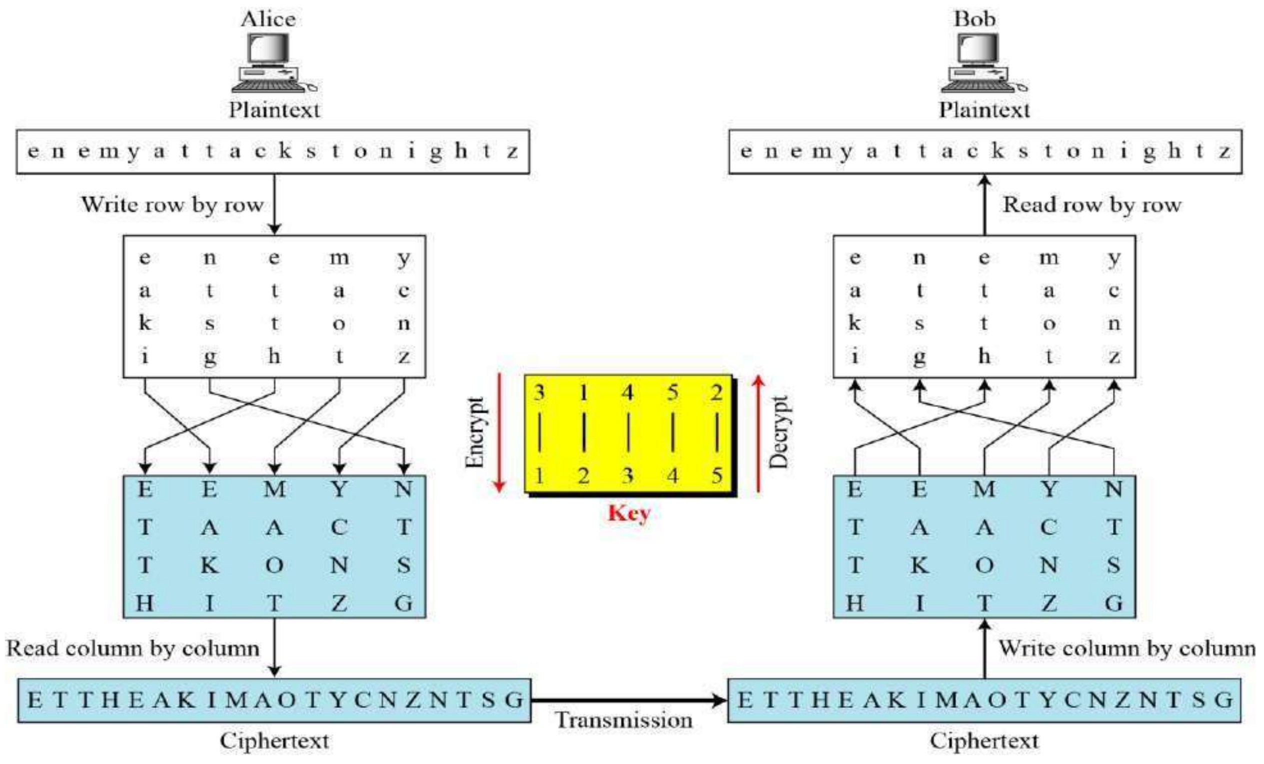
E E M Y N T A A C T T K O N S H I T Z G

Combining Two Approaches for better result

Encryption or decryption is done in 3 steps:

1) Text is written into row by row
2) Permutation is done by reordering columns
3) New table is read column by column

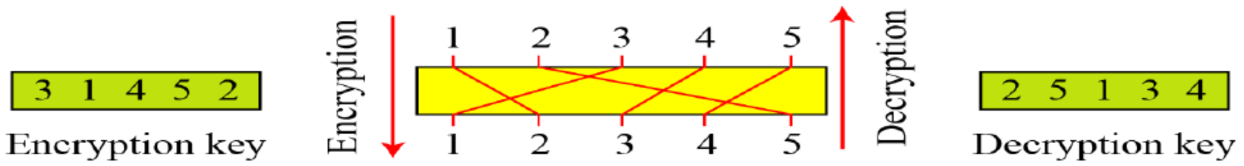
Example



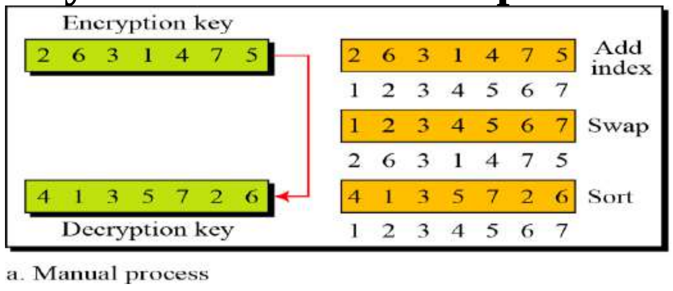
Keys

In the previous Example, a single key was used in two directions for the column exchange: downward for encryption, upward for decryption. It is customary to create two keys.

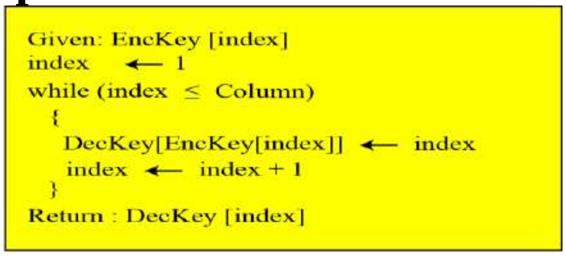
Figure Encryption/decryption keys in transpositional ciphers



Key inversion in a transposition cipher



a. Manual process



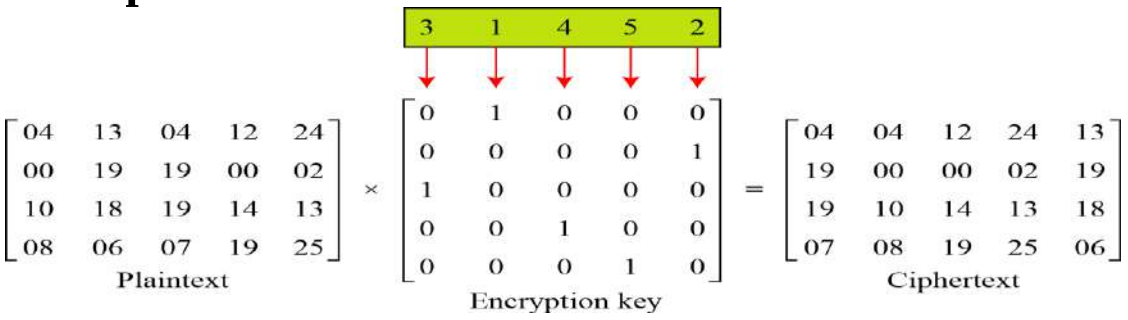
```
Given: EncKey [index]
index ← 1
while (index ≤ Column)
  {
    DecKey[EncKey[index]] ← index
    index ← index + 1
  }
Return : DecKey [index]
```

b. Algorithm

Using Matrices

We can use matrices to show the encryption/decryption process for a transposition cipher. The plain text and cipher text are lxm matrices with numberical values of characters and keys are mxm matri x.
In a permutation matrix, every row or column has exactly one 1 and others are 0's. Encryption multiplies plaintext matrix with key matrix and decryption multiplies ciphertext matrix with inverse of key matrix(This simply the transpostion of key matrix)

Example



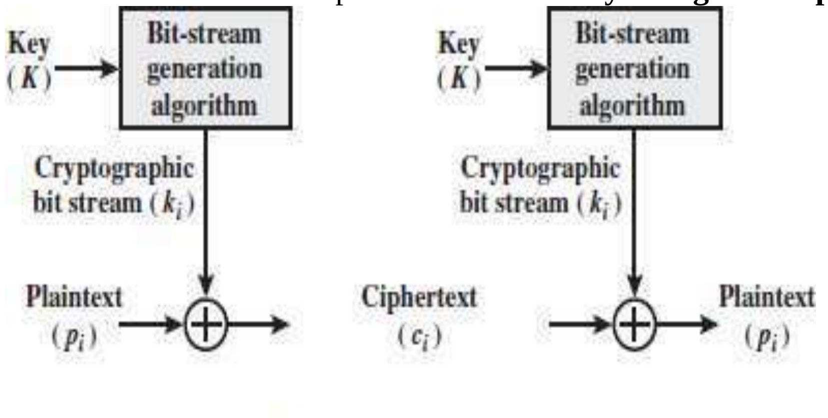
$$\begin{bmatrix} 04 & 13 & 04 & 12 & 24 \\ 00 & 19 & 19 & 00 & 02 \\ 10 & 18 & 19 & 14 & 13 \\ 08 & 06 & 07 & 19 & 25 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 04 & 04 & 12 & 24 & 13 \\ 19 & 00 & 00 & 02 & 19 \\ 19 & 10 & 14 & 13 & 18 \\ 07 & 08 & 19 & 25 & 06 \end{bmatrix}$$

Figure Representation of the key as a matrix in the transposition cipher

<u>Stream Ciphers and Block Ciphers</u>

Stream Ciphers

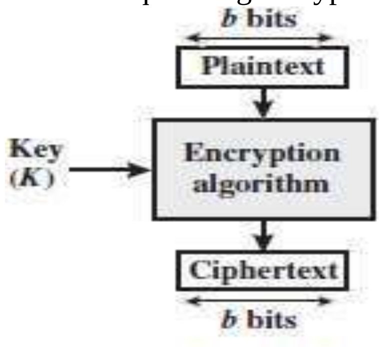
- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the **autokeyed Vigenère cipher and the Vernam cipher.**


Key
(K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Plaintext (p_i)
Key
(K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Ciphertext (c_i)
Plaintext (p_i)

(a) Stream cipher using algorithmic bit-stream generator

Block Ciphers

A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically, a block size of 64 or 128 bits is used.

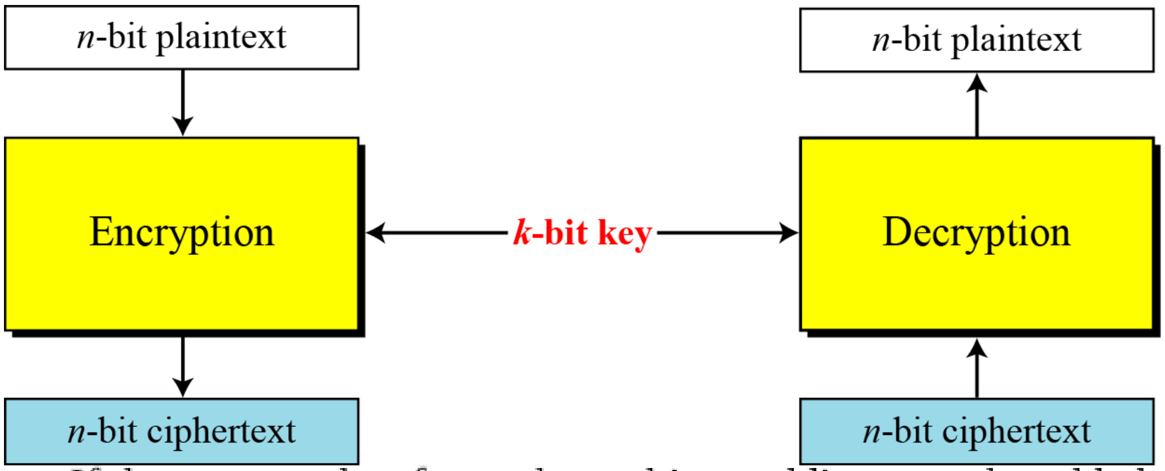


(b) Block cipher

Modern Block Ciphers

A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of cipher text. The encryption or decryption algorithm uses a k-bit key. The Decryption algorithm must be the inverse of the encryption algorithm and must use the same secrete key.

Figure A modern block cipher



If the message has fewer than n bits, padding must be added to make it an n-bit block; if the message has more than n bits, it should be divided into n-bit blocks and the appropriate padding must be added to the last block if necessary. The common values for n are 64, 128, 256, or 512 bits.

Example: How many padding bits must be added to a message of 100 characters if 8-bit ASCII is used for encoding and the block cipher accepts blocks of 64 bits?

Solution

Encoding 100 characters using 8-bit ASCII results in an 800-bit (100x8) message. The plaintext must be divisible by 64. If | M | and |Pad| are the length of the message and the length of the padding,

$$|M| + |Pad| = 0 \bmod 64 \quad \rightarrow \quad |Pad| = -800 \bmod 64 \quad \rightarrow \quad 32 \bmod 64$$

A modern block cipher can be designed to act as a substitution cipher or a transposition cipher.

To be resistant to exhaustive-search attack, a modern block cipher needs to be designed as a substitution cipher.

Example

Suppose that we have a block cipher where $n = 64$. If there are 10 1's in the ciphertext, how many trial-and-error tests does Eve need to do to recover the plaintext from the intercepted ciphertext in each of the following cases?

a. The cipher is designed as a substitution cipher.
b. The cipher is designed as a transposition cipher.

Solution

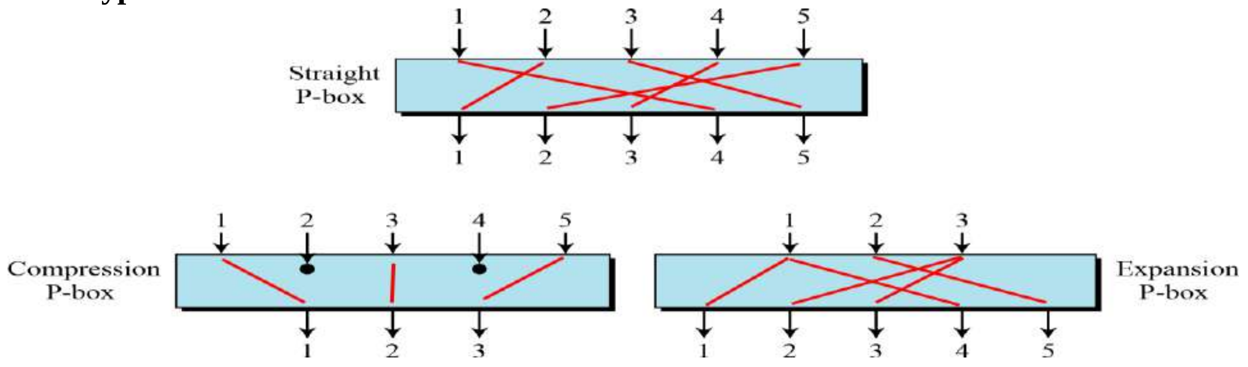
a) In the first case, Eve has no idea how many 1's are in the plaintext. Eve needs to try all possible 2^{64} 64-bit blocks to find one that makes sense.
b) In the second case, Eve knows that there are exactly 10 1's in the plaintext. Eve can launch an exhaustive-search attack using only those 64-bit blocks that have exactly 10 1's.

Components of a Modern Block Cipher

Modern block ciphers normally are keyed substitution ciphers in which the key allows only partial mappings from the possible inputs to the possible outputs. It usses P-Boxes,S-Boxes.

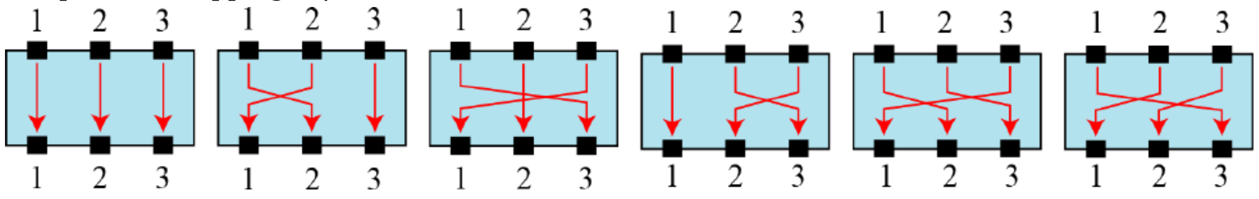
P-Boxes

P-Boxes(also called ad D-Box means Diffusion box)
A P-box (permutation box) parallels the traditional transposition cipher for characters. It transposes bits.
Three types of P-boxes



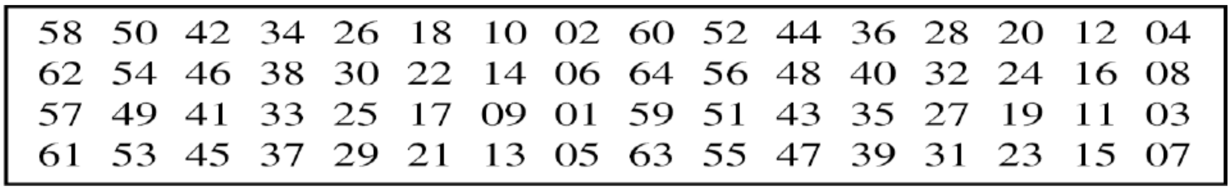
Example
Figure shows all 6 possible mappings of a 3 × 3 P-box.

The possible mappings of a 3 × 3 P-box



Straight P-Boxes

Table Example of a permutation table for a straight P-box(64x64)
At output of P-Box:
Input 58 goes to 1st position, input 50 goes to 2nd position, input 42 to 3rd position,….



58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

Example

Design an 8 × 8 permutation table for a straight P-box that moves the two middle bits (bits 4 and 5) in the input word to the two ends (bits 1 and 8) in the output words. Relative positions of other bits should not be changed.
Solution

We need a straight P-box with the table [4 1 2 3 6 7 8 5]. The relative positions of input bits 1, 2, 3, 6, 7, and 8 have not been changed, but the first output takes the fourth input and the eighth output takes the fifth input.

Compression P-Boxes

Example of a 32 × 24 permutation table

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

Some of the input bits are blocked at output: example: 7,8,9,15,16,23,24,25

Expansion P-Boxes

Example of a 12 × 16 permutation table

01	09	10	11	12	01	02	03	03	04	05	06	07	08	09	12

1,3,9,12 are mapped to two outputs

P-Boxes: Invertibility

A straight P-Box is invertible, that means we use straight P-Box in encryption cipher and its inverse in decryption cipher.

Note

A straight P-box is invertible, but compression and expansion P-boxes are not.

Example

Figure shows how to invert a permutation table represented as a one-dimensional table.

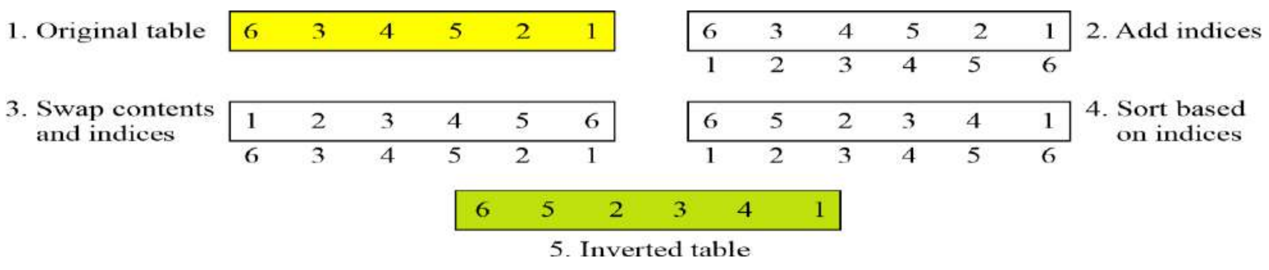
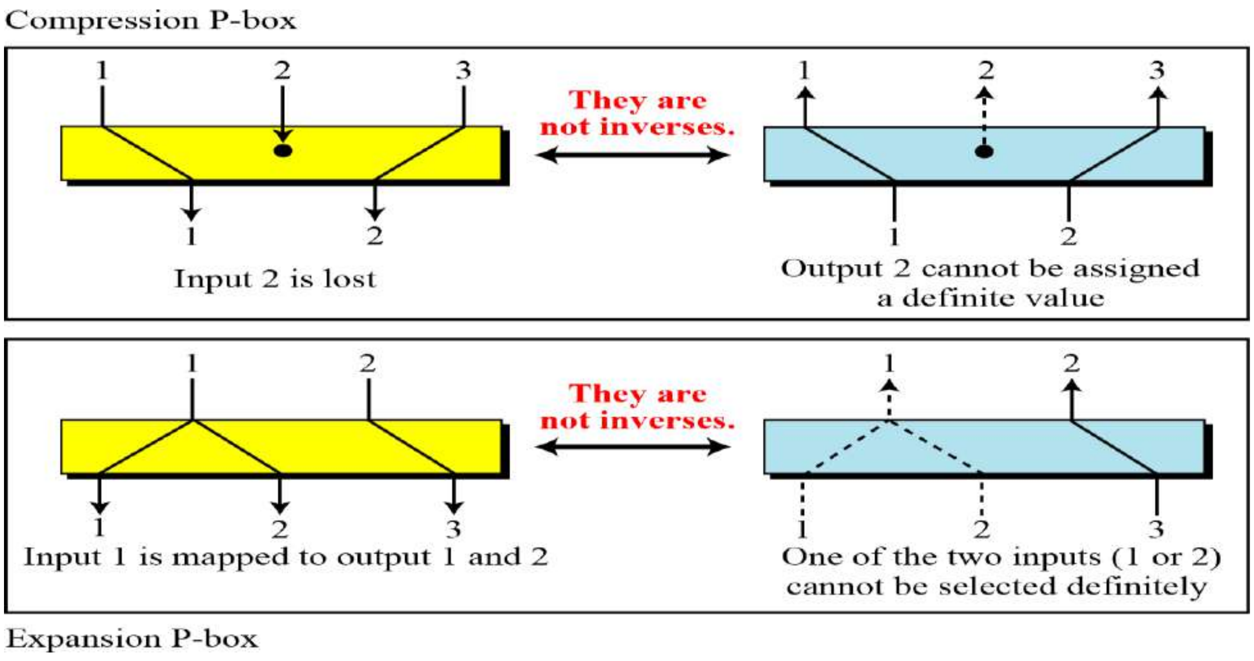


Figure ***Compression and expansion P-boxes are non-invertible***


Compression P-box
1
2
3
1
2
Input 2 is lost
They are not inverses.
1
2
3
1
2
Output 2 cannot be assigned a definite value
1
2
1
2
3
Input 1 is mapped to output 1 and 2
They are not inverses.
1
2
1
2
3
One of the two inputs (1 or 2) cannot be selected definitely
Expansion P-box

S-Box

An S-box (substitution box) can be thought of as a small substitution cipher

Note

An S-box is an $m \times n$ substitution unit, where m and n are not necessarily the same.

Linear S-Box: if the inputs are x1,x2,x3… and outputs are y1,y2,y3… and relationship between them is

Y1=f1(x_1,x_2,x_3..) ,

Y2=f2 (x_1,x_2,x_3..)

…..

Then above relation can be expressed as

Y1=$a_{11}x_1$+$a_{12}x_2$+…

Y2=$a_{21}x_1$+$a_{22}x_2$+…

Example: In a nonlinear s-box, such boxes can have 'and' terms like x_1x_2, x_3x_5…

In an S-box with three inputs and two outputs, we have

$$y_1 = x_1 \oplus x_2 \oplus x_3 \qquad y_2 = x_1$$

The S-box is linear because $a_{1,1} = a_{1,2} = a_{1,3} = a_{2,1} = 1$ and $a_{2,2} = a_{2,3} = 0$. The relationship can be represented by matrices, as shown below:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Example

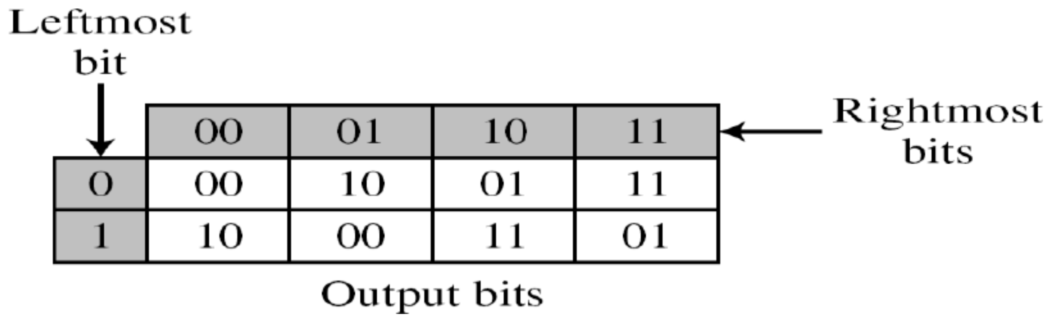
In an S-box with three inputs and two outputs, we have

$$y_1 = (x_1)^3 + x_2 \qquad y_2 = (x_1)^2 + x_1x_2 + x_3$$

where multiplication and addition is in GF(2). The S-box is nonlinear because there is no linear relationship between the inputs and the outputs.

Example

The following table defines the input/output relationship for an S-box of size 3 × 2. The leftmost bit of the input defines the row; the two rightmost bits of the input define the column. The two output bits are values on the cross section of the selected row and column.


Leftmost bit
Rightmost bits
Output bits

	00	01	10	11
0	00	10	01	11
1	10	00	11	01

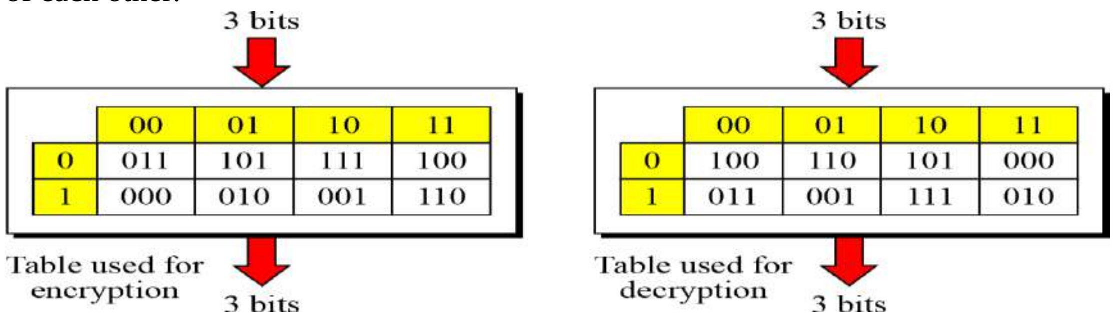
Based on the table, an input of 010 yields the output 01. An input of 101 yields the output of 00.

S-Boxes: Invertibility

An S-box may or may not be invertible. In an invertible
S-box, the number of input bits should be the same as the number of output bits.

Example

Figure shows an example of an invertible S-box. For example, if the input to the left box is 001, the output is 101. The input 101 in the right table creates the output 001, which shows that the two tables are inverses of each other.

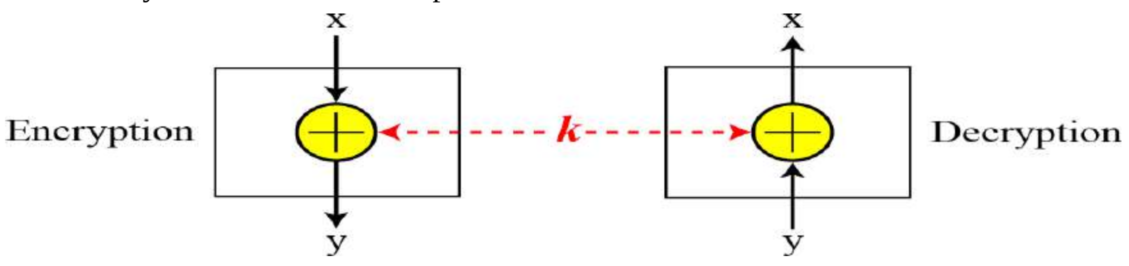

3 bits
3 bits
Table used for encryption
3 bits
3 bits
Table used for decryption

	00	01	10	11
0	011	101	111	100
1	000	010	001	110

	00	01	10	11
0	100	110	101	000
1	011	001	111	010

Exclusive-OR

An important component in most block ciphers is the exclusive-or operation.
Invertibility of the exclusive-or operation



Product Ciphers

Shannon introduced the concept of a product cipher. A product cipher is a complex cipher combining substitution, permutation, and other components .

Combination of S-box and P-box transformation—a product cipher.

Two classes of product ciphers:

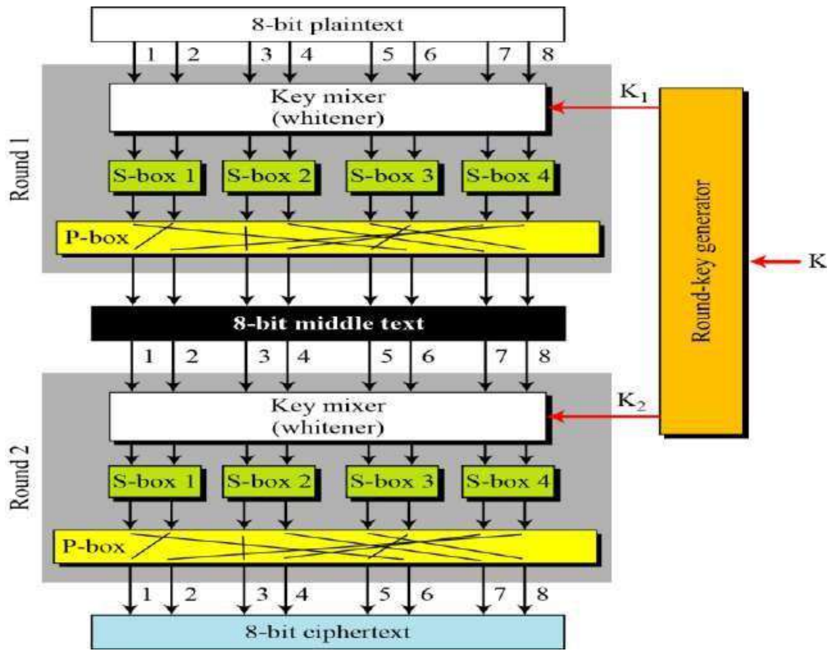
a) Feistel ciphers, Example DES(data encryption standard)
b) Non-feistel Ciphers, Example AES(Advanced Encryptin system)

Diffusion
The idea of diffusion is to hide the relationship between the ciphertext and the plaintext.

Confusion
The idea of confusion is to hide the relationship between the ciphertext and the key.

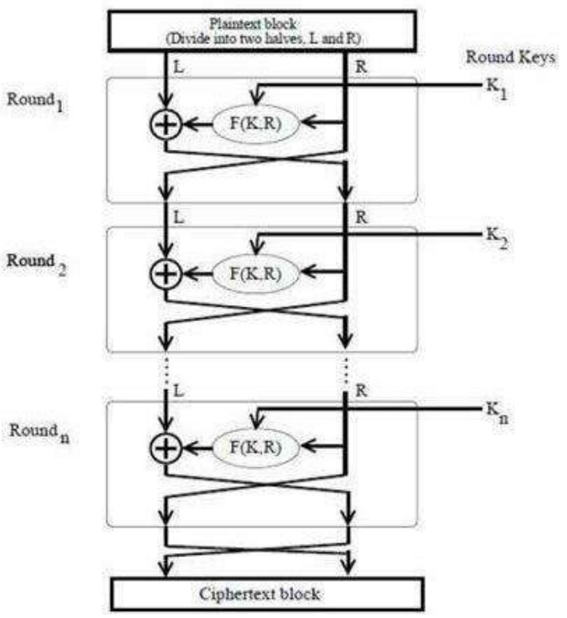
Rounds
Diffusion and confusion can be achieved using iterated product ciphers where each iteration is a combination of S-boxes, P-boxes, and other components.

Figure *A product cipher made of two rounds*



Feistel Cipher Structure:

- Feistel Cipher is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived.
- DES is just one example of a Feistel Cipher.
- A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.
- The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.
- In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function 'f' that takes two input – the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.



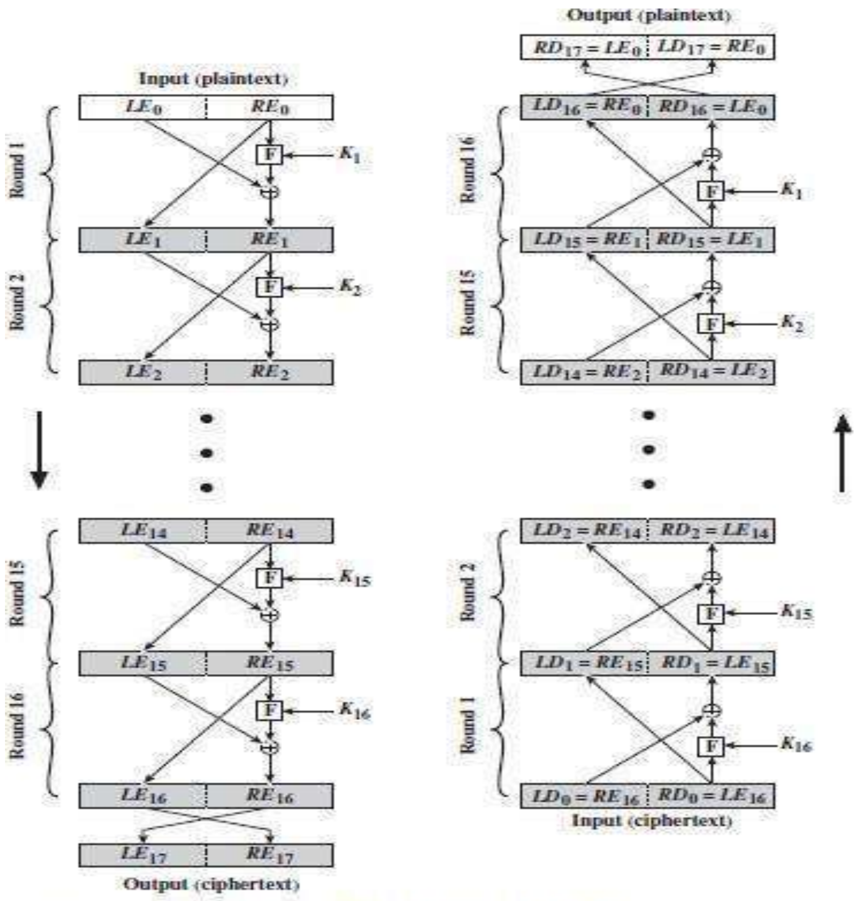


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

$$RE_{i-1} = LE_i$$

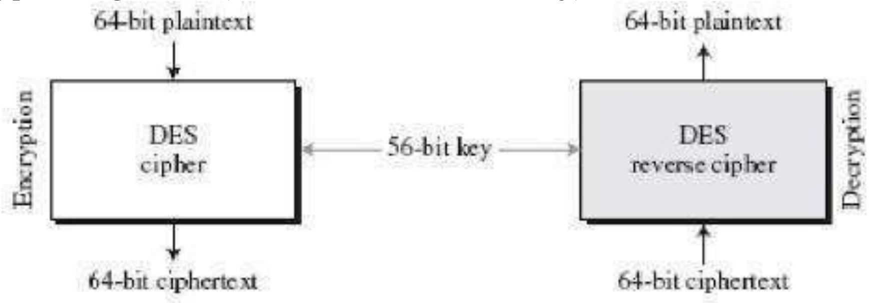
$$LE_{i-1} = RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i)$$

Block Cipher Design Principles

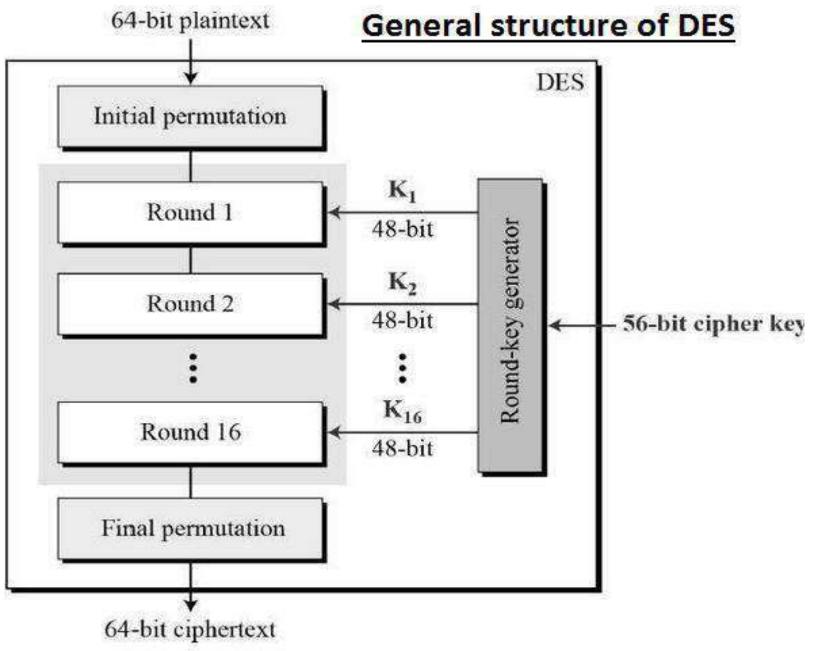
Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion.
Key size: Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion
Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
Round function F: Again, greater complexity generally means greater resistance to cryptanalysis.
Diffusion And Confusion:- The terms *diffusion* and *confusion* were introduced by Claude Shannon to capture the two basic building blocks (Plain Text & Cipher Text) for any cryptographic system.

Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).
DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).



General structure of DES

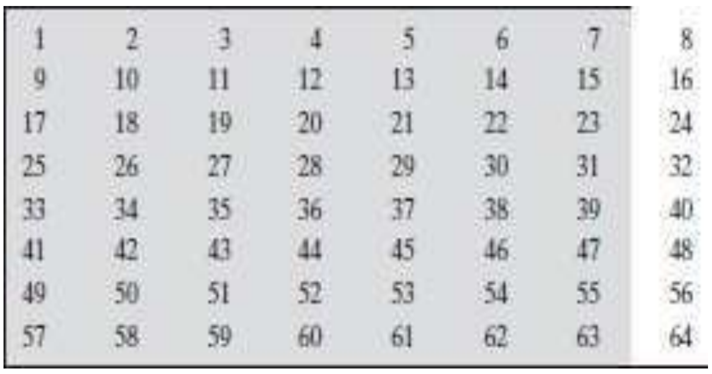


DES Symmetric key Block Cipher algorithm. DES follows Feistel cipher structure.
Plain Text Block Size :
64 Bits Cipher Text Size :
64 Bits
Master Key Size : 64 / 56
Bits No. Of Rounds 16
Round Key / Subkey Size: 48 Bits.

Initial Permutation & Inverse Initial Permutation

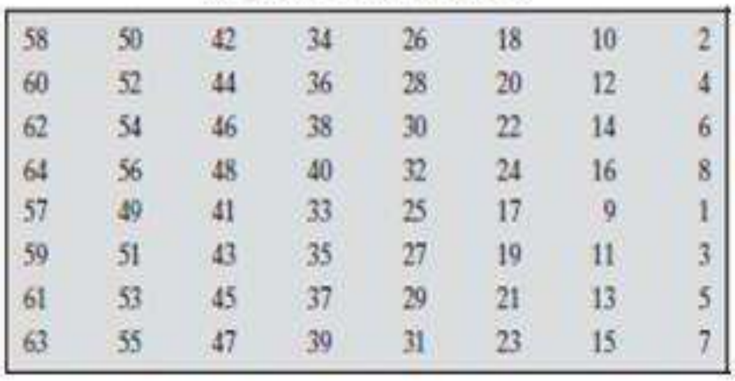
The initial permutation and its inverse are defined by tables, as shown in Tables.
The tables are to be interpreted as follows.
The input to a table consists of 64 bits numbered from 1 to 64.
The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64.
Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.
The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other.
Note:
Initial Permutation & Inverse Initial Permutations have no cryptography significance in DES.

Input Table



1	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

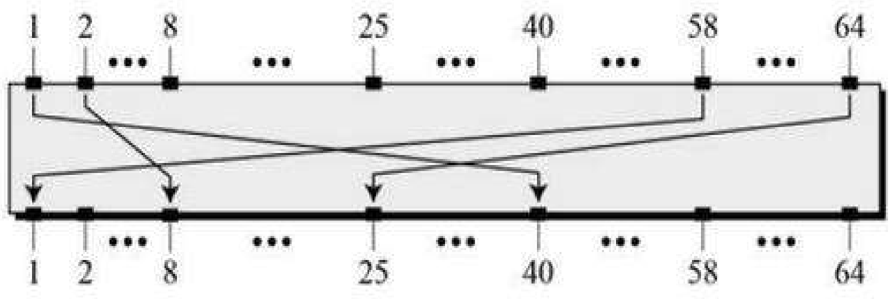
(a) Initial Permutation (IP)



58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

In output
At 1st place 58
At 2nd place 50

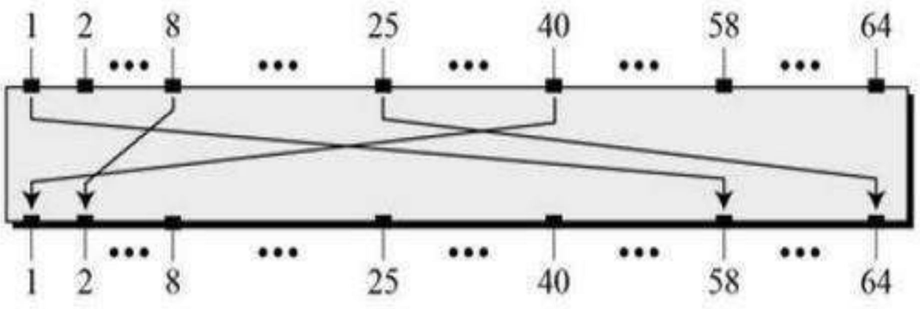
At 3rd place 42 ..



(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

In output
At 1st place 40
At 2nd place 8
At 3rd place 48 ..



Rounds

The left and right halves of each 64- bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right).
As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

The round key **K_i** is 48 bits. The R input is 32 bits. This **R** input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the **R** bits.

The resulting 48 bits are XORed with **Ki**. This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table.
The role of the **S-boxes** in the function F is illustrated in Figure 3.7.The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table 3.3, which is interpreted as follows: The first and last bits of the input to box **Si** form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for **Si**. The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output. For example, in S1, for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

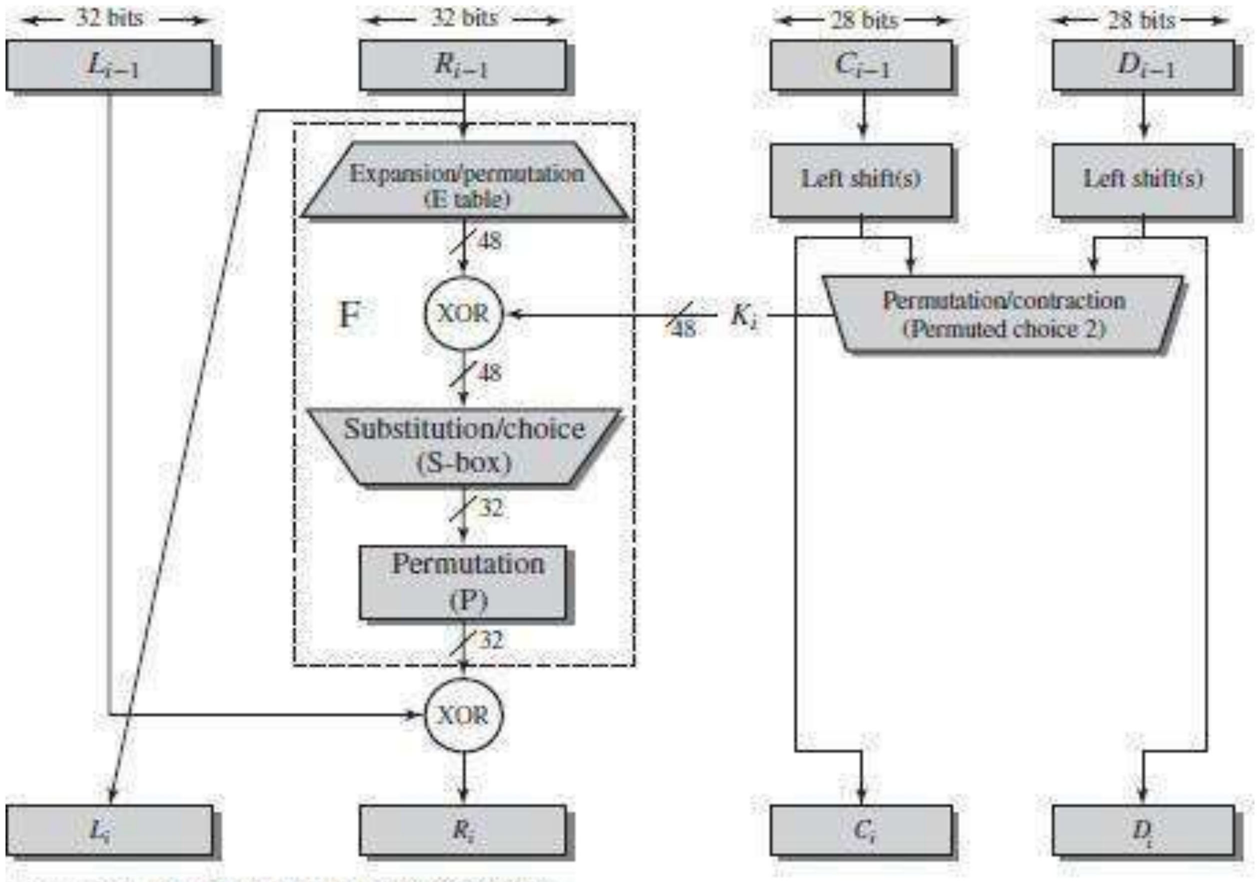
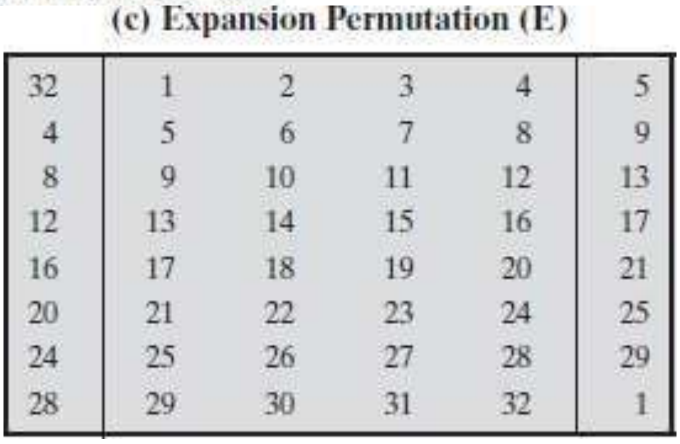


Figure 3.6 Single Round of DES Algorithm



(c) Expansion Permutation (E)

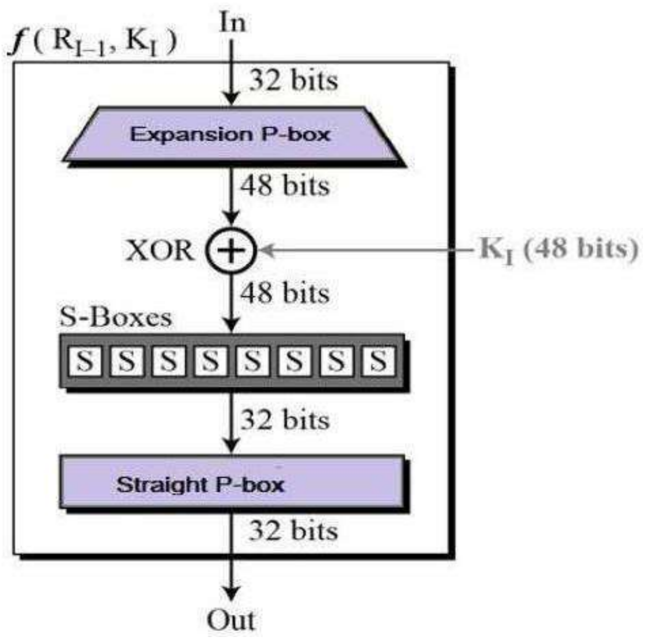
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Permutation Function (P)

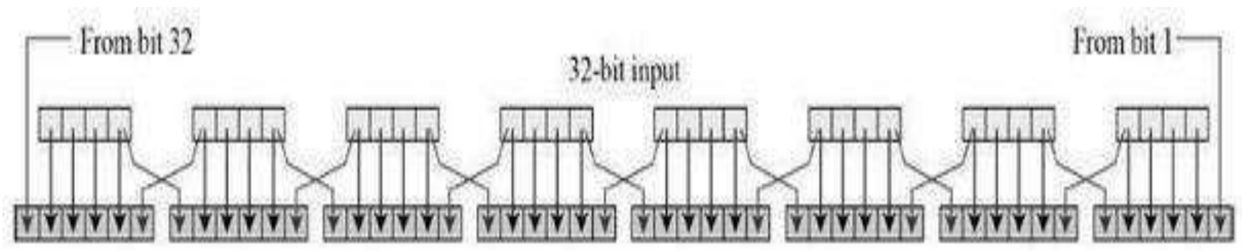
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Round Function

The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



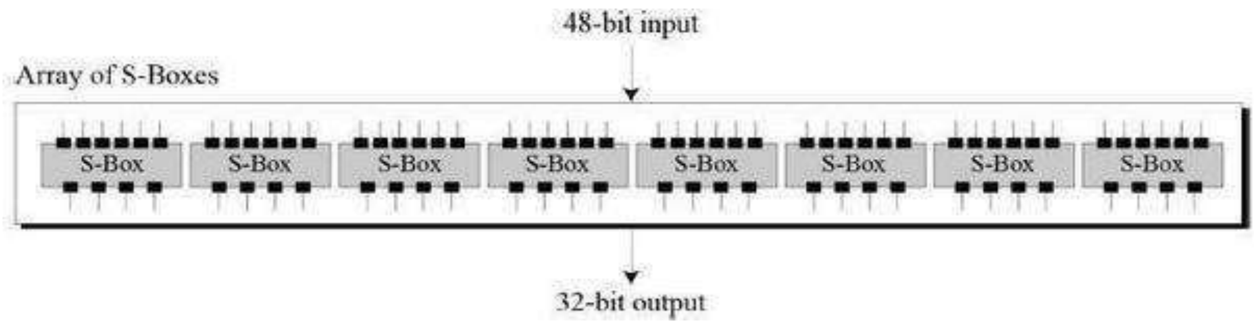
Expansion Permutation Box − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration −



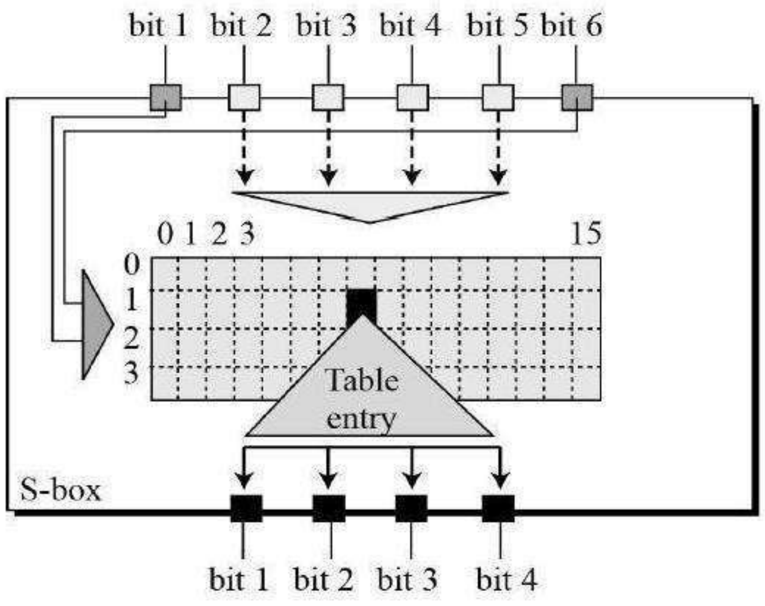
The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown −

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

XOR (Whitener). − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
Substitution Boxes. − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −



The S-box rule is illustrated below −



There are a total of eight S-box tables.
The output of all eight s-boxes is then combined in to 32 bit section.

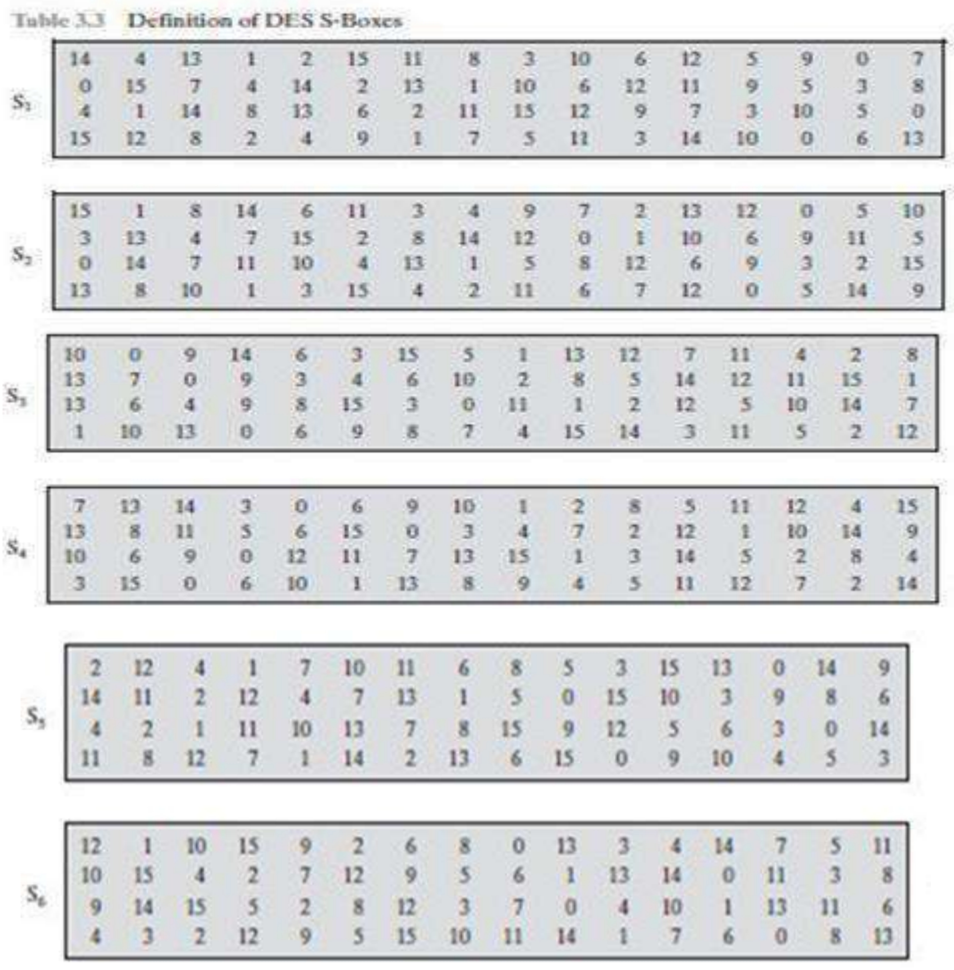


Table 3.3 Definition of DES S-Boxes

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
S_7	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
S_8	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The 32-bit output from the eight S-boxes is then permuted, so that on the next round, the output from each S-box immediately affects as many others as possible.

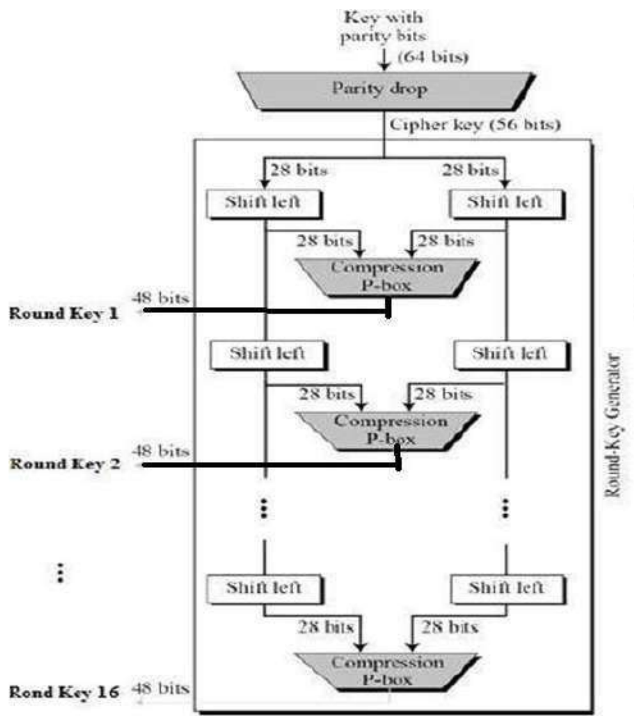
Straight Permutation

− The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

DES Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration −



Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

Table 3.4 DES Key Schedule Calculation

(a) Input Key

1	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

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Decryption

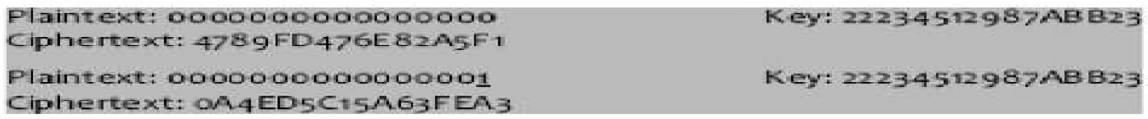
As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

DES Analysis

Two desired properties of a block cipher are the Avalanche effect and the completeness.
Avalanche effect :

A small change in plaintext results in the very great change in the ciphertext.

EXAMPLE 6.7 To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.



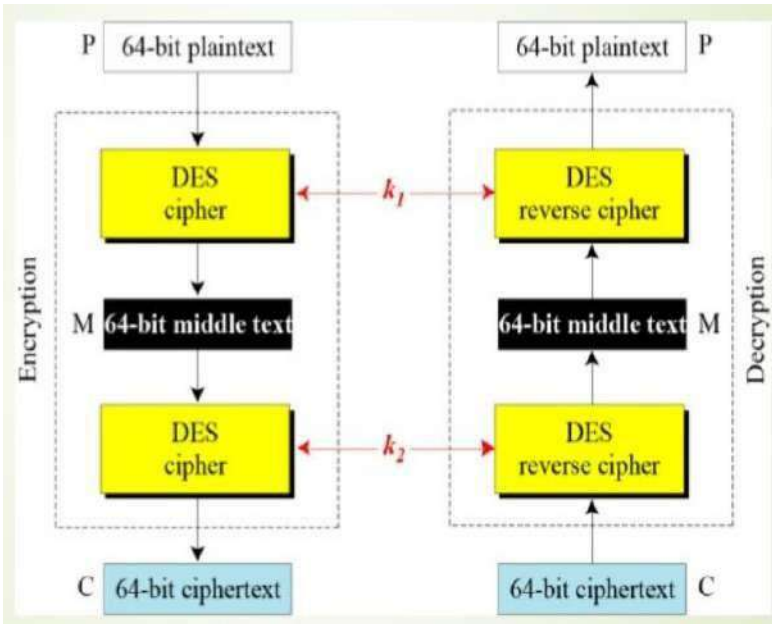
```
Plaintext: 0000000000000000          Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1
Plaintext: 0000000000000001          Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3
```

Completeness effect:
Completeness effect means that each bit of ciphertext needs to depends on many bits on the plaintext. The diffusion and confusion produced by P-Boxes and S-Boxes in DES, show a very strong completeness effect.

DES Weaknesses Analysis

Weakness in Cipher Design:
It is not clear why the designers of DES used the initial and final permutations; these have no security benefits.
In the expansion permutation, the first and fourth bits of every 4-bit series are repeated.
Weakness in Cipher Key:
o DES Key size is 56 bits. To do Brute force attack on a given ciphertext block, the adversary needs to check 2^{56} keys.
With available technology it is possible to check 1 million keys per second

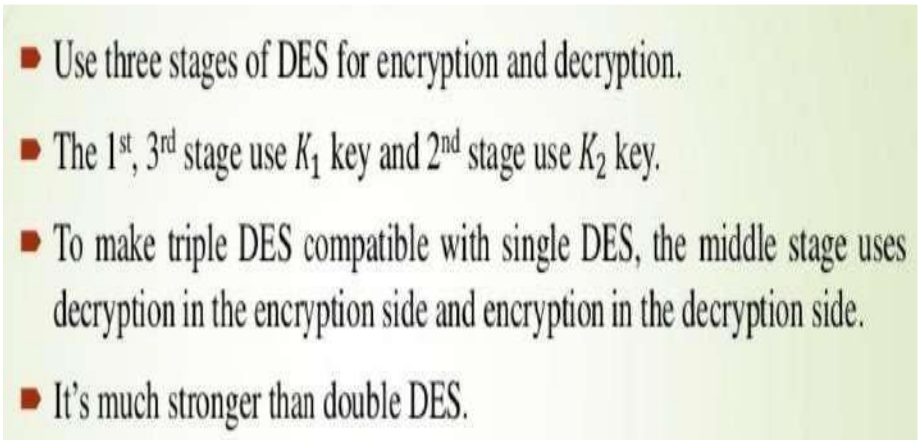
<u>Double – DES</u>



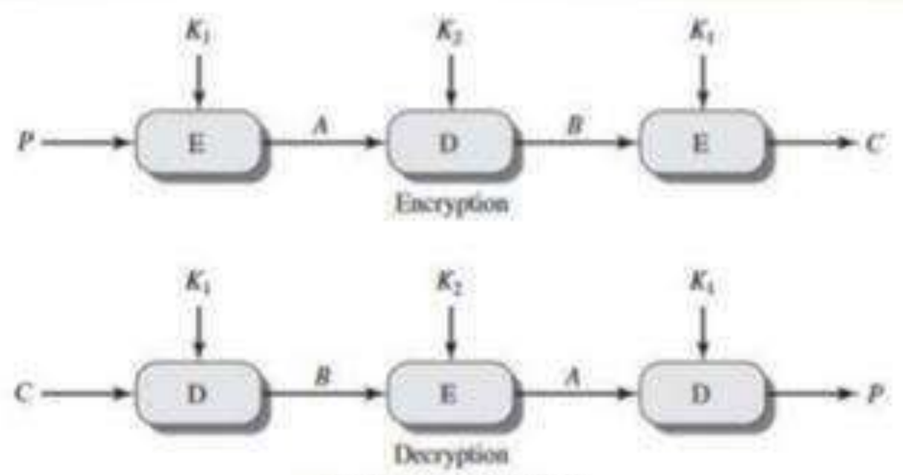
Triple – DES

Triple DES was developed in 1999 by IBM – by a team led by Walter Tuchman. DES prevents a meet-in-the-middle attack. 3-DES has a 168-bit key and enciphers blocks of 64 bits.

3-DES with 2 Keys:



- Use three stages of DES for encryption and decryption.
- The 1st, 3rd stage use K_1 key and 2nd stage use K_2 key.
- To make triple DES compatible with single DES, the middle stage uses decryption in the encryption side and encryption in the decryption side.
- It's much stronger than double DES.



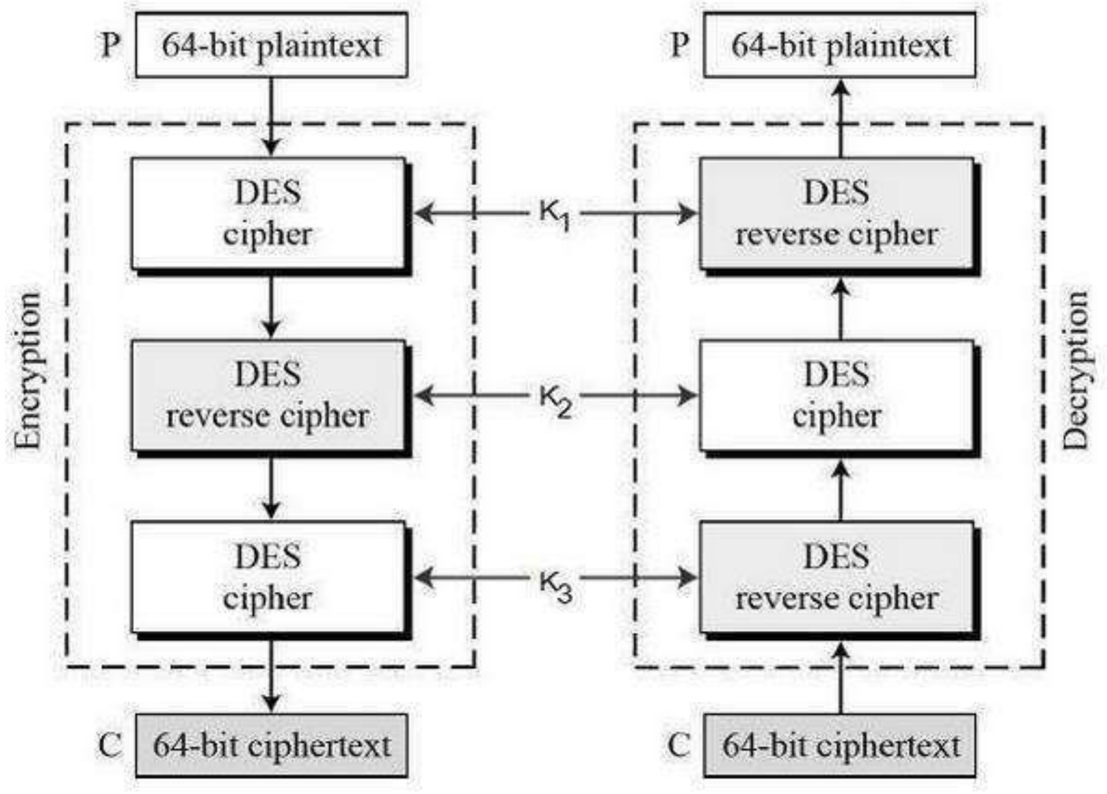
(b) Triple encryption

- The function follows an encrypt-decrypt-encrypt (EDE) sequence.

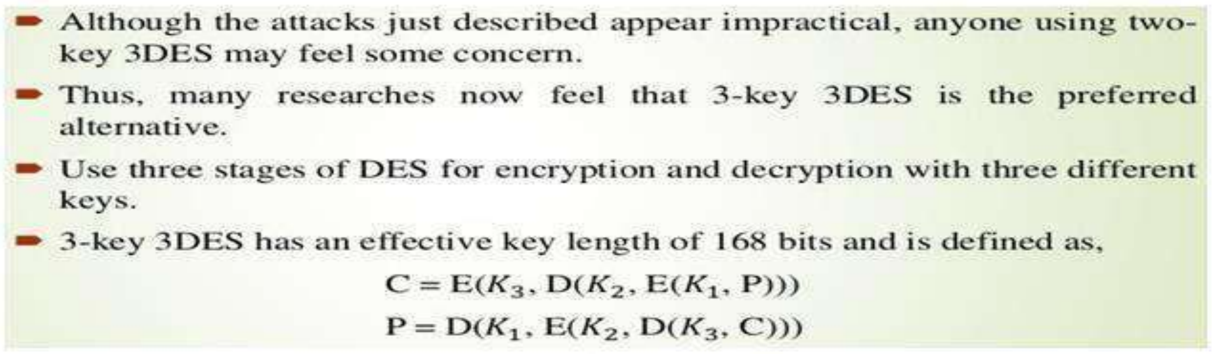
$$C = E(K_1, D(K_2, E(K_1, P)))$$
$$P = D(K_1, E(K_2, D(K_1, C)))$$

- By the use of triple DES with 2-key encryption, it raises the cost of meet-in-the-middle attack to 2^{112}.
- It has the drawback of requiring a key length of $56 \times 3 = 168$bits which may be somewhat unwieldy.

3-DES with 3 Keys:

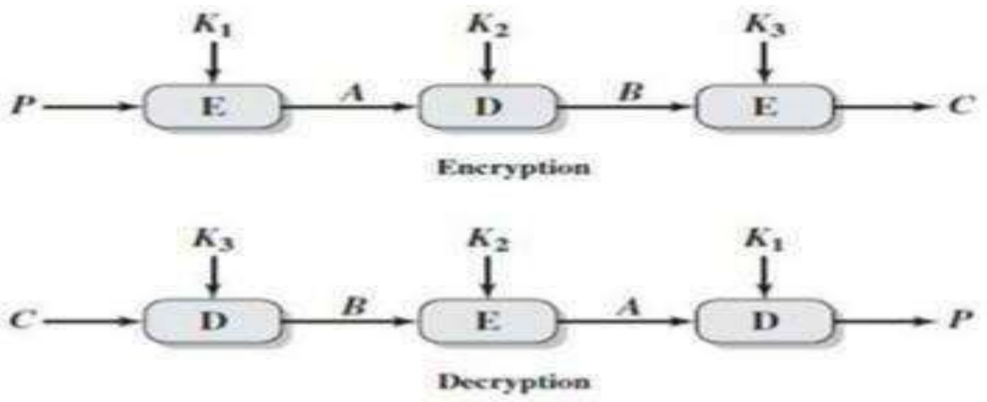


The encryption-decryption process is as follows −
Encrypt the plaintext blocks using single DES with key K_1.
Now decrypt the output of step 1 using single DES with key K_2.
Finally, encrypt the output of step 2 using single DES with key K_3.
The output of step 3 is the ciphertext.
Decryption of a ciphertext is a reverse process. User first decrypt using K_3, then encrypt with K_2, and finally decrypt with K_1.



- Although the attacks just described appear impractical, anyone using two-key 3DES may feel some concern.
- Thus, many researches now feel that 3-key 3DES is the preferred alternative.
- Use three stages of DES for encryption and decryption with three different keys.
- 3-key 3DES has an effective key length of 168 bits and is defined as,

$$C = E(K_3, D(K_2, E(K_1, P)))$$
$$P = D(K_1, E(K_2, D(K_3, C)))$$



Advanced Encryption Standard (AES Algorithm)

- The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001.
- AES is a symmetric block cipher that is intended to replace DES.

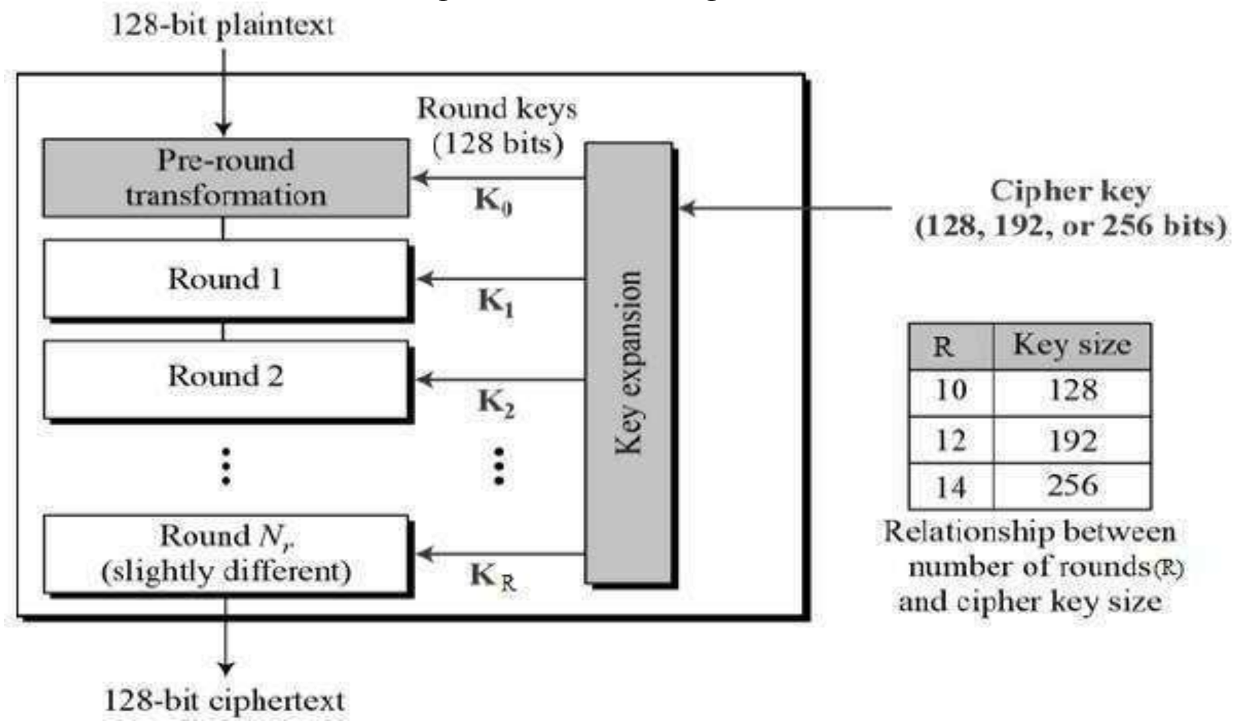
The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).
Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix.

Unlike DES, the number of rounds in AES is variable and depends on the length of the key.
AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration

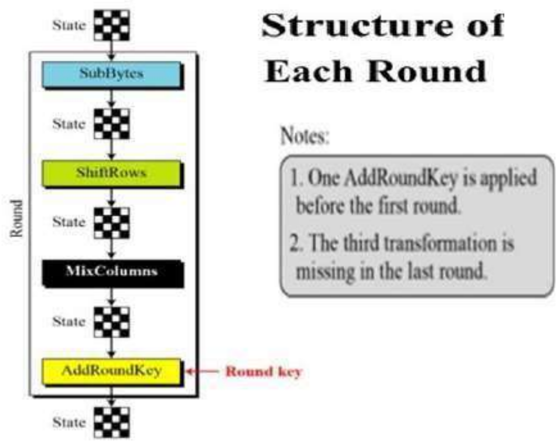


R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

ROUNDS

- Unlike DES, the number of rounds in AES is variable and depends on the length of the key.
- AES uses 10 rounds for 128-bit keys,
- 12 rounds for 192-bit keys and
- 14 rounds for 256-bit keys.
- Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.


Structure of Each Round
State
SubBytes
State
ShiftRows
State
MixColumns
State
AddRoundKey
Round key
State
Round
Notes:
1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

Each round comprise of four sub-processes. The first round process is depicted below −

AES Transformations:

There are four transformation functions used in AES Cipher at each round.

1. Substitute Bytes Transformation

2. ShiftRows Transformation
3. MixColumns Transformation
4. AddRoundKey Transformation

1. Byte Substitution (SubBytes)

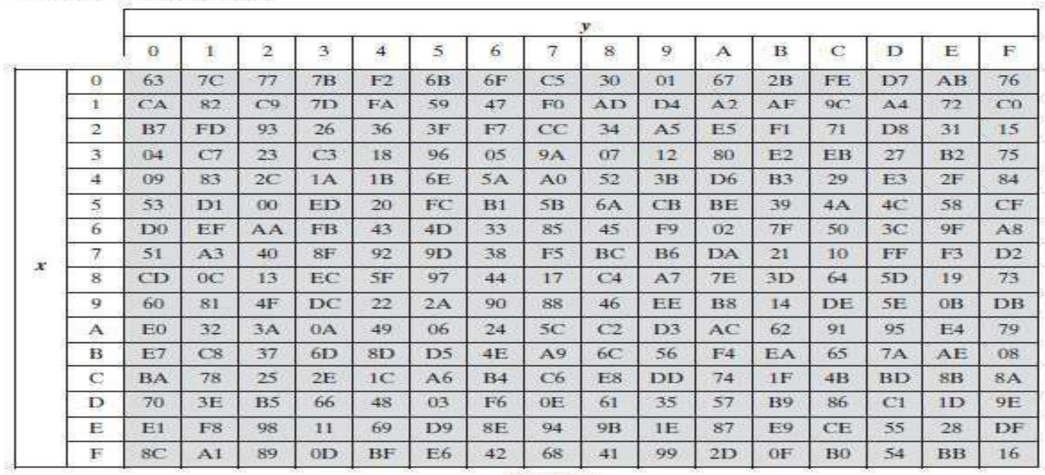
The 16 input bytes are substituted by values as specified in a table
(S-box) given in design.

Each input byte of **State** is mapped into a new byte in the following way:

- The leftmost 4 bits of the byte are used as a row value(in hexadecimal form) and the rightmost 4 bits are used as a column value(in hexadecimal form) in S-box table.

For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2A}. Accordingly, the value {95} is mapped into the value {2A}.

Table 5.2 AES S-Boxes



		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
x	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(a) S-box

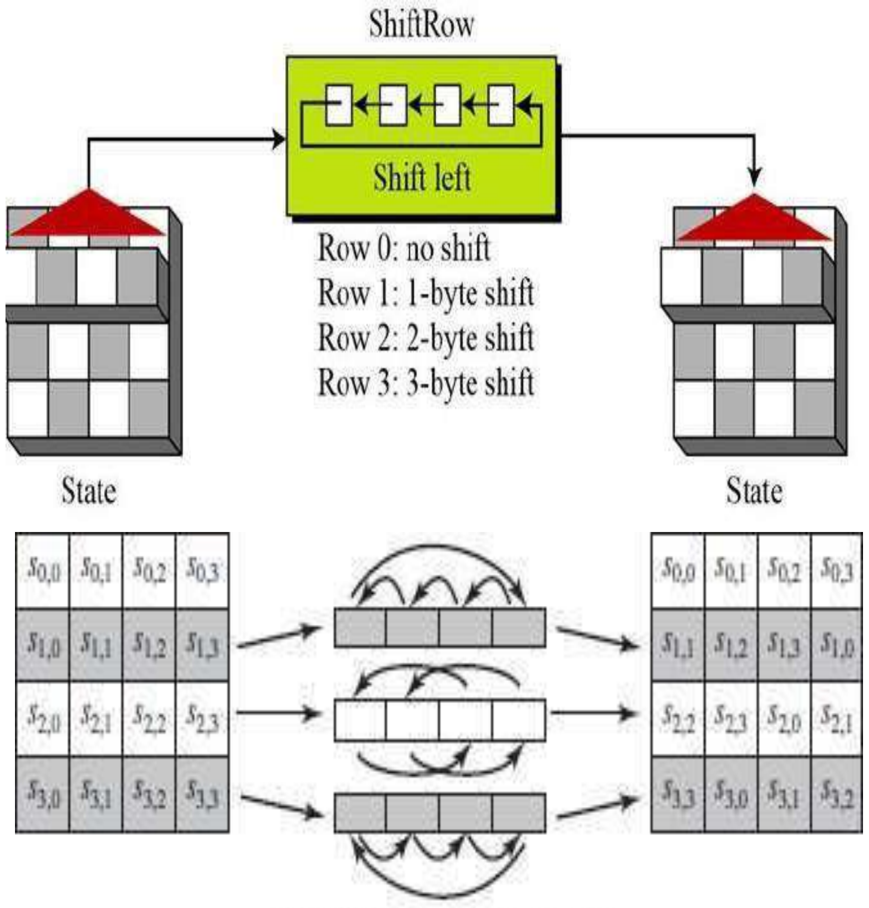
Here is an example of the SubBytes transformation:

EA	04	65	85		87	F2	4D	97
83	45	5D	96	→	EC	6E	4C	90
5C	33	98	B0		4A	C3	46	E7
F0	2D	AD	C5		8C	D8	95	A6

The S-box is constructed in the following fashion (Figure 5.6a).

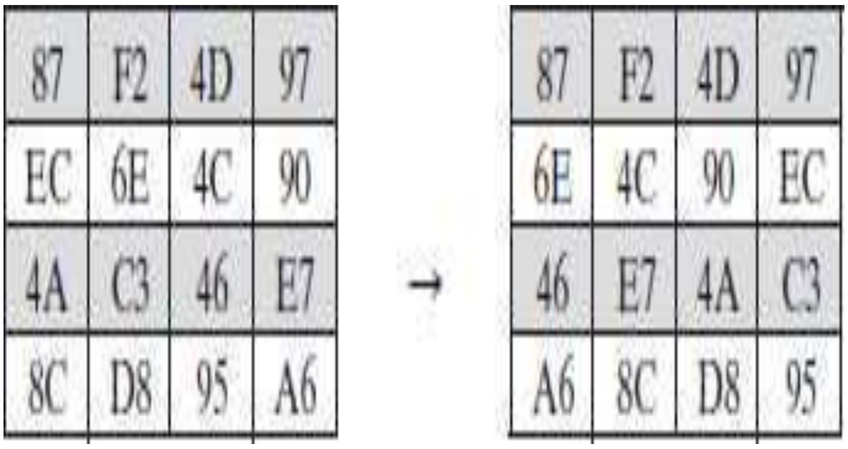
2. ShiftRows Transformation:

- ❑ In this transformation **bytes** are permuted(shifted).
- ❑ In the Encryption, the tranformation is called **Shiftrows** and the shifting is to the left.
- ❑ The number of shifts depends on the row number(0,1,2,or 3) of the state matrix as shown below:



(a) Shift row transformation

The following is an example of ShiftRows.



87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

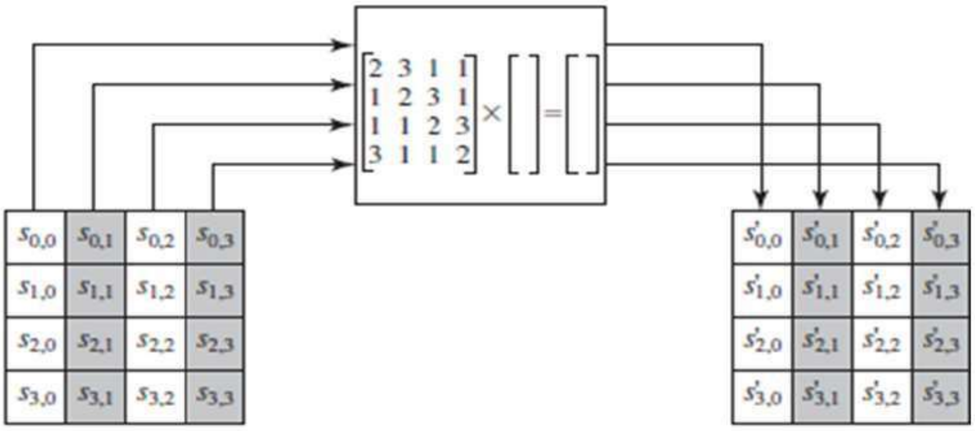
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

The **inverse shift row transformation**, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.

3. MixColumns Transformation:

Mixing is the transformaton that changes bits inside byte.
This operation takes 4 bytes(a column) and by multiplying it with a constant matrix then mixes them that produces new bytes.
MixColumn: operates on each column individually. Each byte of a column is mapped into a new value.

It takes a column from state and multiply it with a constant square matrix.
The byte values are represented as polynomials with coefficients in GF(2) and mulitplications are done in GF(2^8)


2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Constant matrices for multiplications:

Figure *Constant matrices used by MixColumns and InvMixColumns*

$$\underset{C}{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}} \xleftrightarrow{\text{Inverse}} \underset{C^{-1}}{\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}}$$

The following is an example of MixColumns:

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

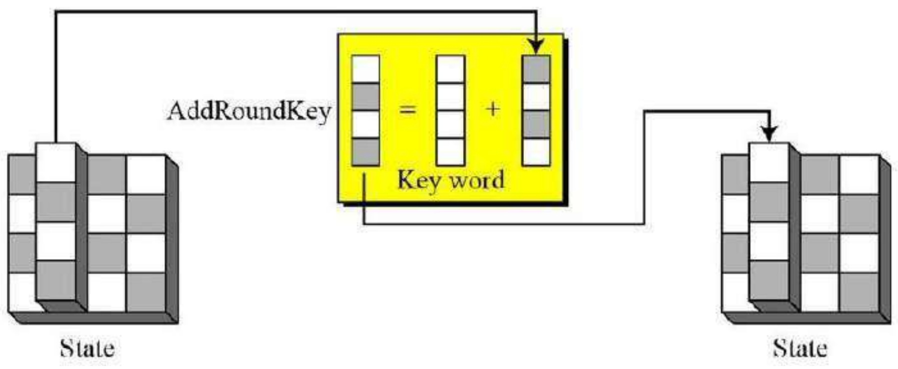
$\rightarrow$

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

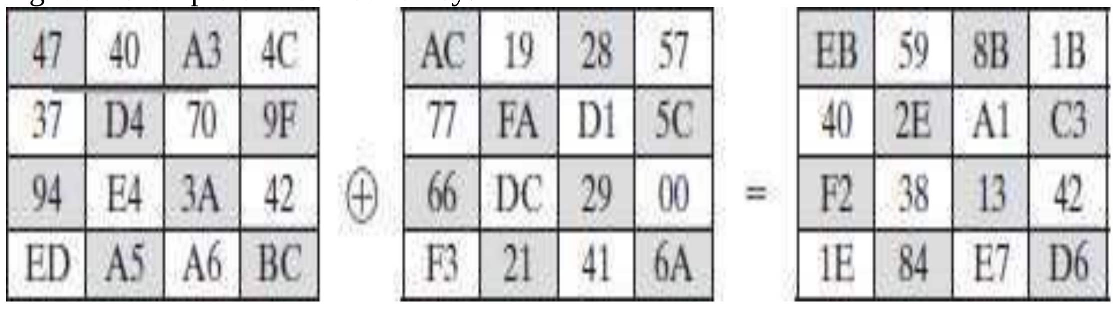
4. AddRoundKey Transformation:

- To make the ciphertext more secrete, we add cipher key to the data in a state.
- AddRoundKey is same as to MixColumns but performs addition operation instead of multiplication.

AddRoundKey transformation



The following is an example of AddRoundKey:

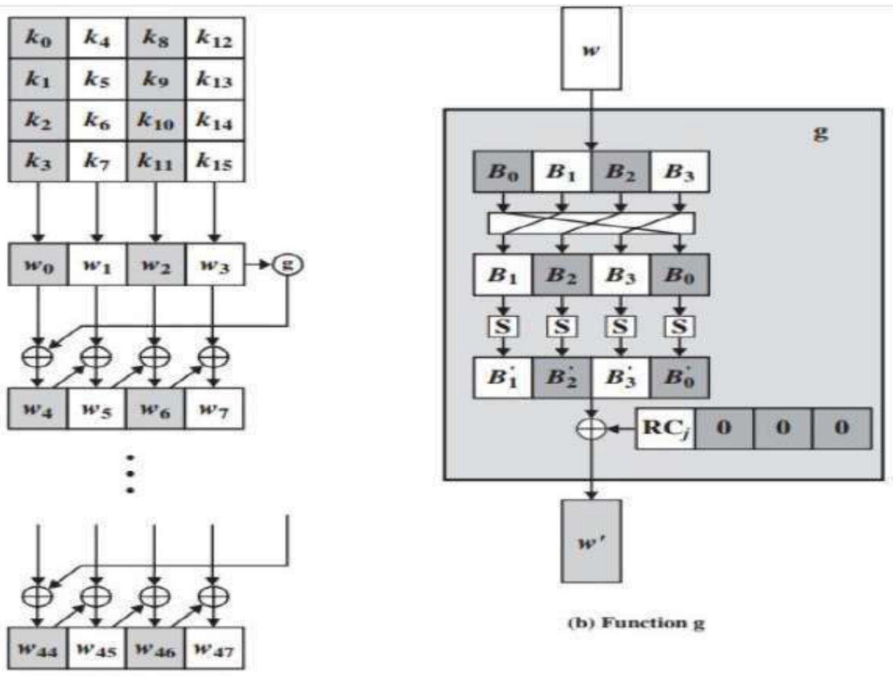


$$\begin{array}{|c|c|c|c|}\hline 47 & 40 & A3 & 4C \\ \hline 37 & D4 & 70 & 9F \\ \hline 94 & E4 & 3A & 42 \\ \hline ED & A5 & A6 & BC \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline AC & 19 & 28 & 57 \\ \hline 77 & FA & D1 & 5C \\ \hline 66 & DC & 29 & 00 \\ \hline F3 & 21 & 41 & 6A \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline EB & 59 & 8B & 1B \\ \hline 40 & 2E & A1 & C3 \\ \hline F2 & 38 & 13 & 42 \\ \hline 1E & 84 & E7 & D6 \\ \hline \end{array}$$

The first matrix is **State**, and the second matrix is the round key.

AES Key Expansion:

- The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.
- The key is copied into the first four words of the expanded key. The remain- der of the expanded key is filled in four words at a time. Each added word **w**[i] depends on the immediately preceding word, **w**[i - 1], and the word four positions back, w[i - 4]. In three out of four cases, a simple XOR is used.
- For a word whose position in the **w** array is a multiple of 4, a more complex function is used. Figure illustrates the generation of the expanded key, using the symbol g to represent that complex function. The function g consists of the following subfunctions.


k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7
w_{44} w_{45} w_{46} w_{47}
w
g
B_0 B_1 B_2 B_3
B_1 B_2 B_3 B_0
S S S S
B_1' B_2' B_3' B_0'
RC_j 0 0 0
w'

(a) Overall algorithm

(b) Function g

Figure AES Key Expansion

1. RotWord performs a one-byte circular left shift on a word. This means that an input word [B_0, B_1, B_2, B_3] is transformed into [B_1, B_2, B_3, B_0].
2. SubWord performs a byte substitution on each byte of its input word, using the S-box (Table 5.2a).
3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j].

The round constant is a word in which the three rightmost bytes are always 0. Thus, the effect of an XOR of a word with Rcon is to only perform an XOR on the left-most byte of the word. The round constant is different for each round and is defined as Rcon[j] = (RC[j], 0, 0, 0), with RC[1] = 1, RC[j] = 2 • RC[j−1] and with multiplication defined over the field GF(2^8). The values of RC[j] in hexadecimal are

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

For example, suppose that the round key for round 8 is

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

Then the first 4 bytes (first column) of the round key for round 9 are calculated as follows:

i (decimal)	temp	After RotWord	After SubWord	Rcon (9)	After XOR with Rcon	w[i−4]	w[i] = temp ⊕ w[i−4]
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

ANALYSIS OF AES

Security

- *AES was designed after DES. Most of the known attacks on DES were already tested on AES.*
- *Brute-Force Attack*
- *AES is definitely more secure than DES due to the larger-size key.*
- *Statistical Attacks*
- *Numerous tests have failed to do statistical analysis of the ciphertext.*
- *Differential and Linear Attacks*
- *There are no differential and linear attacks on AES as yet.*

Implementation

- *AES can be implemented in software, hardware, and firmware. The implementation can use table lookup process or routines that use a well-defined algebraic structure.*

Simplicity and Cost

- *The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.*

UNIT –III

Asymmetric Encryption

Mathematics of Asymmetric Key Cryptography, Asymmetric Key Cryptography

Primes and Related Congruence Equations

PRIMES

Asymmetric-key cryptography uses prime numbers extensively.
A prime is divisible only by itself and 1.

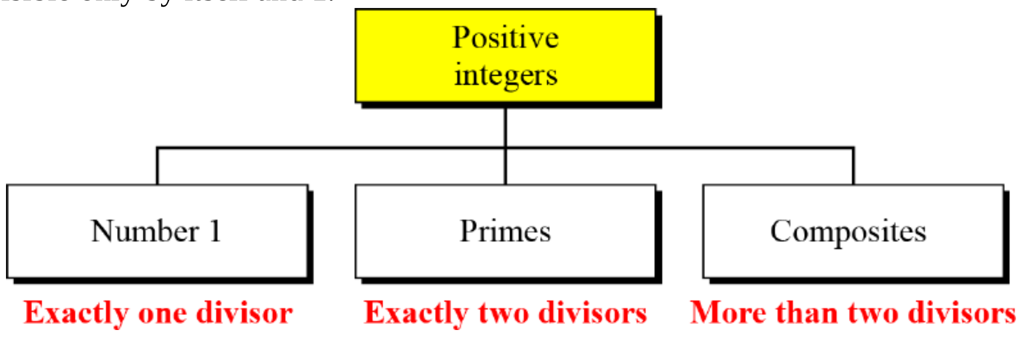


Figure Three groups of positive integers

Example 1:
What is the smallest prime?
The smallest prime is 2, which is divisible by 2 (itself) and 1.

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

Cardinality of Primes

We can use infinite Number of Primes.

Number of Primes

π(x) is the number of primes less than or equal to x. π is not similar to mathematics π.
The primes under 25 are 2, 3, 5, 7, 11, 13, 17, 19 and 23 so π(3) = 2, π(10) = 4 and π(25) = 9.

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

A Table of values of $\pi(x)$

n	x	π(x)
1	10	4
2	100	25
3	1,000	168
4	10,000	1,229
5	100,000	9,592
6	1,000,000	78,498
7	10,000,000	664,579
8	100,000,000	5,761,455

Example 1

Find the number of primes less than 1,000,000.
The approximation gives the range 72,383 to 78,543.

The actual number of primes is 78,498.

Checking for Primeness

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.

Theorem

If n is composite, then n has a prime divisor less than or equal to $\sqrt{n}$.

Example 1:
Is 97 a prime?
The floor of $\pi(97) = 9$. 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.
Example 2:
Is 301 a prime?
The floor of $\pi(301) = 17$. We need to check 2, 3, 5, 7, 11, 13, and 17. The numbers 2, 3, and 5 do not divide 301, but 7 does. Therefore 301 is not a prime.

Fermat's Little Theorem

First Version: if p is prime and a is positive integer, then

$$a^{p-1} \equiv 1 \bmod p$$

Second Version:

$$a^{p} \equiv a \bmod p$$

This means that if we divide a^p by p then the remainder should be 'a'.

Example 1:
Find the result of 6^{10} mod 11.
We have 6^{10} mod 11 = 1. This is the first version of Fermat's little theorem where $p = 11$.
Example 2
Find the result of 3^{12} mod 11.
Here the exponent (12) and the modulus (11) are not the same. With substitution this can be solved 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

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

Example
The answers to multiplicative inverses modulo a prime can be found without using the extended Euclidean algorithm:

a. $8^{-1} \bmod 17 = 8^{17-2} \bmod 17 = 8^{15} \bmod 17 = 15 \bmod 17$

b. $5^{-1} \bmod 23 = 5^{23-2} \bmod 23 = 5^{21} \bmod 23 = 14 \bmod 23$

c. $60^{-1} \bmod 101 = 60^{101-2} \bmod 101 = 60^{99} \bmod 101 = 32 \bmod 101$

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

Example:

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

Solution:

1. 5^{-1} mod 23 = 5^{23-2} mod 23 (Ref: a^{-1} mod p= a^{p-2} mod p)
2. 5^{23-2} mod 23 = 5^{21} mod 23
3. Calculate following to solve 5^{21} mod 23:

   5^1 mod 23 = 5

   5^2 mod 23=25 mod 23=2

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

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

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

Now binary equivalence of 21 is 10101, so multiply 5^1 , 5^4 and 5^{16} values, leave 5^2 and 5^8 because these are 0's in binary form.

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

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

Euler's totient function

Euler's totient function, also known as **phi-function** $\phi(n)$, this function counts the number of integers that are both smaller than n and relatively prime to n (coprime). Two numbers are coprime if their greatest common divisor equals 1.

Here are values of $\phi(n)$ for the first few positive integers:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\phi(n)$	0	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8	16	6	18	8

Example: Find co-primes of 9?

If we check gcd(9,1), gcd(9,2), gcd(9,4), gcd(9,5), gcd(9,7), gcd(9,8) =1,

So, coprimes to 9 are 1,2,4,5,7,8 and their count ϕ(9)=6

Properties

- $\phi(1)=0$
- If p is a prime number, $\phi(p)=p-1$
- If a and b are relatively prime, then: $\phi(ab)=\phi(a)\cdot\phi(b)$.
- If p is a prime, $\phi(p^e)=p^e - p^{e-1}$

Examples:

1) Find $\phi(7)$?

   $\phi(7)$=7-1=6

2)Find $\phi(21)$?

$\phi(21)= \phi(3x7) = \phi(3)x \phi(7)$=2x6=12

3)Find $\phi(77)$?

$\phi(77)= \phi(7x11) = \phi(7)x \phi(11)$=6x10=60

4) Find $\phi(3^2)$?

$\phi(3^2) = (3^2) - (3^{2-1}) = 9-3=6$

5) What is the value of ϕ (13)?

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

6)What is the value of ϕ (10)?

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

7)What is the value of ϕ (240)?

We can write $240 = 2^4 \times 3^1 \times 5^1$. Then

$$\phi(240) = (2^4 - 2^3) \times (3^1 - 3^0) \times (5^1 - 5^0) = 64$$

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

No. The third rule applies when *m* and *n* are relatively prime. Here $49 = 7^2$. We need to use the fourth rule: $\phi(49) = 7^2 - 7^1 = 42$.

9)What is the number of elements in $Z_{14}*$?

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

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

Euler's Theorem

First Version:For every a and n, they are relatively prime then

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

Second Version

$$\mathbf{a^{k \times f(n) + 1} \equiv a \pmod n}$$

Note: The second version of Euler's theorem is used in the RSA cryptosystem.

Example: if a=3 and n=10, show that $3^{\phi(10)} \equiv 1 \bmod 10$

$\phi(10)=\phi(2)x\phi(5)=1x4=4$

$3^{\phi(10)}=3^4=81$

$3^{\phi(10)} \equiv 1 \bmod 10$

$81 \equiv 1 \bmod 10$ is true because $81 \bmod 10 = 1$

Example 2:

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

Solution

We have $6^{24} \bmod 35 = 6^{\phi(35)} \bmod 35 = 1$.

Example :

Find 3^4 mod 10 ?

Solution

We have $3^4 = 3^{\phi(10)} \bmod 10 = 1$ because $\phi(10)= \phi(2)x\ \phi(5)=1x4=4$

Example 3:

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

Solution

If we let $k = 1$ on the second version,

we have f(77)= f(7)x f(11)=6x10=60

$20^{62} \bmod 77 = (20 \bmod 77)\ (20^{60+1} \bmod 77) \bmod 77=$

$(20 \bmod 77)\ (20^{f(77)+1} \bmod 77) \bmod 77$

$= (20)(20) \bmod 77 = 15.$

Multiplicative Inverses

Euler's theorem can be used to find multiplicative inverses modulo a composite.

$$\boxed{a^{-1} \bmod n = a^{\phi(n)-1} \bmod n}$$

Example:
The answers to multiplicative inverses modulo a composite can be found without using the extended Euclidean algorithm if we know the factorization of the composite:

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

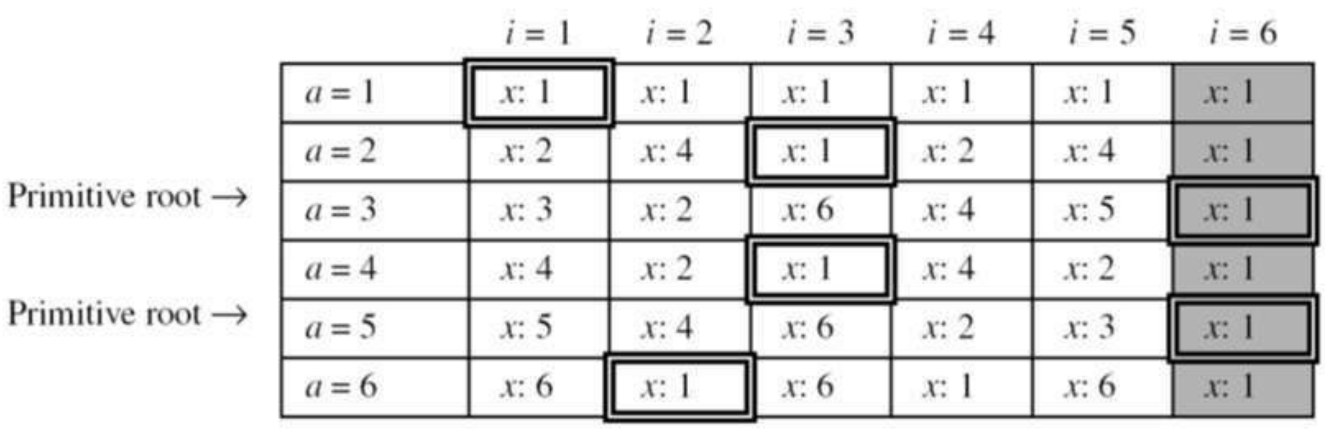
Primitive Root and Multiplicative Orders

Multiplicative Order:

If 'a' and 'n' are relatively prime, then
The multiplicative order of ‘a’ modulo n is smallest positive integer 'k' with
$a^k \equiv 1 \pmod n$
The order of modulo ‘n’ is written as $ord_n(a)$ or $O_n(a)$
Example 1: Define multiplicative order of 4 mod 7
$4^1=4 \equiv 3 \pmod 7$
$4^2=16 \equiv 2 \pmod 7$
$4^3=64 \equiv 1 \pmod 7$
$Ord_7(4)=3$ because 4^3 is congruent to 1 modulo 7.
Example 2: Define multiplicative order of 2 mod 7
$2^1=2 \equiv 2 \pmod 7$
$2^2=4 \equiv 4 \pmod 7$
$2^3=8 \equiv 1 \pmod 7$
$Ord_7(2)=3$ because 2^3 is congruent to 1 modulo 7.

Primitive Root :

***Primitive Roots** In the group $G = <Z_n*, \times>$, when the order of an element is the same as $\phi(n)$, that element is called the primitive root of the group.*
Table shows the result of $a^i \equiv x \pmod 7$ for the group $G = <Z_7*, \times>$. In this group, $\phi(7) = 6$.



		$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
	$a = 1$	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
	$a = 2$	x: 2	x: 4	x: 1	x: 2	x: 4	x: 1
Primitive root →	$a = 3$	x: 3	x: 2	x: 6	x: 4	x: 5	x: 1
	$a = 4$	x: 4	x: 2	x: 1	x: 4	x: 2	x: 1
Primitive root →	$a = 5$	x: 5	x: 4	x: 6	x: 2	x: 3	x: 1
	$a = 6$	x: 6	x: 1	x: 6	x: 1	x: 6	x: 1

The order of elements are ord(1)=1, ord(2)=3,ord(3)=6, ord(4)=3, ord(5)=6, ord(6)=2. The elements 3 and 5 have the order at i= ϕ(7)=6.
Therefore elements 3 and 5 are primitive roots.

The group G = <Z_n*, ×> has primitive roots only if n is 2, 4, p^t, or $2p^t$.

If the Group G=<Z_n^*,x> has any primitive root, the number of primitive roots is

ϕ(ϕ (n))

Example: Find the Number of primitive roots of 25
ϕ (25)=20
Find the primitive root of 761
ϕ (ϕ (761))= ϕ (760)
= ϕ (2^3x5x19) = ϕ (2^3)x ϕ (5)x ϕ (19)
=(2^3 - 2^2)x 4x18=4x4x18
=288

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:

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

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

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

Example:
Find the solution to the simultaneous equations:

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

Solution:
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. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 = 23 \bmod 105$

Example 2:
Find an integer that has a remainder of 3 when divided by 7 and 13, but is divisible by 12.
Solution
This is a CRT problem. We can form three equations and solve them to find the value of x.

$$\begin{aligned} x &= 3 \bmod 7 \\ x &= 3 \bmod 13 \\ x &= 0 \bmod 12 \end{aligned}$$

If we follow the four steps, we find x = 276. We can check that
276 = 3 mod 7, 276 = 3 mod 13 and 276 is divisible by 12 (the quotient is 23 and the remainder is zero).

Example 3
Assume we need to calculate $z = x + y$ where $x = 123$ and $y = 334$, but our system accepts only numbers less than 100.

$$\begin{aligned} x &\equiv 24 \pmod{99} & \quad y &\equiv 37 \pmod{99} \\ x &\equiv 25 \pmod{98} & \quad y &\equiv 40 \pmod{98} \\ x &\equiv 26 \pmod{97} & \quad y &\equiv 43 \pmod{97} \end{aligned}$$

Adding each congruence in *x* with the corresponding congruence in *y* gives

$$\begin{aligned} x + y &\equiv 61 \pmod{99} & \rightarrow \quad z &\equiv 61 \pmod{99} \\ x + y &\equiv 65 \pmod{98} & \rightarrow \quad z &\equiv 65 \pmod{98} \\ x + y &\equiv 69 \pmod{97} & \rightarrow \quad z &\equiv 69 \pmod{97} \end{aligned}$$

Now three equations can be solved using the Chinese remainder theorem to find z. One of the acceptable answers is $z = 457$.

QUADRATIC CONGRUENCE

Quadratic Congruence is a congruence of the equation 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 equation of the form.

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

There are two ways:

1. Quadratic Congruence Modulo a Prime
2. Quadratic Congruence Modulo a Composite

<u>Quadratic Congruence Modulo a Prime</u>

In this, we consider the modulus is a prime number. That is the form. **$x^2 \equiv a$ (mod p)**

Where p is a prime and ‘a’ is an integer.

Example 1: Solve the **$x^2 \equiv 3$ (mod 11)**

Solution: 3 congruent to modulo 11 are 3,14,25 (25 is 5x5 or (-5)x(-5))

The given equation has two solutions:

$x^2 \equiv 25$ (mod 11)

$x \equiv 5$ (mod 11) and $x \equiv -5$ (mod 11),

But $-5 \equiv 6$ (mod 11)

So, the solutions are 5 and 6

Check the result: substitute x=5

$5^2 \equiv 25 = 3$ (mod 11)

substitute x=6

$6^2 \equiv 36 = 3$ (mod 11)

Example 2: Solve the **$y^2 \equiv 10$ (mod 13)**

Solution: The number 10 congruent to 13 are 10,23,36 (36 is 6x6 or (-6)x(-6))

The given equation has two solutions:

$x \equiv$ **6 (mod 13) and $x \equiv -6$ (mod 13),**

But $-6 \equiv 7$ (mod 13)

So, the solutions are 6 and 7

Check the result: substitute x=6

$6^2 \equiv 36 \equiv 10$ (mod 13)

substitute x=7

$7 \equiv 49 \equiv 10$ (mod 13)

<u>Quadratic Congruence Modulo a Composite</u>

Quadratic Congruence Modulo a Composite can be solved by set of Quadratic Congruence Modulo a Prime.

Decomposition of congruence modulo a composite:

$$\begin{tikzcd}
\boxed{\begin{array}{l} x^2 \equiv a \bmod (n) \\ n = p_1 \times p_2 \times \ldots \times p_k \end{array}} \arrow[r] & \boxed{\begin{array}{c} x^2 \equiv a_1 \pmod{p_1} \\ x^2 \equiv a_2 \pmod{p_1} \\ \bullet\bullet\bullet \\ x^2 \equiv a_k \pmod{p_k} \end{array}} \arrow[r] & \boxed{\begin{array}{c} x_1 \equiv \pm b_1 \pmod{p_1} \\ x_2 \equiv \pm b_2 \pmod{p_1} \\ \bullet\bullet\bullet \\ x_k \equiv \pm b_k \pmod{p_k} \end{array}}
\end{tikzcd}$$

Example: Assume that $x^2 \equiv 36$ (mod 77).

We know that 77 = 7 × 11. We can write

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

The answers are $x \equiv +1$ (mod 7), $x \equiv -1$ (mod 7),

$x \equiv +5$ (mod 11), and $x \equiv -5$ (mod 11). Now we can make four sets of equations out of these:

$$\begin{array}{lll}
\textbf{Set 1:} & x \equiv +1 \pmod{7} & x \equiv +5 \pmod{11} \\
\textbf{Set 2:} & x \equiv +1 \pmod{7} & x \equiv -5 \pmod{11} \\
\textbf{Set 3:} & x \equiv -1 \pmod{7} & x \equiv +5 \pmod{11} \\
\textbf{Set 4:} & x \equiv -1 \pmod{7} & x \equiv -5 \pmod{11}
\end{array}$$

The answers are x = ± 6 and ± 27.

ASYMMETRIC KEY /PUBLIC KEY CRYPTOGRAPHY

Asymmetric key cryptosystems / public-key cryptosystems use a pair of keys: public key (encryption key) and private key (decryption key).

<u>Public Key Cryptography ?</u>

- Public key cryptography also called as **asymmetric cryptography**.
- It was invented by whitfield **Diffie** and Martin **Hellman** in 1976. Sometimes this cryptography also called as **Diffie-Helman Encryption**.
- Public key algorithms are based on mathematical problems which admit no efficient solution that are inherent in certain integer factorization, discrete logarithm and Elliptic curve relations.

Public key Cryptosystem Principles:

- The concept of public key cryptography is invented for two most difficult problems of Symmetric key encryption.
  - The Key Exchange Problem
  - The Trust Problem

The Key Exchange Problem: The key exchange problem arises from the fact that communicating parties must somehow share a secret key before any secure communication can be initiated, and both parties must then ensure that the key remains secret. Of course, direct key exchange is not always feasible due to risk, inconvenience, and cost factors.

The Trust Problem: Ensuring the integrity of received data and verifying the identity of the source of that data can be very important. Means in the symmetric key cryptography system, receiver doesn‟t know whether the message is coming for particular sender.

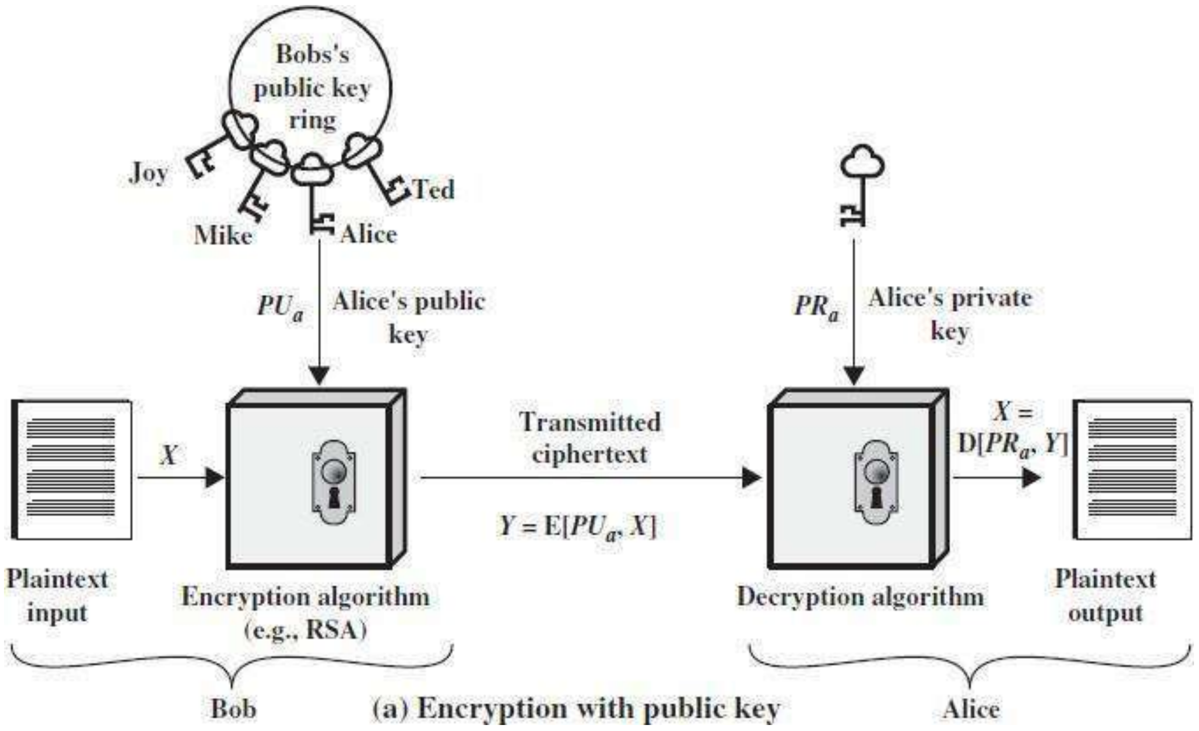
- This public key cryptosystem uses two keys as pair for encryption of plain text and Decryption of cipher text.
- These two keys are names as "**Public key**" and "**Private key**". The private key is kept secret where as public key is distributed widely.
- A message or text data which is encrypted with the public key can be decrypted only with the corresponding private-key

This two key system very useful in the areas of confidentiality (secure) and authentication

A **public-key encryption** scheme has six ingredients		
1	**Plaintext**	This is the readable message or data that is fed into the algorithm as input.
2	**Encryption algorithm**	The encryption algorithm performs various transformations on the plaintext.
3	**Public key**	This is a pair of keys that have been selected so that if one is used for

4	**Private key**	encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input
5	**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.
6	**Decryption algorithm**	This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

Public key cryptography for providing confidentiality (secrecy)



The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As the above Figure 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.

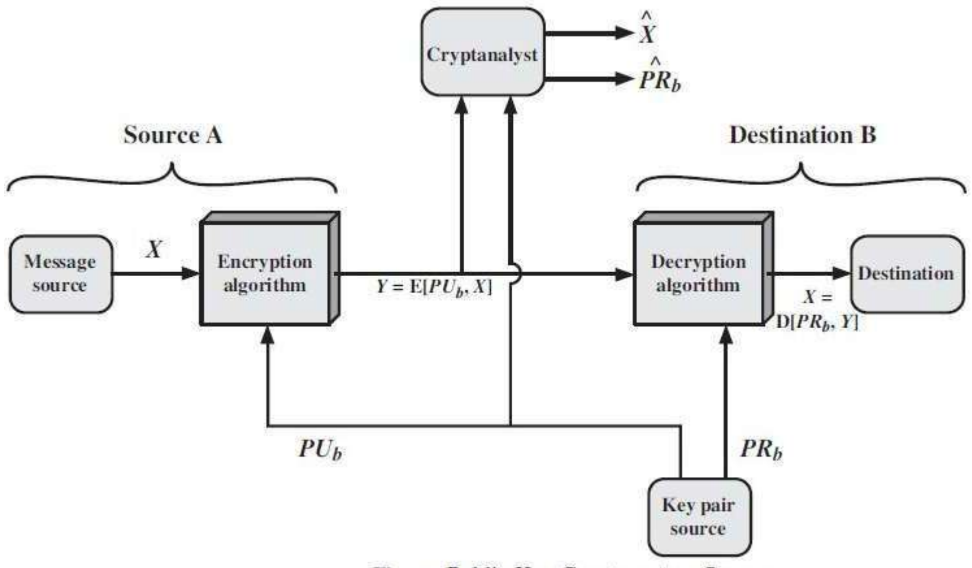


Figure Public-Key Cryptosystem: Secrecy

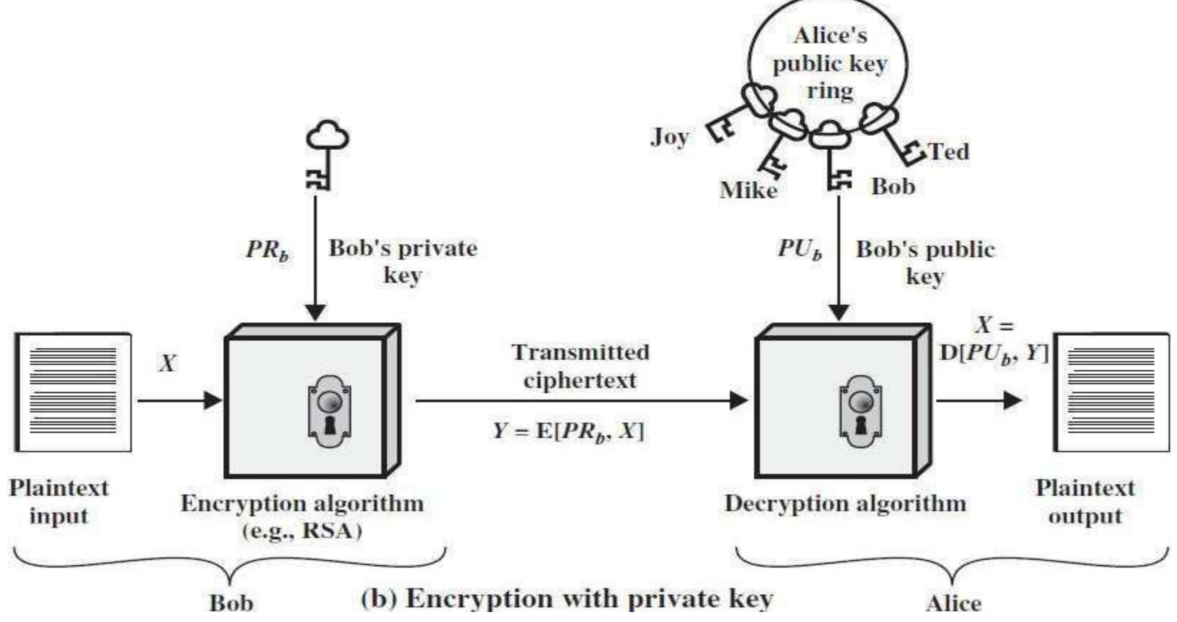
There is some source A that produces a message in plaintext $\boldsymbol{X = [X_1, X_2, \ldots, X_M]}$.
The ***M*** elements of ***X*** are letters in some finite alphabet. The message is intended for destination **B**. B generates a related pair of keys: a public key, $\boldsymbol{PU_b}$**,** and a private key, $\boldsymbol{PR_b}$.
$\boldsymbol{PR_b}$ is known only to B, whereas $\boldsymbol{PU_b}$ is publicly available and therefore accessible by A.
With the message X and the encryption key $\boldsymbol{PU_b}$ as input, A forms the ciphertext $Y = [Y_1, Y_2, \ldots, YN]$:

$$Y = \mathrm{E}(PU_b, X)$$

$$X = \mathrm{D}(PR_b, Y)$$

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

<u>Public key cryptography for proving Authentication:</u>



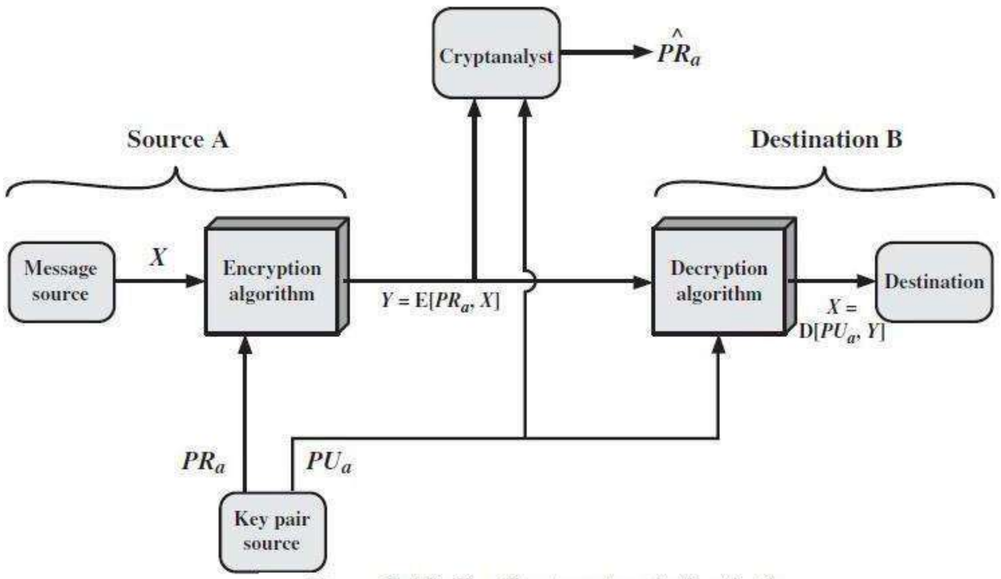


Figure Public-Key Cryptosystem: Authentication

The above diagrams show the use of public-key encryption to provide authentication:

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

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

- In this case, A prepares a message to B and encrypts it using A‟s private key before transmitting it. B can decrypt the message using A‟s public key. Because the message was encrypted using A‟s private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a **digital signature.**

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

Public key cryptography for both authentication and confidentiality (Secrecy)

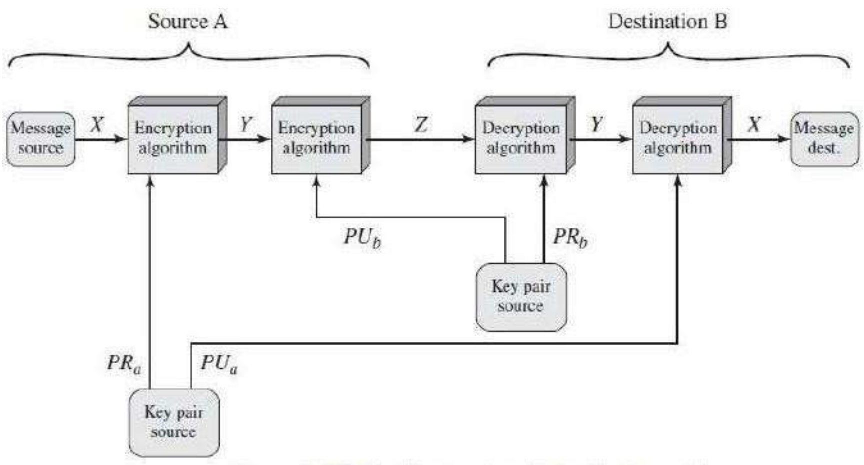


Figure Public-Key Cryptosystem: Authentication and Secrecy

It is, however, possible to provide both the authentication function and confidentiality by a double use of

$$Z = E(PU_b, E(PR_a, X))$$

$$X = D(PU_a, D(PR_b, Z))$$

the public-key scheme (above figure):
In this case, we begin as before by encrypting a message, using the sender‟s private key. This provides the digital signature. Next, we encrypt again, using the receiver‟s public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided.

Applications for Public-Key Cryptosystems

Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender‟s private key or the receiver‟s public key, or both, to perform some type of cryptographic function. 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.

Applications for Public-Key Cryptosystems

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

Public-Key Cryptanalysis

As with symmetric encryption, a public-key encryption scheme is vulnerable to a brute-force attack. The countermeasure is the same: Use large keys. However, there is a tradeoff to be considered. Public- key systems depend on the use of some sort of invertible mathematical function. The complexity of calculating these functions may not scale linearly with the number of bits in the key but grow more rapidly than that. Thus, the key size must be large enough to make brute-force attack impractical but small enough for practical encryption and decryption. In practice, the key sizes that have been proposed do make brute-force attack impractical but result in encryption/decryption speeds that are too slow for general-purpose use. Instead, as was mentioned earlier, public-key encryption is currently confined to key management and signature applications.

RSA Algorithm

- It is the most common public key algorithm.
- This RSA name is get from its inventors first letter (Rivest (R), Shamir (S) and Adleman (A)) in the year 1977.
- The RSA scheme is a block cipher in which the plaintext & 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}

Description of the Algorithm:

- RSA algorithm uses an expression with exponentials.
- In RSA plaintext is encrypted in blocks, with each block having a binary value less than some number

n. that is, the block size must be less than or equal to **$\log_2(n)$**

- RSA uses two exponents e and d where e public and d private.
- Encryption and decryption are of following form, for some PlainText M and CipherText block C

$$\boxed{C = M^e \bmod n}$$

$$\boxed{M = C^d \bmod n}$$

$$M = C^d \bmod = (M^e \bmod n)^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both sender and receiver must know the value of n.
The sender knows the value of **e** & only the receiver knows the value of **d** thus this is a public key encryption algorithm with a

Public key PU={e, n}
Private key PR={d, n}

Steps of RSA algorithm:

Step 1→Select 2 prime numbers p & q
Step 2→Calculate n=pq
Step 3→Calculate Ø(n)=(p-1)(q-1)
Step 4→ Select or find integer e (public key) which is relatively prime to Ø(n).
ie., e with gcd (Ø(n), e)=1 where 1<e< Ø(n).
Step 5→ Calculate "d" (private key) by using following condition. $\boxed{ed \equiv 1 \bmod Ø(n)}$
d< Ø(n).
Step 6→ Perform encryption by using $\boxed{C = M^e \bmod n}$
Step 7→ perform Decryption by using $\boxed{M = C^d \bmod n}$

Example:

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate $n = pq$ = 17 × 11 = 187.
3. Calculate Ø(n) = (p - 1)(q - 1) = 16 × 10 = 160.
4. Select e such that e is relatively prime to Ø(n) = 160 and less than Ø (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;
d can be calculated using the extended Euclid"s algorithm

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

The example shows the use of these keys for a plaintext input of M= 88. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

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

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

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

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

The Security of RSA

Four possible approaches to attacking the RSA algorithm are

- **Brute force:** This involves trying all possible private keys.
- **Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.
- **Timing attacks:** These depend on the running time of the decryption algorithm.
- **Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm.

Trapdoor one-way function

- A trapdoor function is a function that is easy to perform one way, but has a secret that is required to perform the inverse calculation efficiently.
- That is, if f is a trapdoor function, then y=f(x) is easy to compute, but x=f−1(y) is hard to compute without some special knowledge k. Given k, then it is easy to compute y=f−1(x,k).
- The analogy to a "trapdoor" is something like this: It's easy to fall through a trapdoor, but it's very hard to climb back out and get to where you started unless you have a ladder.
- An example of such trapdoor one-way functions may be finding the prime factors of large numbers. Nowadays, this task is practically infeasible.
- On the other hand, knowing one of the factors, it is easy to compute the other ones.

For example: RSA is a one-way trapdoor function

Diffie-Hellman Key Exchange

- Diffie-Hellman key exchange is the first published public key algorithm
- This Diffie-Hellman key exchange protocol is also known as exponential key agreement. And it is based on mathematical principles.
- The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages.
- This algorithm itself is limited to exchange of the keys.
- This algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.
- The discrete logarithms are defined in this algorithm in the way of define a primitive root of a prime number.
- **Primitive root:** we define a primitive root of a prime number P as one whose power generate all the integers from 1 to P-1 that is if **'a'** is a primitive root of the prime number P, then the numbers are distinct and consist of the integers form 1 through P-1 in some permutation.

$$a \bmod P,\ a^2 \bmod P,\ a^3 \bmod P,\ \ldots\ a^{P-1} \bmod P$$

For any integer **b** and **a**, here **a** is a primitive root of prime number P, then

$$b \equiv a^i \bmod P \qquad 0 \le i \le (P-1)$$

The exponent i → is refer as discrete logarithm or index of b for the base a, mod P.
The value denoted as **ind $_{a,p}$(b)**

Algorithm for Diffie-Hellman Key Exchange:

Step 1→ Select global public numbers q, α
q→ Prime number
α→ primitive root of q and α< q.

Step 2 → if A & B users wish to exchange a key

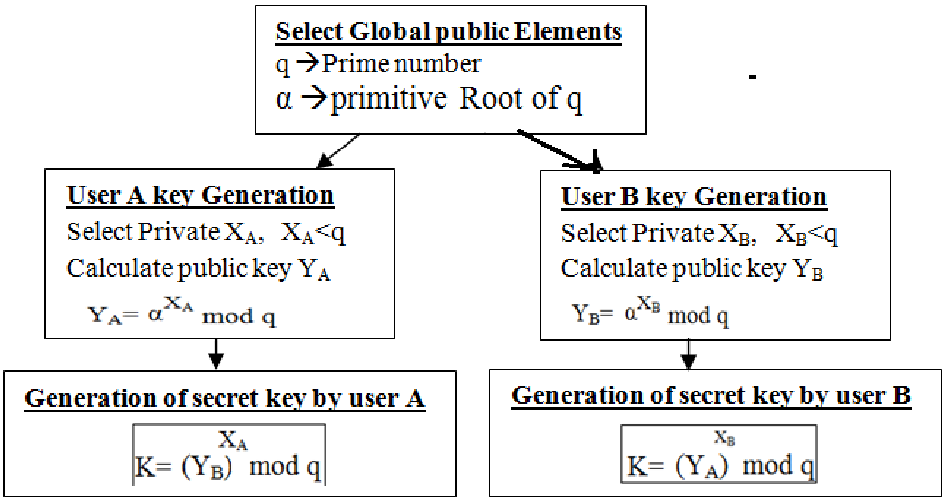
a) User A select a random integer X_A<q and computes $Y_A = \alpha^{X_A} \bmod q$

b) User B independently select a random integer X_B <q and computes $Y_B = \alpha^{X_B} \bmod q$

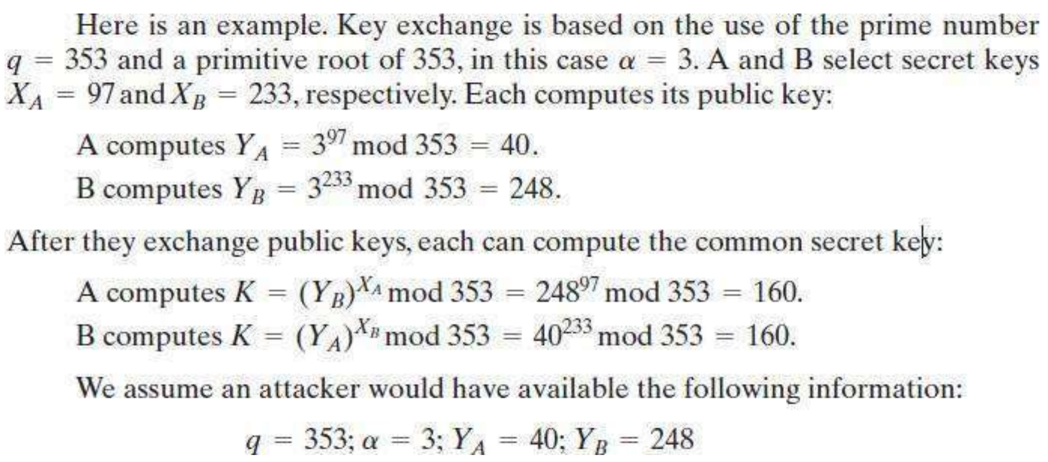
c) Each side keeps the X value private and Makes the Y value available publicly to the outer side.

Step 3→ User A Computes the key as $\boxed{K = (Y_B)^{X_A} \bmod q}$
User B Computes the key as $\boxed{K = (Y_A)^{X_B} \bmod q}$

Step 4→ two calculation produce identical results
The result is that the two sides have exchanged a secret key.



Example:



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

A computes $Y_A = 3^{97} \bmod 353 = 40$.
B computes $Y_B = 3^{233} \bmod 353 = 248$.

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

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.
B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

We assume an attacker would have available the following information:

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

MAN-in the Middle Attack (MITM)

Definition: A man in the middle attack is a form of eavesdropping where communication between two users is monitored and modified by an unauthorized party.
Generally the attacker actively eavesdrops by intercepting (stoping) a public key message exchange.

The Diffie- Hellman key exchange is insecure against a “Man in the middle attack”.
Suppose user A & B wish to exchange keys, and D is the adversary (opponent). The attack proceeds as follows.

1. D prepares for the attack by generating two random private keys X_{D1} & X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}.
2. A transmits Y_A to B
3. D intercepts Y_A and transmits Y_{D1} to B. and D also calculates $K2 = (Y_A)^{X_{D2}} \bmod q$.
4. B receives Y_{D1} & calculate $K1 = (Y_{D1})^{X_B} \bmod q$.
5. B transmits Y_B to A
6. Dintercepts Y_B and transmits Y_{D2} to „A" and „D" calculate K1 $K1 = (Y_B)^{X_{D1}} \bmod q$
7. A receives Y_{D2} and calculates $K2 = (Y_{D2})^{X_A} \bmod q$

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key *K*1 and Alice and Darth share secret key *K*2. All future communication between Bob and Alice is compromised in the following way.

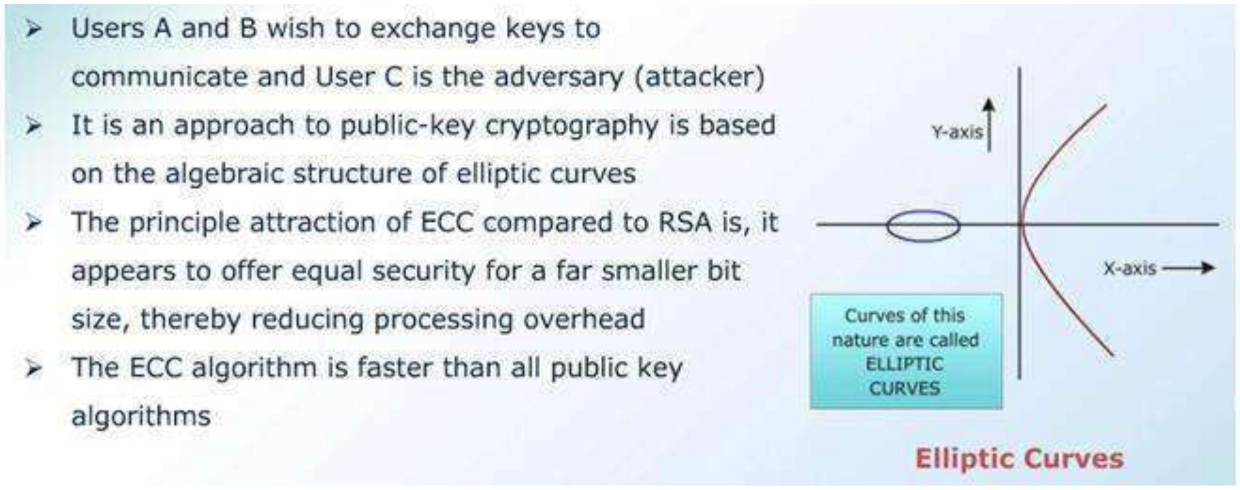
1. A sends an encrypted message *M*: E(*K*2, *M*) .
2. D intercepts the encrypted message and decrypts it to recover *M*.
3. D sends B *E*(*K*1, *M*) or E(*K*1, *M*′) , where *M*′ is any message. In the first case, D simply wants to eavesdrop on the communication without altering it. In the second case, D wants to modify the message going to B

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates.

Elliptic Curve Cryptography

- Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers
- 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



- Users A and B wish to exchange keys to communicate and User C is the adversary (attacker)
- It is an approach to public-key cryptography is based on the algebraic structure of elliptic curves
- The principle attraction of ECC compared to RSA is, it appears to offer equal security for a far smaller bit size, thereby reducing processing overhead
- The ECC algorithm is faster than all public key algorithms

Elliptic Curves

ECC-Key Exchange:

Take two Global public Elements

$E_q(a,b)$: Elliptic curve with parameters a,b, & q

G : Point on elliptic curve whose order is large value n

Alice Key Generation:

Select private key n_A : $n_A < n$

Calculate public key P_A: $P_A = n_A x G$

Bob Key Generation:

Select private key n_B : $n_B < n$

Calculate public key P_B: $P_B = n_B x G$

Secrete Key calculation by Alice

$K=n_A x P_B$

Secrete Key calculation by Bob

$K=n_B x P_A$

ECC- Encryption

- Let the message be M
- First encode the message M into a point on the elliptic curve
- Let this point be P_m
- Now this point is encrypted
- For encryption choose a random positive integer k
- Then $C_m=\{ kG, P_m+kP_B \}$ where G is the base point

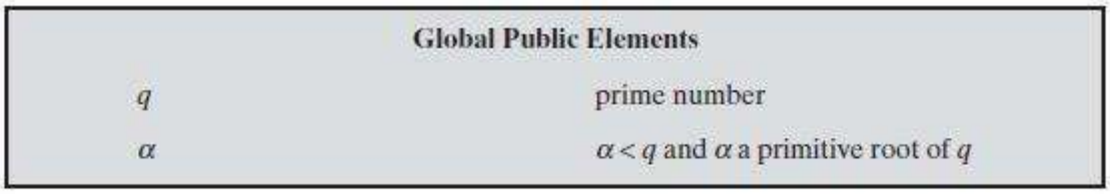
ECC-Decryption

- Multiply first point in the pair with receivers secrete key
  i.e, kG x nB
- Then subtract it from second point in the pair
  i.e, $P_m + kP_B - (kG x n_B)$

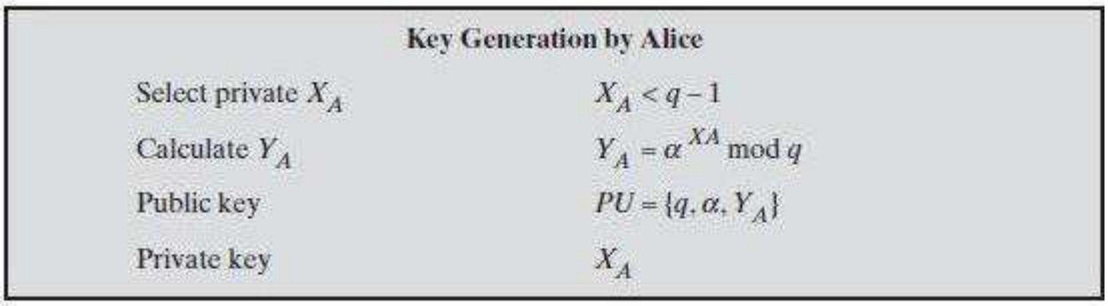
ELGAMAL CRYPTOGRAPHIC SYSTEM

- In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique.
- EIGamal Algorithms are used for both digital signatures as well as encryption.

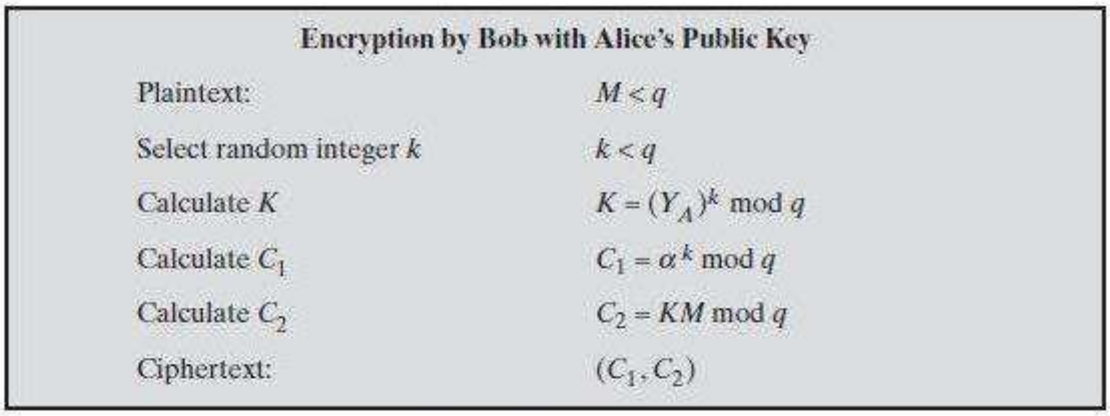
EIGamal Algorithm:-



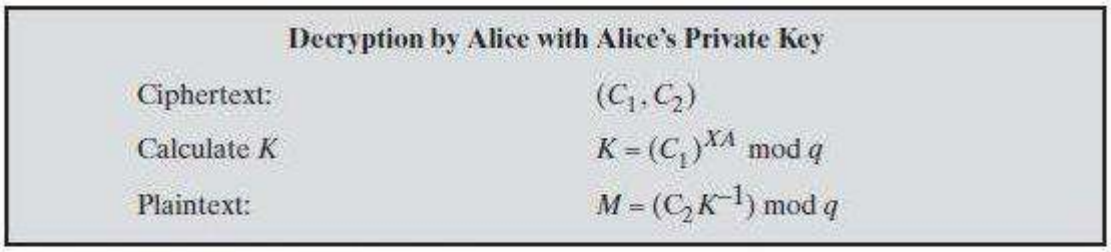
Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q



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:	$M = (C_2K^{-1}) \bmod q$

Thus, functions as a one-time key, used to encrypt and decrypt the message.
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 }. We choose $\alpha = 10$.
Alice generates a key pair as follows:

1. Alice chooses $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^5 \bmod 19 = 3$
3. Alice's private key is 5; Alice's pubic 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
   $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$
   $C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
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, $M = (C_2K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$.

RABIN CRYPTOSYSTEM

Rabin Cryptosystem is an public-key cryptosystem invented by Michael Rabin, is a variation of the RSA. RSA is based on the exponentiation congruence; Robin is based on **quadratic congruence**.
The public key in the Rabin is n, private key is the tuple(p,q). Everyone can encrypt a message using n, only Bob can decrypt the message using p and q.
Decryption of the message is infeasible It uses asymmetric key encryption for communicating between two parties and encrypting the message.
The security of Rabin cryptosystem is related to the difficulty of factorization. It has the advantage over the others that the problem on which it banks has proved to be hard as **integer factorization**.
It has the disadvantage also, that each output of the Rabin function can be generated by any of four possible inputs. if each output is a cipher text, extra complexity is required on decryption to identify which of the four possible inputs was the true plaintext.

Steps in Rabin cryptosystem

Key generation

1. Generate two very large prime numbers, p and q, which satisfies the condition
   $p \neq q \rightarrow p \equiv q \equiv 3 \pmod 4$
   For example:
   p=139 and q=191
2. n = p.q
3. Public_key=n
4. Private_key=(p,q)
5. Return public_key, Private_keys

Encryption

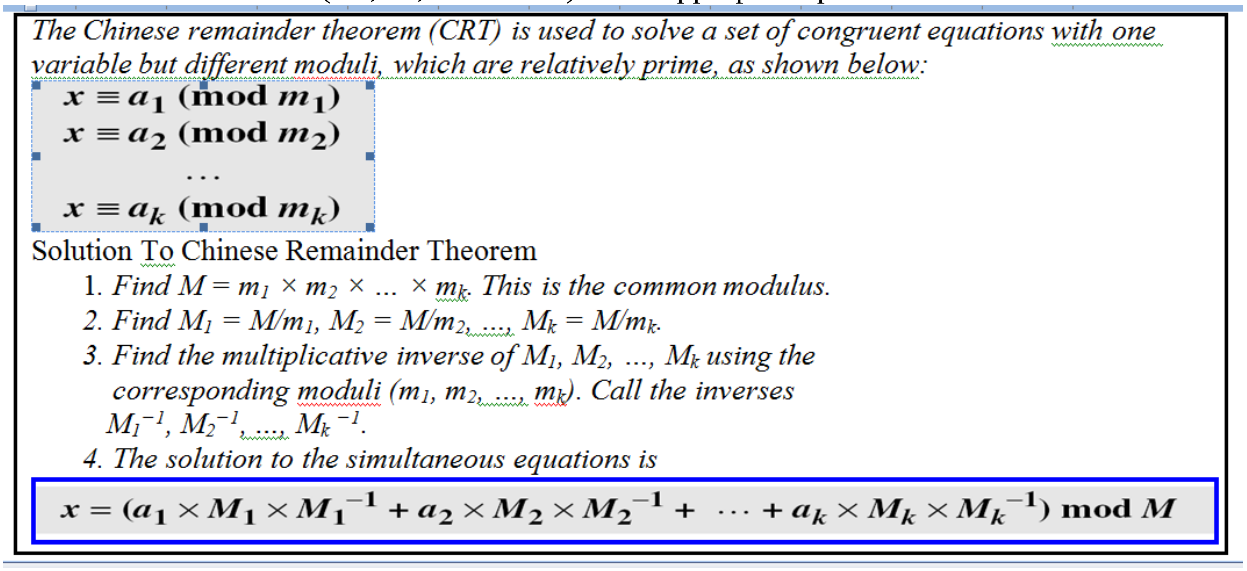
1. Get the public key n.
2. Convert the message to ASCII value. Then convert it to binary and extend the binary value with itself, and change the binary value back to decimal **M**.
3. Encrypt with the formula:
   $C = M^2 \bmod n$
4. Send C to recipient.

Decryption

1. Accept C from sender.
2. Compute:

   a1 = $C^{(p+1)/4}$ mod p
   a2= - $C^{(p+1)/4}$ mod p
   b1= $C^{(q+1)/4}$ mod q
   b2= - $C^{(q+1)/4}$ mod q
3. Calculate four Plain text by using Chinese Remainder Algorithm:

   M_1=Chainese_Remainder(a1,b1,p,q)
   M_2=Chainese_Remainder(a1,b2,p,q)
   M_3=Chainese_Remainder(a2,b1,p,q)
   M_4=Chainese_Remainder(a2,b2,p,q)
4. Choose one of the above (M_1,M_2,M_3 and M_4) is the appropriate plaintext.


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:

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

Solution To Chinese Remainder Theorem

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

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

The Rabin cryptosystem is not deterministic: Decryption creates four equally probable plain texts

Example:
1.Bob selects p=23 and q=7, note both are congruent to 3 mod 4
2.Bob calculates n=pxq=161
3.Bob announces n publickly; he keeps p and q private
4.Allice want to send plain text P=24. Note that 161and 24 are relatively prime; 24 is in Z_{161}*
She calculates C=24^2 mod 161 =93 mod 161, and sends the ciphertext 93 to Bob
5.Bob receives 93 and calculates four values:

a. a_1=+($93^{(23+1)/4}$ mod 23=1 mod 23
b. a2=-($93^{(23+1)/4}$ mod 23=22 mod 23
c. b1=+($93^{(7+1)/4}$ mod 7=4 mod 7
d. b2=-($93^{(7+1)/4}$ mod 7=3 mod 7

6.Bob takes four possible answers, (a1,b1), (a1,b2), (a2,b1),(a2,b2) and uses Chinese Remainder Theorem to find 4 possible plain texts: 116,24,137 and 45.

Case 1:
By using (a1=1,b1=4) combinations with modulo (p=23,q=7), Let X is plain text:
X = 1 mod 23

X= 4 mod 7
By using Chinese Remainder Theorem:
M=23x7=161, M_1=M/23=161/23=7, M_2=M/7=161/7=23
$M_1^{-1}=7^{-1}$ mod 23 = 7^{23-2} mod 23 = 7^{21} mod 23=10
$M_2^{-1}=23^{-1}$ mod 7 = 23^{7-2} mod 7 = 23^{5} mod 7=4
X= (a_1 x M_1xM_1^{-1}+a_2xM_2xM_2^{-1}) mod M
=(1 x 7 x 10 + 4 x 23 x 4) mod 161 = 438 mod 161=116

Case 2:
By using (a1=1,b2=3) combinations with modulo (p=23,q=7), Let X is plain text:
X = 1 mod 23
X= 3 mod 7
By using Chinese Remainder Theorem:
M=23x7=161, M_1=M/23=161/23=7, M_2=M/7=161/7=23
$M_1^{-1}=7^{-1}$ mod 23 = 7^{23-2} mod 23 = 7^{21} mod 23=10
$M_2^{-1}=23^{-1}$ mod 7 = 23^{7-2} mod 7 = 23^{5} mod 7=4
X= (a_1 x M_1xM_1^{-1}+a_2xM_2xM_2^{-1}) mod M
=(1 x 7 x 10 + 3 x 23 x 4) mod 161 = 346 mod 161=24

Case 3:
By using (a2=22,b1=4) combinations with modulo (p=23,q=7), Let X is plain text:
X = 22 mod 23
X= 4 mod 7
By using Chinese Remainder Theorem:
M=23x7=161, M_1=M/23=161/23=7, M_2=M/7=161/7=23
$M_1^{-1}=7^{-1}$ mod 23 = 7^{23-2} mod 23 = 7^{21} mod 23=10
$M_2^{-1}=23^{-1}$ mod 7 = 23^{7-2} mod 7 = 23^{5} mod 7=4
X= (a_1 x M_1xM_1^{-1}+a_2xM_2xM_2^{-1}) mod M
=(22 x 7 x 10 + 4 x 23 x 4) mod 161 = (1540+368) mod 161=137

Case 4:
By using (a2=22,b2=3) combinations with modulo (p=23,q=7), Let X is plain text:
X = 22 mod 23
X= 4 mod 7
By using Chinese Remainder Theorem:
M=23x7=161, M_1=M/23=161/23=7, M_2=M/7=161/7=23
$M_1^{-1}=7^{-1}$ mod 23 = 7^{23-2} mod 23 = 7^{21} mod 23=10
$M_2^{-1}=23^{-1}$ mod 7 = 23^{7-2} mod 7 = 23^{5} mod 7=4
X= (a_1 x M_1xM_1^{-1}+a_2xM_2xM_2^{-1}) mod M
=(22 x 7 x 10 + 3 x 23 x 4) mod 161 = (1540+276) mod 161=45

So, Finally from four cases: we got four plain text messages
Case 1: 116
Case 2: 24
Case 3: 137
Case 4: 45.
Only second answer(24) is Alice plain text, Bob needs to make a decision based on the situation

Secure of the Rabin System:
The Rabin System is secure as long as p and q are large numbers

UNIT-IV

Data Integrity, Digital Signature Schemes & Key Management

Message Integrity and Message Authentication, Cryptographic Hash Functions, Digital Signature, Key Management.

Message Integrity and Message Authentication

1. Message Integrity:

The cryptography systems that we have studied so far provide secrecy, or confidentiality, but not integrity.

However, there are occasions where we may not even need secrecy but instead must have integrity(Data will not changed).

Document and Fingerprint:

One way to preserve the integrity of a document is through the use of a fingerprint.

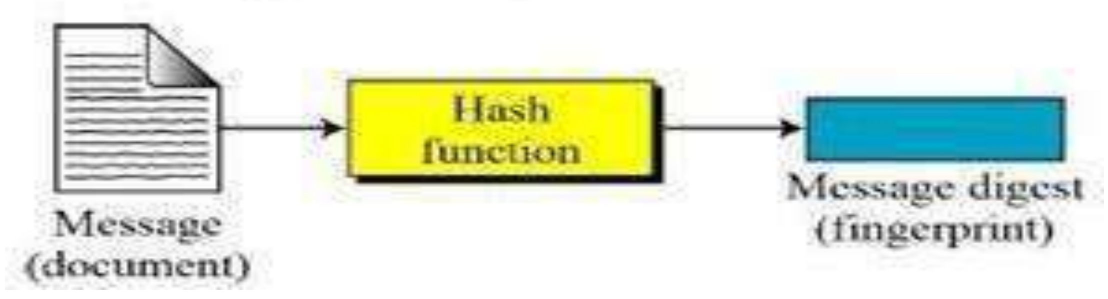
If Alice needs to be sure that the contents of her document will not be changed, she can put her fingerprint at the bottom of the document.

Message and Message Digest:

The electronic equivalent of the document and fingerprint pair is the message and digests pair.

To preserve the integrity of a message, the message is passed through an algorithm called a cryptographic hash function.

Figure *Message and digest*



Difference:

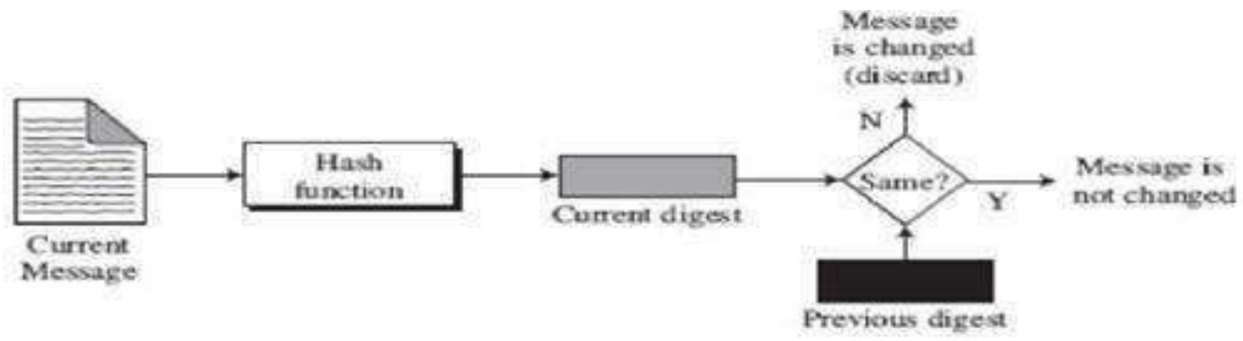
The two pairs (document / fingerprint) and (message / message digest) are similar, with some differences.

The document and fingerprint are physically linked together. The messa ge and message digest can be unlinked separately, and, most importantly, the message digest needs to be safe from change.

Note: The message digests needs to be safe from change.

Checking Integrity:

To check the integrity of a message, or document, we run the cryptographic hash function again and compare the new message digest with the previous one. If both are the same, we are sure that the original message has not been changed. Figure shows the idea.

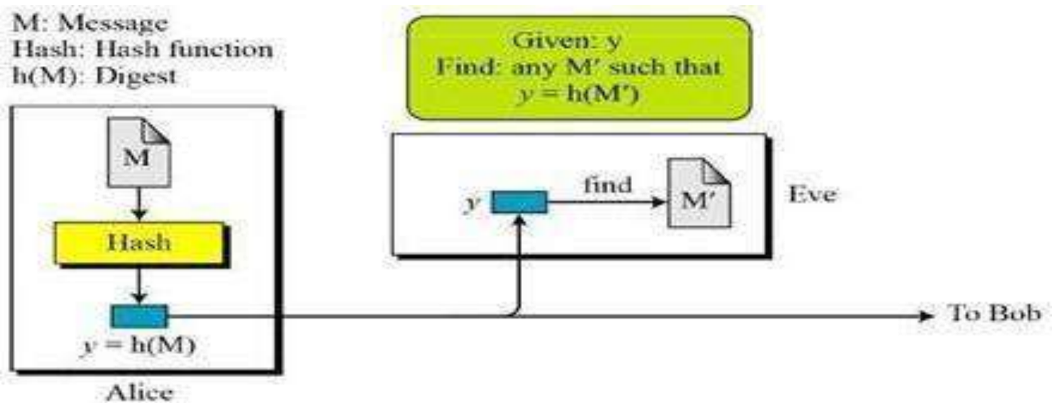


Cryptographic Hash Function Criteria:

A cryptographic hash function must satisfy three criteria

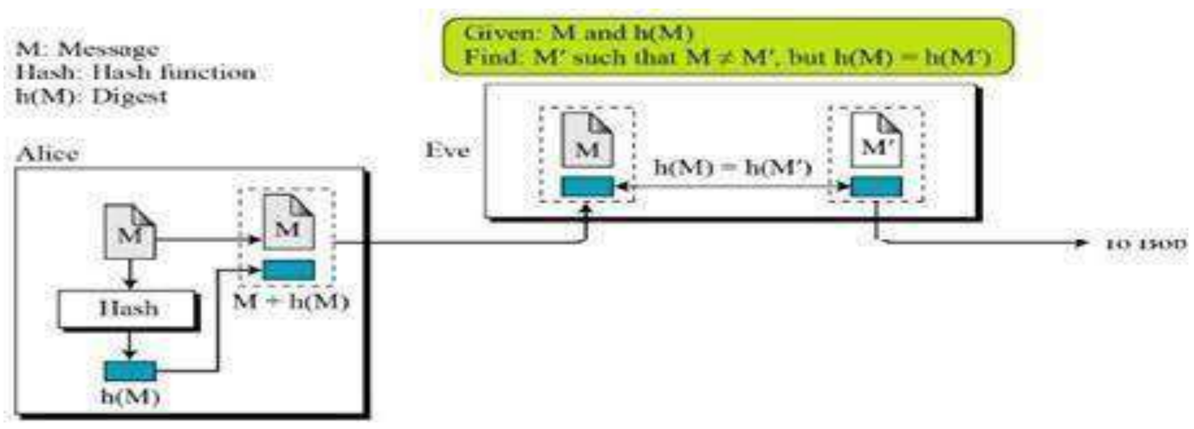
1. Pre-image Resistance
2. Second Pre-image Resistance
3. Collision Resistance.

Preimage Resistance: The hash function must be a one-way function: For any given code h, it is computationally infeasible to find h^{-1}.



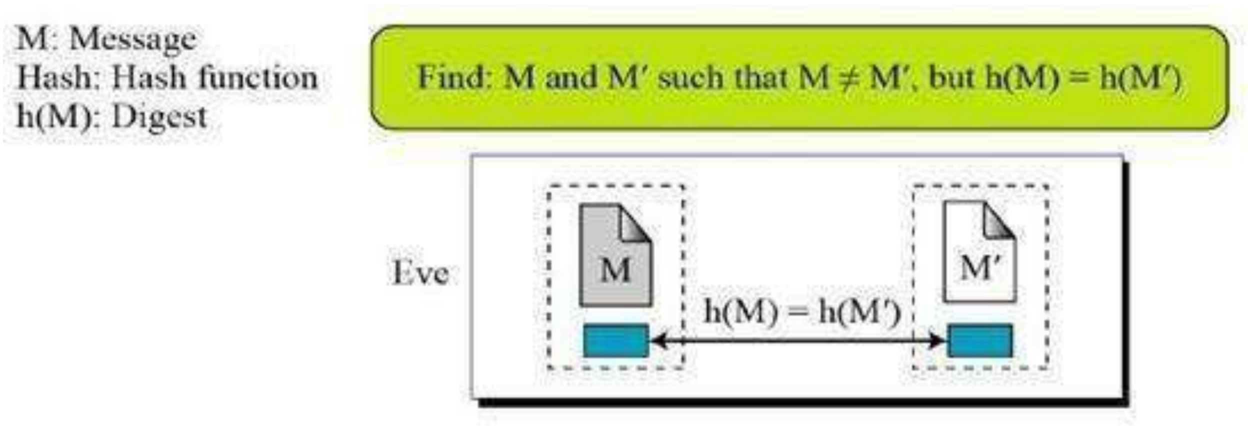
Second Preimage Resistance: In this criterion, an adversary is provided with the value of x and is asked to compute the value of x1 ≠ x, such that h(x) = h(x1).

If it difficult for the attacker to perform this computation we claim that the hash function is second pre-image resistant.

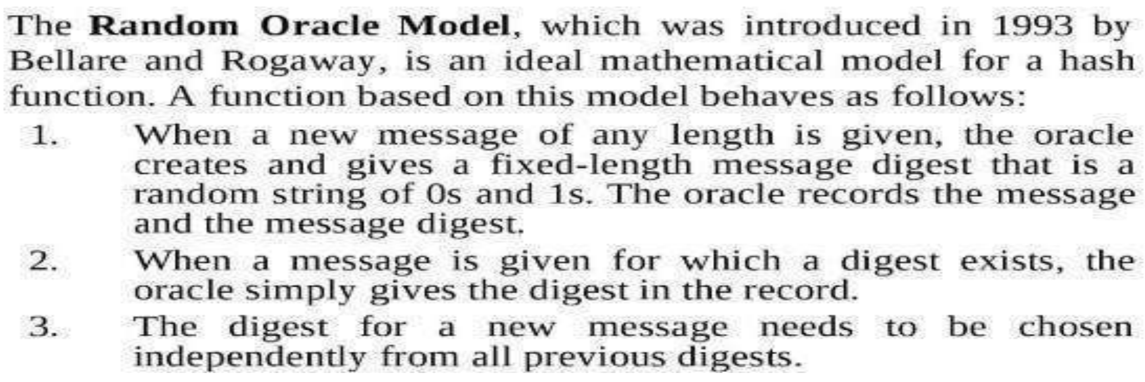


A function without preimage resistance is usually also not second preimage resistant: Given a message x_1, calculate $h := H(x_1)$ and then get a preimage x_2 from h, then we usually have $x_1 \neq x_2$ and $H(x_1) = H(x_2)$.

Collision Resistance: Collision of a hash function is the event when two values x and x1, such that x1 ≠ x hash to the same value, i.e., h(x) = h(x^1).



Random Oracle Model:



The **Random Oracle Model**, which was introduced in 1993 by Bellare and Rogaway, is an ideal mathematical model for a hash function. A function based on this model behaves as follows:

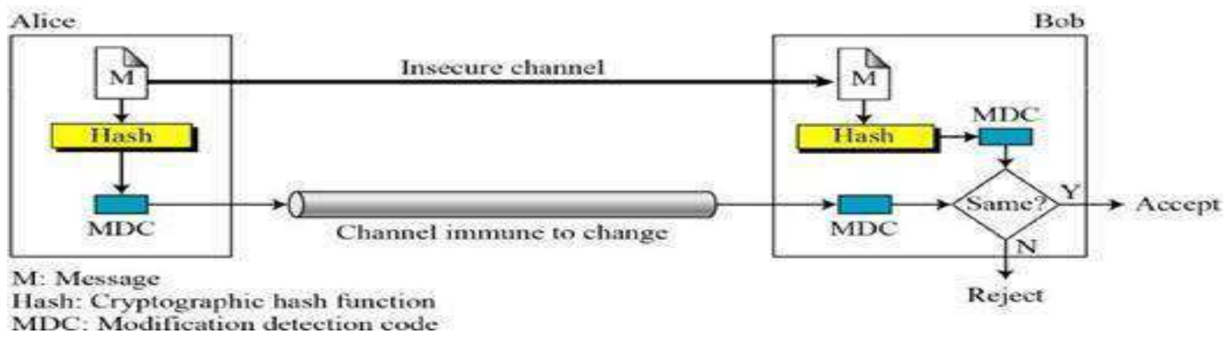
1. When a new message of any length is given, the oracle creates and gives a fixed-length message digest that is a random string of 0s and 1s. The oracle records the message and the message digest.
2. When a message is given for which a digest exists, the oracle simply gives the digest in the record.
3. The digest for a new message needs to be chosen independently from all previous digests.

2. Message Authentication:

- A message digest guarantees the integrity of a message. It guarantees that the message has not been changed.
- A message digest does not authenticate the sender of the message.
- When Alice sends a message to Bob, Bob needs to know if the message is coming from Alice.
- To provide message authentication, Alice needs to provide proof that it is Alice sending the message and not a fraud.
- The digest created by a cryptographic hash function is normally called a **Modification Detection Code (MDC**). This code can detect any modifications in the message.
- What we need for message authentication is a **Message Authentication Code** (MAC).

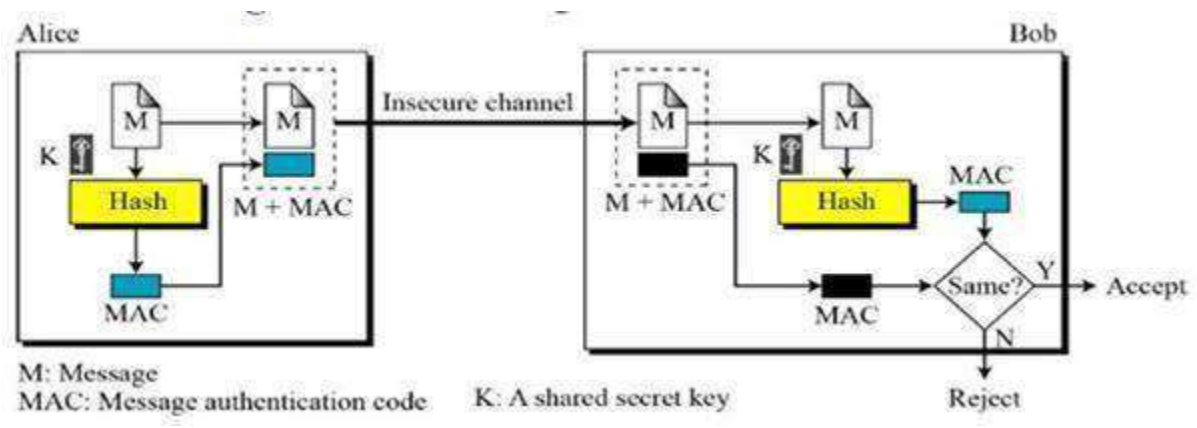
Modification Detection Code (MDC):

- A modification detection code (MDC) is a message digest that can prove the integrity of the message: that message has not been changed.
- If Alice needs to send a message to Bob and be sure that the message will not change during transmission,
- Alice can create a message digest, MDC, and send both the message and the MDC to Bob. Bob can create a new MDC from the message and compare the received MDC and thenew MDC. If they are the same, the message has not been changed.



Message Authentication Code (MAC):

- To ensure the integrity of a message and the data origin authentication – we need to change a modification detection code (MDC) to a Message Authentication Code (MAC).
- The difference between MDC and MAC is that the second include a secrete key between Alice and Bob.



Alice uses a hash function to create a MAC from the concatenation of the key and the message, h(K | M). She sends the message and the MAC to Bob over the insecure channel. Bob separates the message from the MAC. He then makes a new MAC from the concatenation of the message and the secret key. Bob then compares the newly created MAC with the one received. If the two MACs match, the message is authentic and has not been modified by an adversary.

MAC Security

How can Eve forge a message without having the key?

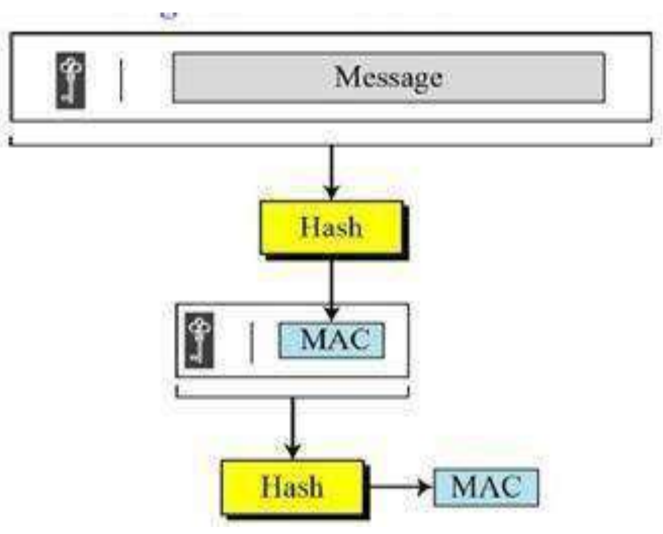
1. If size of the key allows exhaustive search, Eve may try all possible keys to digest the message.
2. Use preimageattack.
3. Given some pairs of messages and their MACs, Eve can manipulate them to come up with a new message and its digest

Note: The security of a MAC depends on the security of the underlying hash algorithm.

Nested MAC:

✓ To improve MAC security, nested MACs were designed in which hashing is performed twice.

- In 1st step, the key is concatenated with the message and is hashed to create an intermediate digest.
- In 2nd step, the key is concatenated with the intermediate digest to create the final digest.



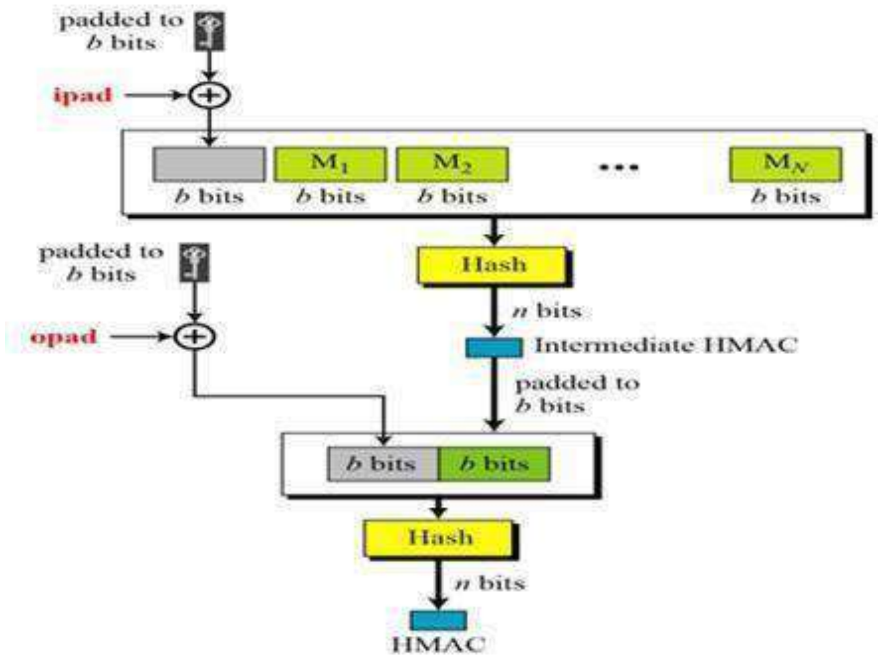
HMAC (Hashed MAC):

- HMAC algorithm stands for Hashed or Hash based Message Authentication Code
- it uses the Hashing concept twice, so great resistant to attacker
- HMAC consists of twin benefits of Hashing and MAC

✓ **The working of HMAC starts** with taking a message M containing blocks of length b bits.

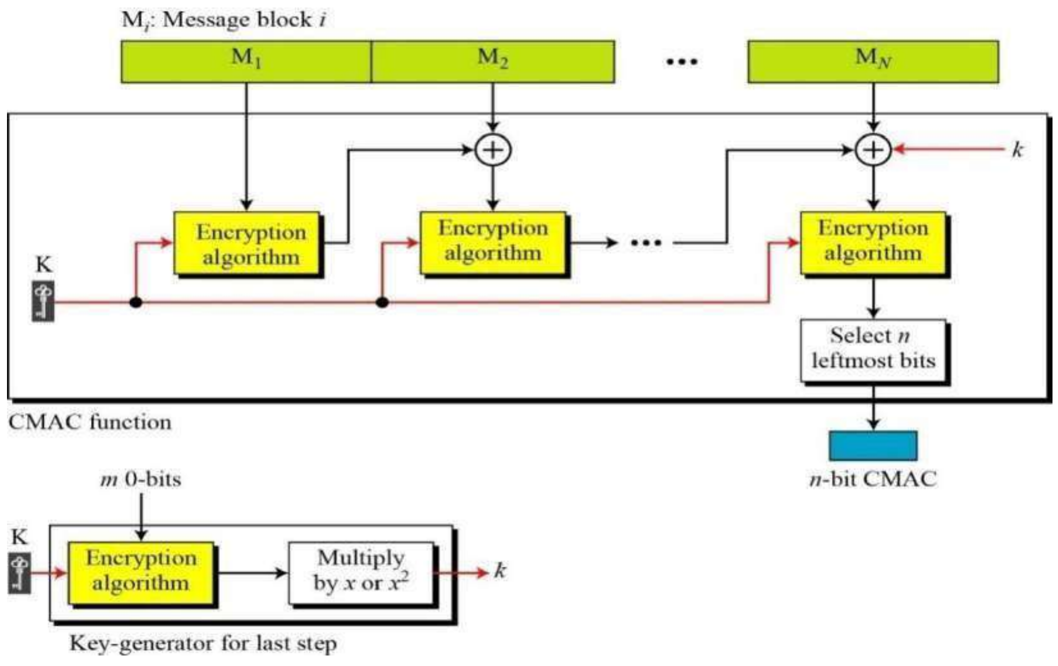
✓ An input signature is padded to the left of the message and the whole is given as input to a hash function which gives us a intermediate HMAC.

✓ Intermediate HMAC again is appended to an output signature and the whole is applied a hash function again, the result is our final HMAC of n bits



CMAC (Cipher based MAC)

- This is similar to CBC(Cipher Block Chaining),
- It takes N blocks of message but creates one block of MAC
- The message is divided into N blocks of m-bit size. If last block is not m-bit size,then
  padded with start 1 then 0000…, like 100000…
- The block is encrypted with key K then its output is XOR with the next block for 2nd
  encryption, so on.
- The last block is encrypted with some addtional k value for more scurity.



Cryptographic Hash Functions

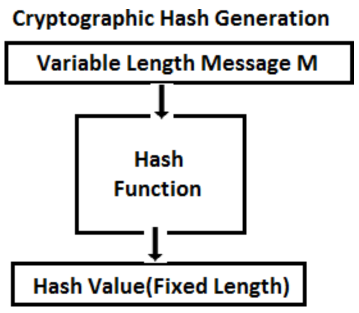
A ***cryptographic hash function*** takes a message of arbitrary length and creates a message digest of fixed length, also called hash.

A cryptographic hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value.

There are two most promising cryptographic hash algorithms –

- SHA-512

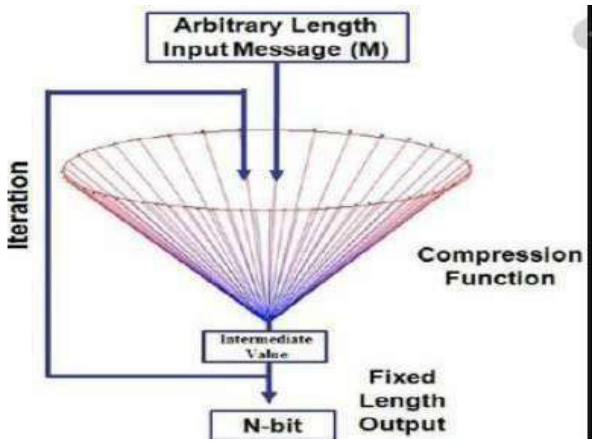
- Whirlpool



Iterated Hash Function

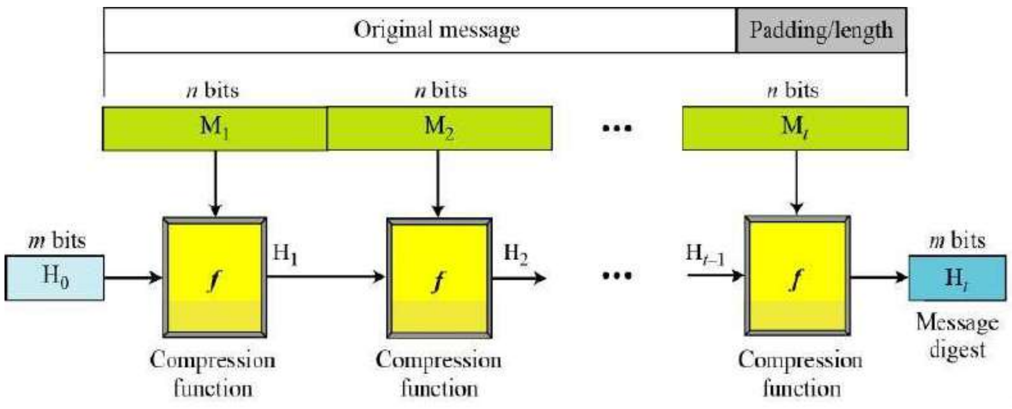
All cryptographic functions need to create a fixed size digest out of a variable-size message. Actually, the hash function is fixed size input function, but performs number of times.

This fixed-size hash function is referred to as a compression function, it compresses m-bit string input to n bit string.



Merkle-Damgard Scheme

- This is an iterated hash function that is collision resistant
- This is the basis for many cryptographic hash functions today.



- Message is divided into t-blocks of n-bit size. If necessary some bits are padded
- The blocks are $M_1, M_2, \ldots M_t$ and the digest created at each compression function are $H_1, H_2, \ldots H_t$
- Before starting the iteration, the digest H_0 is set to fixed Value called IV(initial value or initial vector)

The compression function operates on H_{i-1} and M_i to create a new H_i. $H_i=f(H_i-1,M_i)$ where f is a compression function

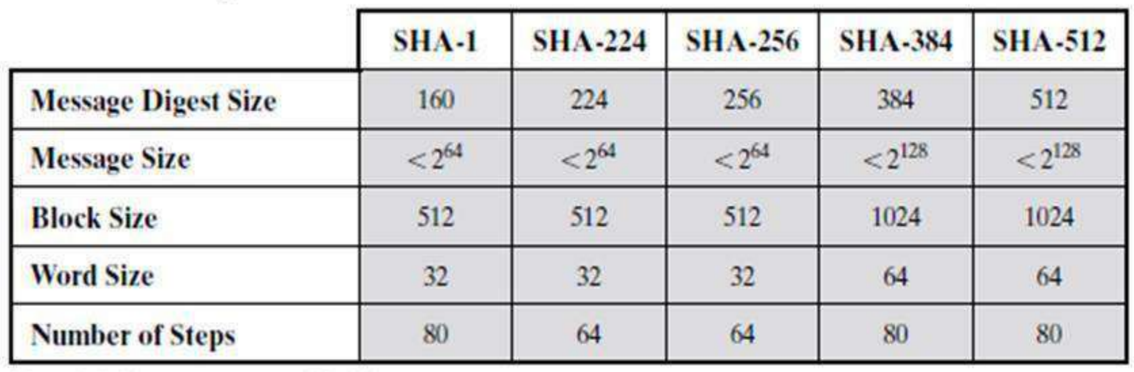
Hash Functions Invention

- Several Hash functions were designed by Ron Rivest.
- These are MD(Message Digest), MD2, MD4, and MD5
- MD5 takes blocks of size 512-bits and creates 128-bit digest.
- The 128-bit size digest is too small to resist collision attack.

Secure Hash Algorithm(SHA)

- SHA originally designed by NIST & NSA in 1993
- SHA was revised in 1995 as SHA-1
- adds 3 additional versions of SHA
- SHA-256, SHA-384, SHA-512structure & detail is similar to SHA-1

Table Comparison of SHA Parameters



	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

SHA – 512

- SHA-512 is family of Secure Hash Algorithm
- SHA-512 creates a 512 bit message digest .
- The original message divided into multiple blocks of size 1024bits.
- The Processing of each block involves 80 rounds
- Each block of size(1024bits) can be assumed as 16 words of size 64bits
- The maximum size of message is less than 2^{128}. This means that if the length of a message equal to or greater than 2^{128}, it will not be processed by SHA-512

- SHA-512 based on Merkle-Damgard scheme.

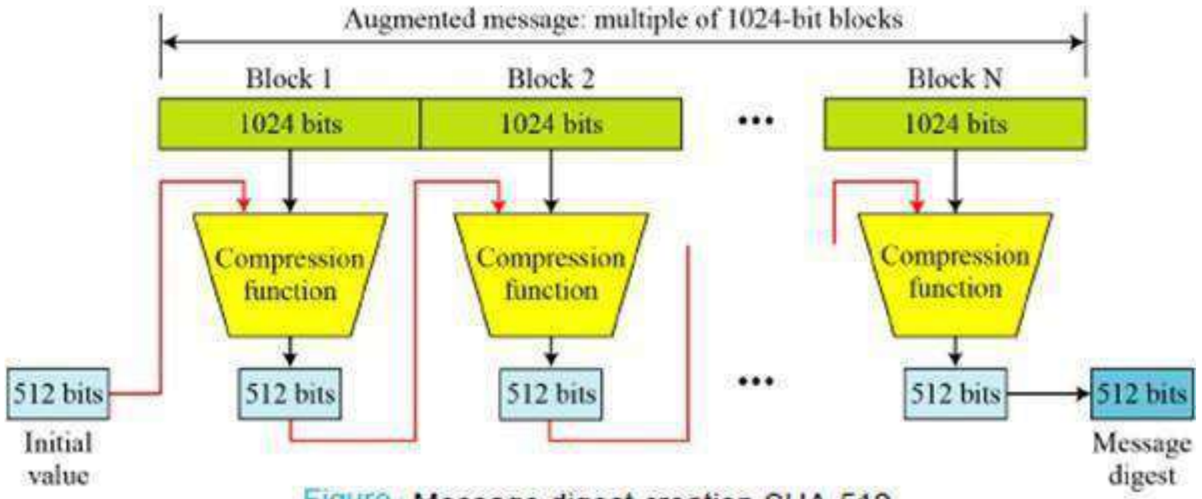
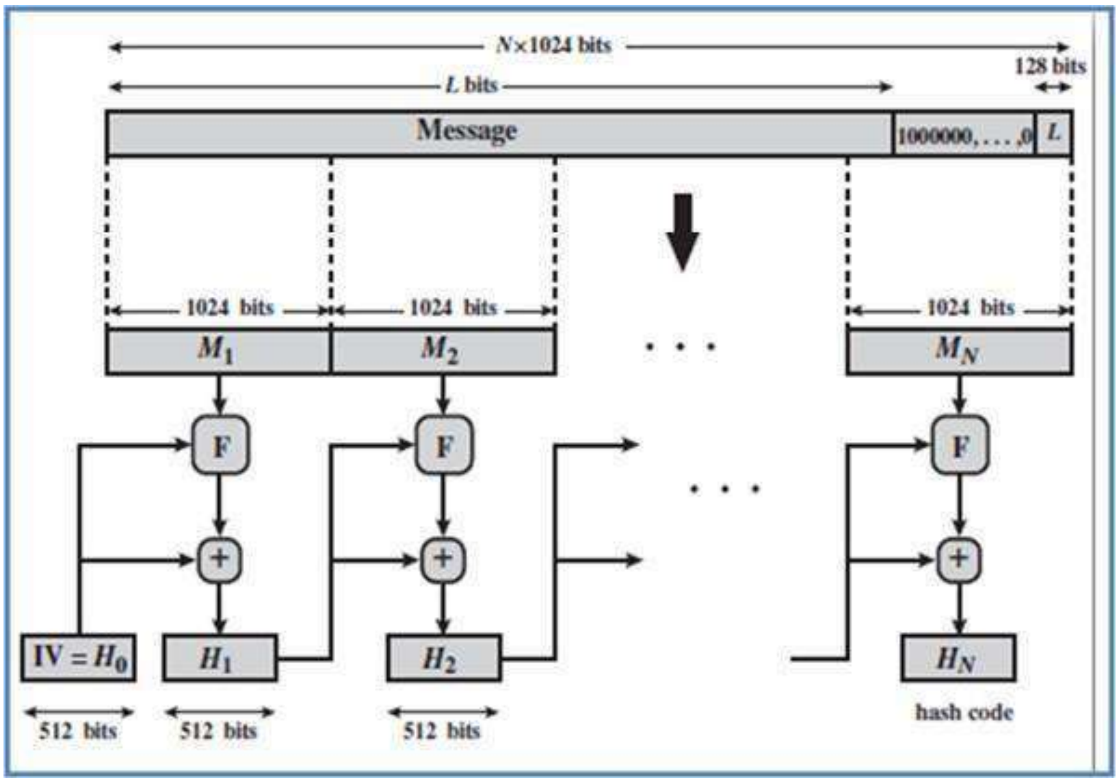


Figure Message digest creation SHA-512

The Following Figure shows internal logic of the SHA-512



STEPS:

1. **Append padding bits:**

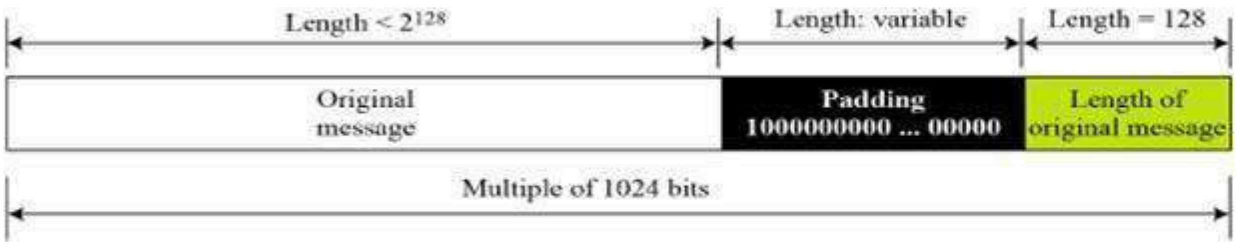
The message is padded with 1000000…. To make the message multiples of 1024.

2. Append length of the message:

A block of 128 bits is appended to the message. Contains the length of the original message.
Before addition of the length of message , we need to pad as specified in the first step.
The size of padding bits is calculated
as: $(|M|+|P|+128)=0 \bmod 1024$
$|P|=-|M|-128 \bmod 1024$
Example: What is the number of padding bits if the length of the original message is 2590
Solution: $|P|=-2590-128 \bmod 1024$
$=-2718 \bmod 1024 = -670 \bmod 1024$
$=(1024-670) \bmod 1024 = 354$
The padding consists of one 1 followed by 353 0's

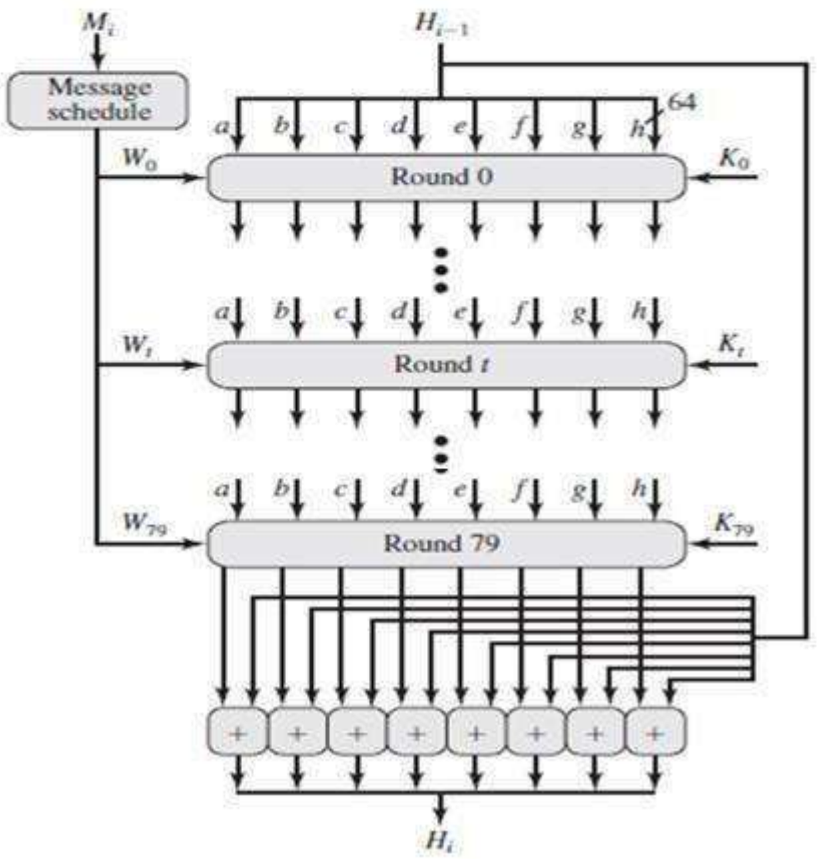
Length Field and Padding:

Before the message digest can be created, SHA-512 requires the addition of a 128-bit length field (0-(2^{128}-1)to the message that defines the length of the message in bits.



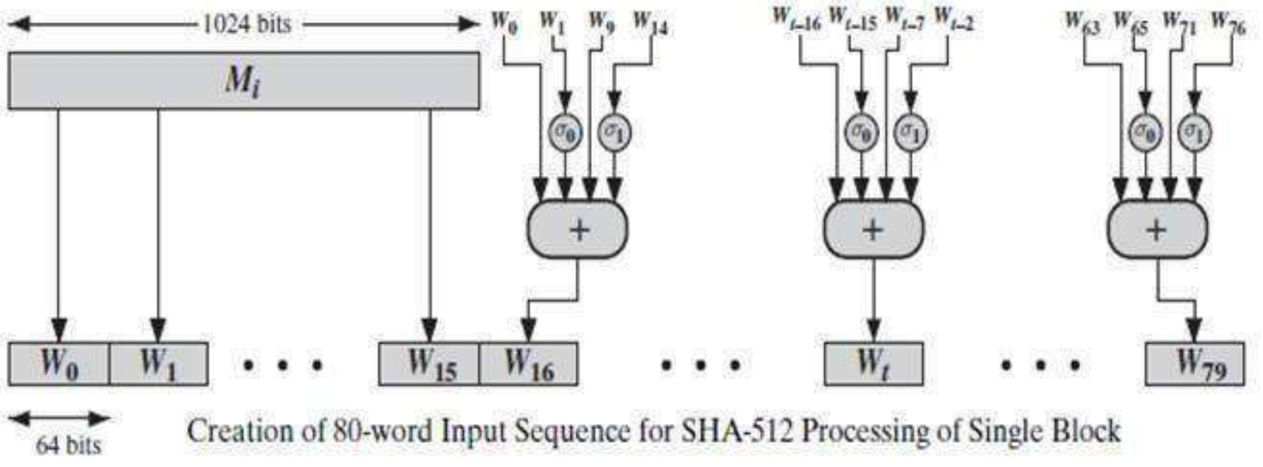
Compression Function

The heart of the algorithm is a module that consists of 80 rounds; this module is labeled as F in Block Diagram.
Each round t takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer.
Each round t makes use of a 64-bit value Wt, derived from the current 1024-bit block being processed (Mi).
Each round t also makes use of an additive constant Kt (64-bit)
The output of the 80th round is added to the input to the first round (Hi-1) to produce Hi.



Firgure:Processing of a single 1024-bit block

80-Word Input Sequence


1024 bits
M_i
W_0 W_1 W_9 W_{14}
W_{t-16} W_{t-15} W_{t-7} W_{t-2}
W_{63} W_{65} W_{71} W_{76}
σ_0 σ_1
+
W_0 W_1 W_{15} W_{16} W_t W_{79}
64 bits

Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\begin{aligned}
\sigma_0^{512}(x) &= \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus \mathrm{SHR}^7(x)\\
\sigma_1^{512}(x) &= \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^6(x)\\
\mathrm{ROTR}^n(x) &= \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits}\\
\mathrm{SHR}^n(x) &= \text{left shift of the 64-bit argument } x \text{ by } n \text{ bits with padding by zeros}\\
& \text{on the right}\\
+ &= \text{addition modulo } 2^{64}
\end{aligned}$$

<u>Constants</u>

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4

…..

<u>Initialize hash buffer</u>

```
a = 6A09E667F3BCC908     e = 510E527FADE682D1
b = BB67AE8584CAA73B     f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B     g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1     h = 5BE0CD19137E2179
```

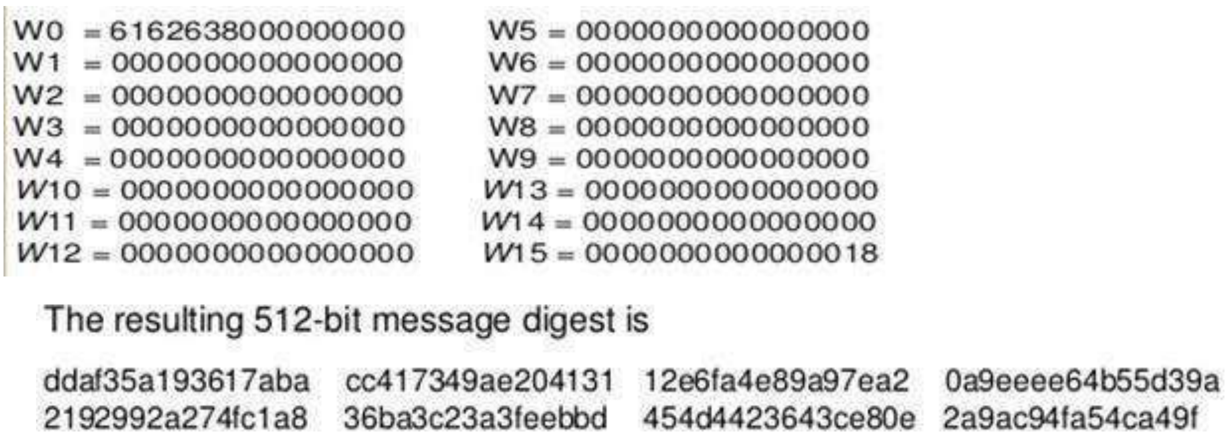
<u>Example of SHA-512</u>

ASCII characters: "abc", which is equivalent to the following 24-bit binary string:

01100001 01100010 01100011 = 616263 in Hexadecimal

The original length is 24 bits, or a hexadecimal value of 18.
the 1024-bit message block, in hexadecimal, is

```
6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018
```



```
W0  = 6162638000000000     W5  = 0000000000000000
W1  = 0000000000000000     W6  = 0000000000000000
W2  = 0000000000000000     W7  = 0000000000000000
W3  = 0000000000000000     W8  = 0000000000000000
W4  = 0000000000000000     W9  = 0000000000000000
W10 = 0000000000000000     W13 = 0000000000000000
W11 = 0000000000000000     W14 = 0000000000000000
W12 = 0000000000000000     W15 = 0000000000000018
```

The resulting 512-bit message digest is

```
ddaf35a193617aba  cc417349ae204131  12e6fa4e89a97ea2  0a9eeee64b55d39a
2192992a274fc1a8  36ba3c23a3feebbd  454d4423643ce80e  2a9ac94fa54ca49f
```

DIGITAL SIGNATURE

- A digital signature is a technique used to validate the authenticity and integrity of a message.
- In the physical world, A person signs a document to show that it originated from him or was approved by him. The signature is proof to recipient that the document comes from the correct entity.
- Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.
- Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

COMPARISON of conventional signature & DIGITAL SIGNATURE

Inclusion

A conventional signature is included in the document; it is part of the document.
But when we sign a document digitally, we send the signature as a separate document.

Verification Method

For a conventional signature, when the recipient receives a document, he compares the signature on the document with the signature on file.
For a digital signature, the recipient receives the message and the signature. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.

Relationship

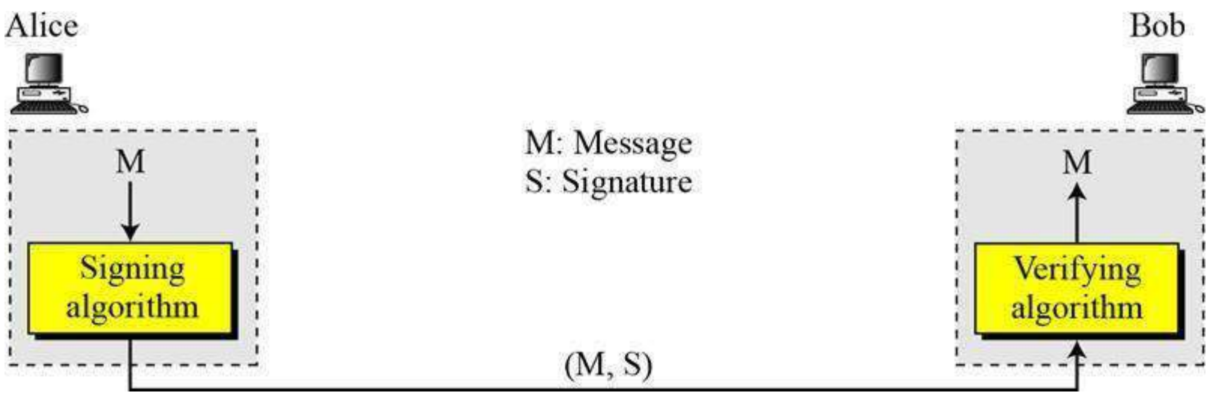
For a conventional signature, there is normally a one-to-many relationship between a signature and documents. For a digital signature, there is a one-to-one relationship between a signature and a message.

Duplicity

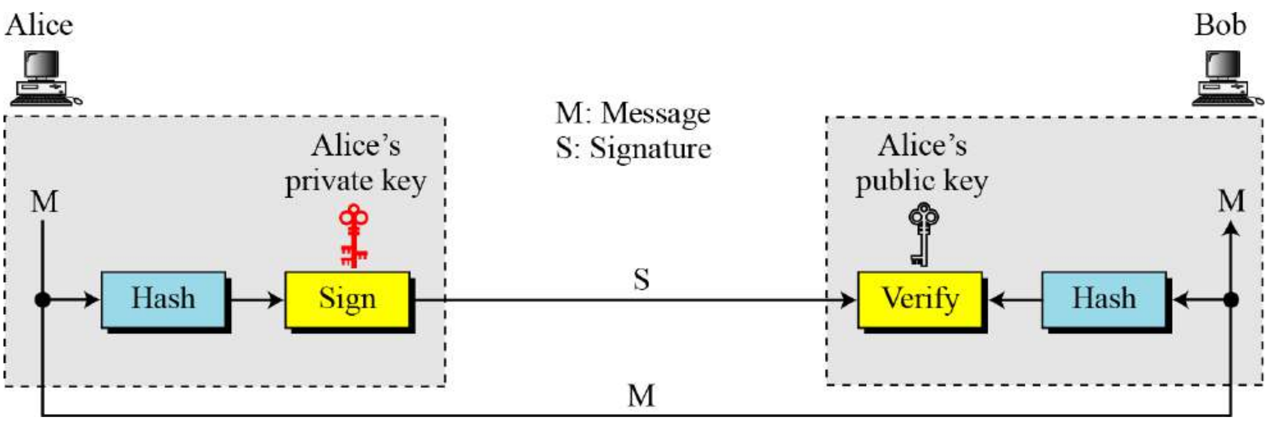
In conventional signature, a copy of the signed document can be distinguished from the original one on file. In digital signature, there is no such distinction unless there is a factor of time on the document.

PROCESS OF DIGITAL SIGNATURE

Figure shows the digital signature process. The sender uses a signing algorithm to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the verifying algorithm to the combination. If the result is true, the message is accepted; otherwise, it is rejected.



SIGNING THE DIGEST



The drawback of Asymmetric key cryptosystems that is "inefficient for long messages" .t In a digital signature system can be overcome by "signing the digest of the message".

SERVICES

The services in cryptography are:
Message confidentiality, authentication, Integrity and Non-repudiation.

- A digital signature system can provide Message authentication, Integrity and Non-repudiation, but still need encryption/decryption for message confidentiality.

Message Authentication

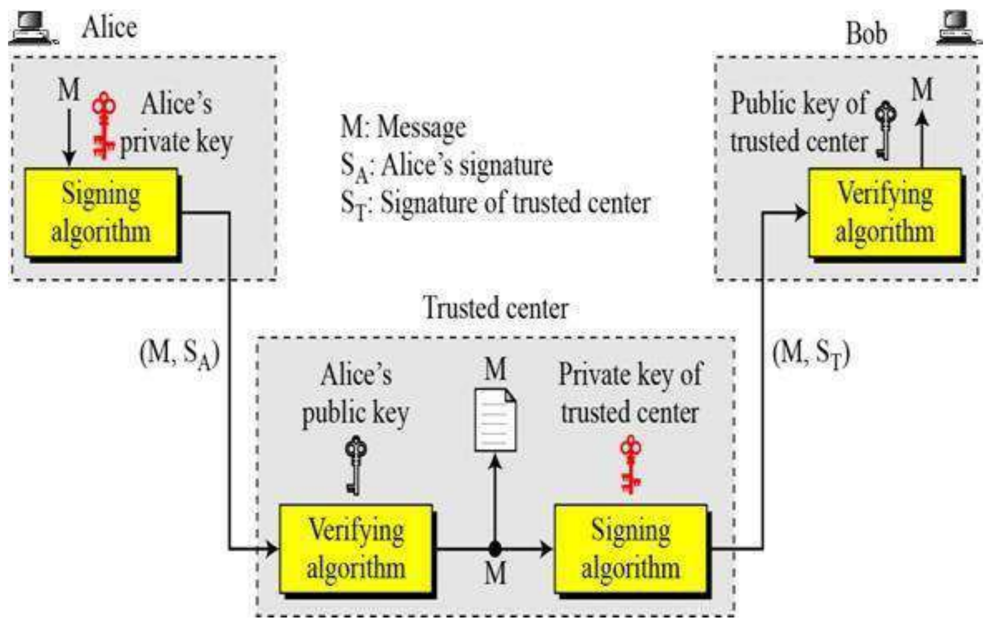
- A secure digital signature scheme, like a secure conventional signature can provide message authentication
- *Example, Bob can verify that the message is sent by Alice because Alice's public key is used in verification.*

Message Integrity

The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.

Nonrepudiation

Nonrepudiation can be provided using a trusted party.



Confidentiality

A digital signature does not provide privacy.
If there is a need for privacy, another layer of encryption/decryption must be applied.

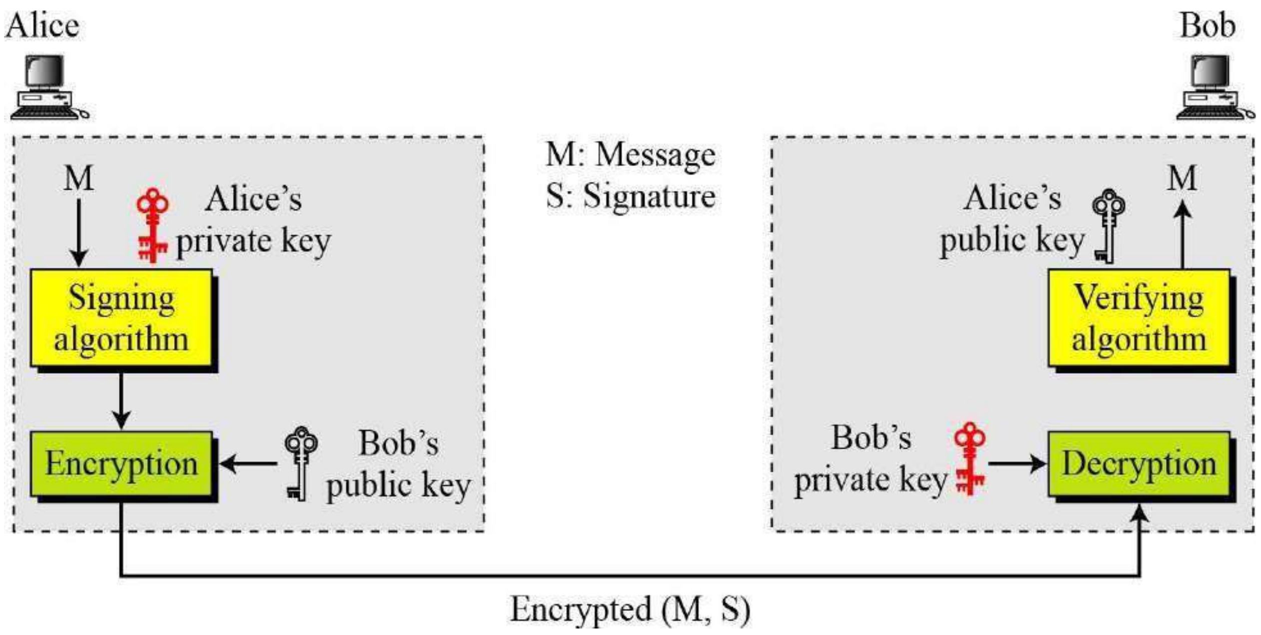


Figure *Adding confidentiality to a digital signature scheme*

ATTACKS ON DIGITAL SIGNATURE

Attack Types

1. *Key-Only Attack*

In key-only attack, the public key of A is available to every one and C makes use of this fact and try to recreate the signature of A and digitally sign the documents that A does not intend to do.

2. *Known-Message Attack*

In the known message attack, C has few previous messages and signatures of A. Now C tries to forge the signature of A on to the documents that A does not intend to sign by using the brute force method by analyzing the previous data to recreate the signature of A

3. *Chosen-Message Attack*

In this method C has the knowledge about A's public key and obtains A's signature on the messages and replaces the original message with the message C wants A to sign with having A's signature on them unchanged.

Forgery Types

1. ***Existential Forgery***

Adversary can create a pair (message, signature), such that the signature of the message is valid. Adversary has no control on the messages whose signature is forged

2. *Selective Forgery*

Adversary is able to create valid signatures on a message chosen by someone else, with a significant probability.

Adversary controls the messages whose signature is forged

DIGITAL SIGNATURE SCHEMES

Several digital signature schemes have evolved during the last few decades. Some of them have been implemented.

1. RSA Digital Signature Scheme
2. ElGamal Digital Signature Scheme
3. Schnorr Digital Signature Scheme
4. Digital Signature Standard (DSS)
5. Elliptic Curve Digital Signature Scheme

RSA DIGITAL SIGNATURE SCHEMES

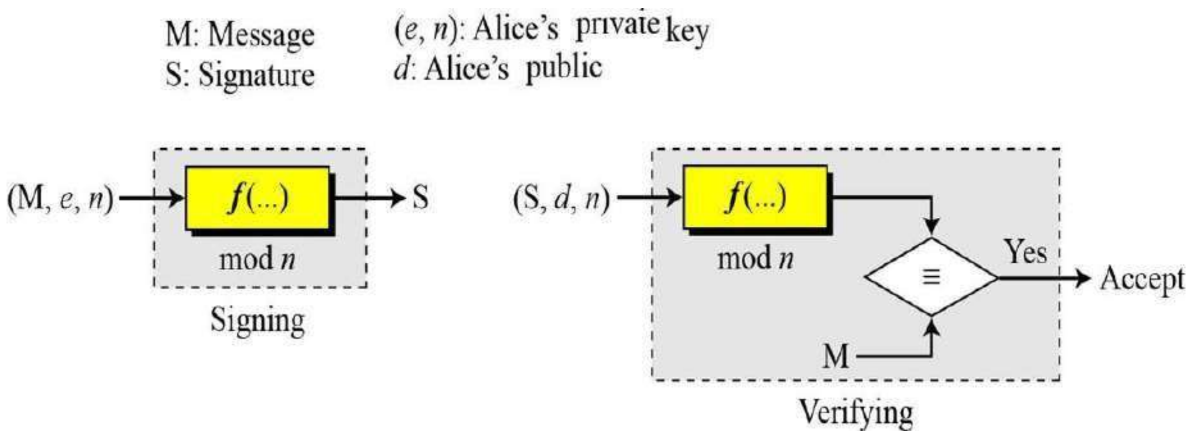


Figure : General idea behind the RSA digital signature scheme

The sender uses his own private key tosign the documemnet, the receivr uses the senders public key to verify it

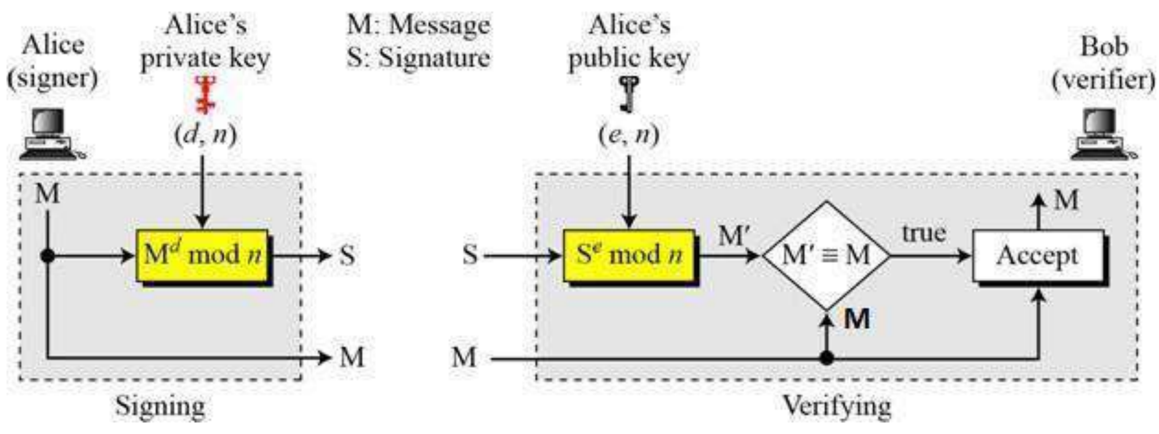
RSA DIGITAL SIGNATURE SCHEMES – Key Generation

Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA.

1. Sender chooses two prime numbers p and q
2. Calculate n=pxq
3. Calculate f(n) = (p-1) x (q-1)
4. Chooses the public exponent e and calculates d (private exponent) such that e x d = 1 mod f(n)

In the RSA digital signature scheme, *d* is private; *e* and *n* are public. RSA

DIGITAL SIGNATURE SCHEMES – Signing and verifying



Signing: Alice create a signature out of the message using her private exponent,
$S=M^d$ mod n and sends the signature to Bob

Verifying: Bob receives M and S. Bob applies A lice public exponent to the signature to create a copy of the message $M^1 = S^e$ mod n. Bob compares M and M^1 · If both are congruent, accepts the message.
$M^1 \equiv M \pmod{n} \rightarrow S^e \equiv M \pmod{n} \rightarrow M^{dxe} \equiv M \pmod{n}$

RSA DIGITAL SIGNATURE SCHEMES – EXAMPLE

As a trivial example, suppose that Alice chooses *p* = 823 and *q* = 953, and calculates *n* = 784319. The value of f(n) is 782544. Now she chooses *e* = 313 and calculates *d* = 160009. At this point key generation is complete. Now imagine that Alice wants to send a message with the value of M = 19070 to Bob. She uses her private exponent, 160009, to sign the message:

$$M: 19070 \quad \rightarrow \quad S = (19070^{160009}) \bmod 784319 = 210625 \bmod 784319$$

Alice sends the message and the signature to Bob. Bob receives the message and the signature. He calculates

$$M' = 210625^{313} \bmod 784319 = 19070 \bmod 784319 \quad \rightarrow \quad M \equiv M' \bmod n$$

Bob accepts the message because he has verified Alice's signature

<u>ElGamal Digital Signatures</u>

• signature variant of ElGamal, related to D-H
  – so uses exponentiation in a finite Galois field
  – security based difficulty of computing discrete logarithms, as in D-H
• use private key for encryption (signing)

- uses public key for decryption (verification)
- each user (eg. A) generates their key

– chooses a secret key: $1 < x_A < q-1$
– compute their **public key**: $y_A = a^{x_A} \bmod q$

- ➢ Alice signs a message M to Bob by computing
  - the hash *m = H(M)*, 0 <= m <= (q-1)
  - chose random integer K with 1 <= K <= (q-1) and gcd(K,q-1)=1
  - compute temporary key: $S_1 = a^k \bmod q$
  - compute K^{-1} the inverse of K mod (q-1)
  - compute the value: $S_2 = K^{-1}(m-x_AS_1) \bmod (q-1)$
  - signature is:(S_1,S_2)
- ➢ any user B can verify the signature by computing
  - $V_1 = a^m \bmod q$
  - $V_2 = y_A^{S1} S_1^{S2} \bmod q$
  - signature is valid if $V_1 = V_2$

ElGamal Signature Example

- ➢ use field GF(19) q=19 and a=10
- ➢ Alice computes her key:
  - A chooses x_A=16 & computes $y_A=10^{16} \bmod 19 = 4$
- ➢ Alice signs message with hash m=14 as (3,4):
  - choosing random K=5 which has gcd(18,5)=1
  - computing $S_1 = 10^5 \bmod 19 = 3$
  - finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - computing $S_2 = 11(14-16.3) \bmod 18 = 4$
- ➢ any user B can verify the signature by computing
  - $V_1 = 10^{14} \bmod 19 = 16$
  - $V_2 = 4^3.3^4 = 5184 = 16 \bmod 19$

since 16 = 16signature is valid

Schnorr Digital Signatures

- ➢ also uses exponentiation in a finite (Galois)
  - security based on discrete logarithms, as in D-H
- ➢ minimizes message dependent computation
  - multiplying a 2*n-bit* integer with an *n-bit* integer
- ➢ main work can be done in idle time
- ➢ have using a prime modulus *p*
  - *p–1* has a prime factor *q* of

appropriate size typically *p* 1024-bit and *q* 160-bit numbers

Schnorr Key Setup

- ➢ choose suitable primes *p* , *q*
- ➢ choose *a* such that $a^q = 1 \bmod p$

- (a,p,q) are global parameters for all
- each user (eg. A) generates a key
  - chooses a secret key (number): $0 < s_A < q$
  - compute their **public key**: $v_A = a^{-sA} \bmod q$

- user signs message by
  - choosing random r with $0<r<q$ and computing $x = a^r \bmod p$
  - concatenate message with x and hash result to computing: $e = H(M \| x)$
  - computing: $y = (r + se) \bmod q$
  - signature is pair (e, y)
- any other user can verify the signature as follows:
  - computing: $x' = a^y v^e \bmod p$
  - verifying that: $e = H(M \| x')$

Digital Signature Standard (DSS)

- US Govt approved signature scheme
- designed by NIST & NSA in early 90's
- published as FIPS-186 in 1991
- revised in 1993, 1996 & then 2000
- uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
- DSA is digital signature only unlike RSA is a public-key technique

Digital Signature Algorithm (DSA)

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms
- variant of ElGamal & Schnorr schemes

DSA Key Generation

- have shared global public key values (p,q,g):
  - choose 160-bit prime number q
  - choose a large prime p with $2^{L-1} < p < 2^L$
    - where L= 512 to 1024 bits and is a multiple of 64
    - such that q is a 160 bit prime divisor of (p-1)
  - choose $g = h^{(p-1)/q}$
    - where $1<h<p-1$ and $h^{(p-1)/q} \bmod p > 1$
- users choose private & compute public key:
  - choose random private key: $x<q$
  - compute public key: $y = g^x \bmod p$

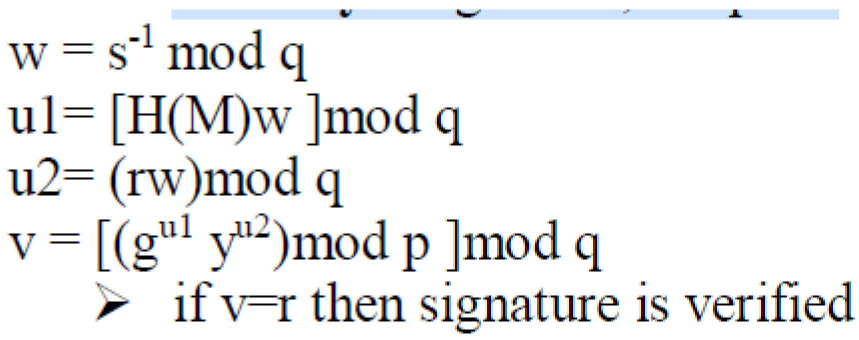
DSA Signature Creation

- to **sign** a message M the sender:
  - generates a random signature key k, $k<q$
  - nb. k must be random, be destroyed after use, and never be reused
- then computes

signature pair: $r = (g^k \bmod$
$p) \bmod q$
$s = [k^{-1}(H(M)+ xr)] \bmod q$

- sends signature (r,s) with message M
- having received M & signature (r,s)

- to **verify** a signature, recipient



$w = s^{-1} \bmod q$

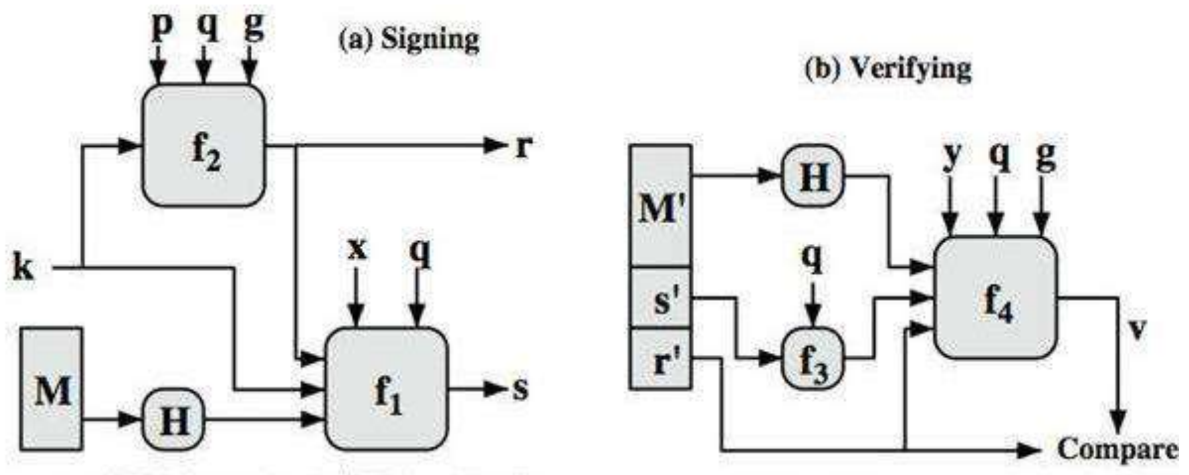
$u1 = [H(M)w] \bmod q$

$u2 = (rw) \bmod q$

$v = [(g^{u1}\, y^{u2}) \bmod p] \bmod q$

- if $v=r$ then signature is verified

DSS Overview



$s = f_1(H(M), k, x, r, q) = (k^{-1}(H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q}\, y^{r'w \bmod q}) \bmod p) \bmod q$

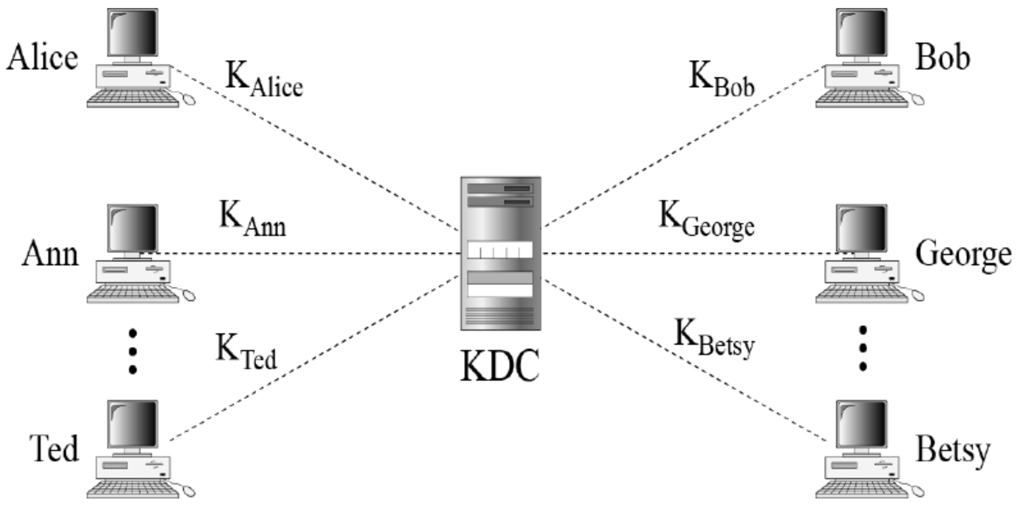
KEY MANAGEMENT

SYMMETRIC-KEY DISTRIBUTION

- Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages.
- Symmetric-key cryptography, however, needs a shared secret key between two parties.
- Example: If Alice needs to exchange confidential messages with N people, she need N different keys and if N people need to exchange with each other, they need N(N-1) keys. If 1 million people need to communicate with each other , they need more than trillions of keys.
- This proble normally referred as N^2 problem, because the number of required keys for N entitesis N^2
- We also has a problem of the distribution of keys through the internet which is unsecure.

Key-Distribution Center: KDC

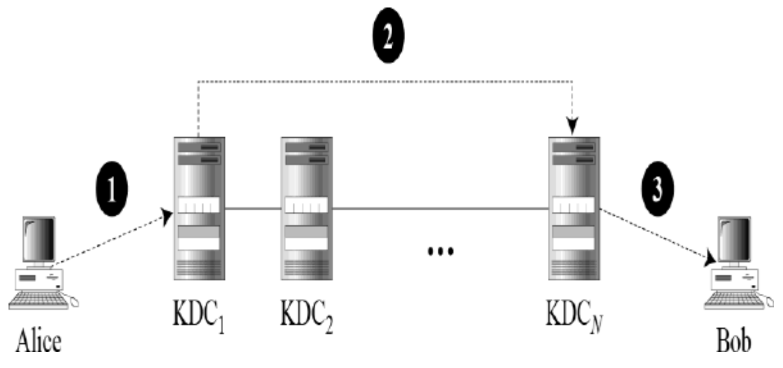
A practical solution for the above problem is the use of a trusted thord party, referred as Key-Distribution Center(KDC)



1. Alice sends a request to the KDC stating that she needs a session secrete key between her and Bob
2. KDC inform Bob about Alice request

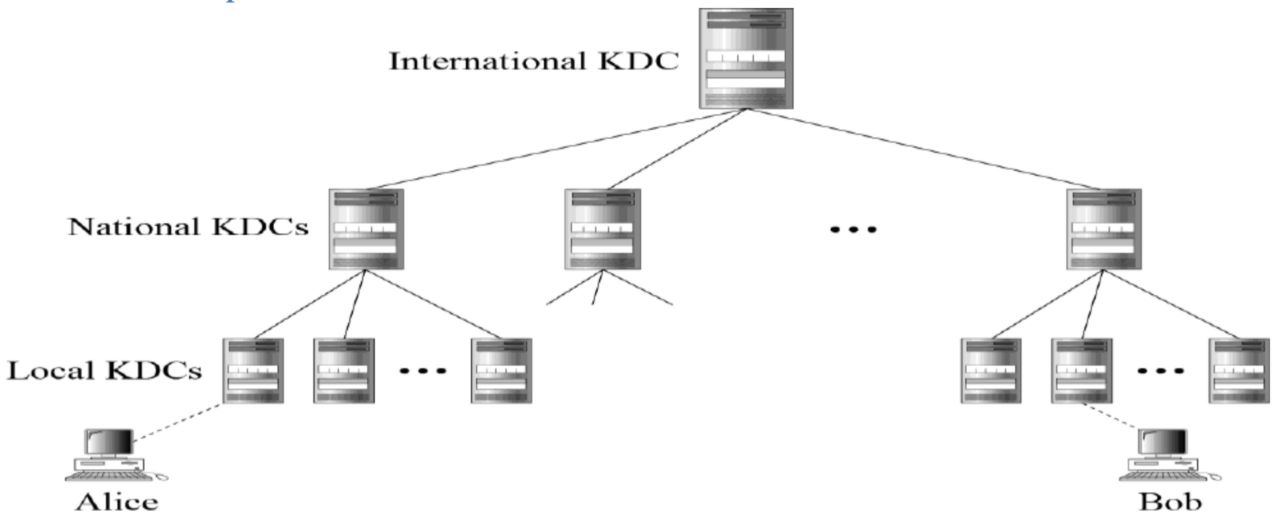
If Bob agrees, a session key is created between the two.

Flat Multiple KDCs



When the number of people using a KDC increases, the system becomes unmanageable.
To solve the problem, we use multiple KDCs. We devide the world into domains

Hierarchical Multiple KDCs



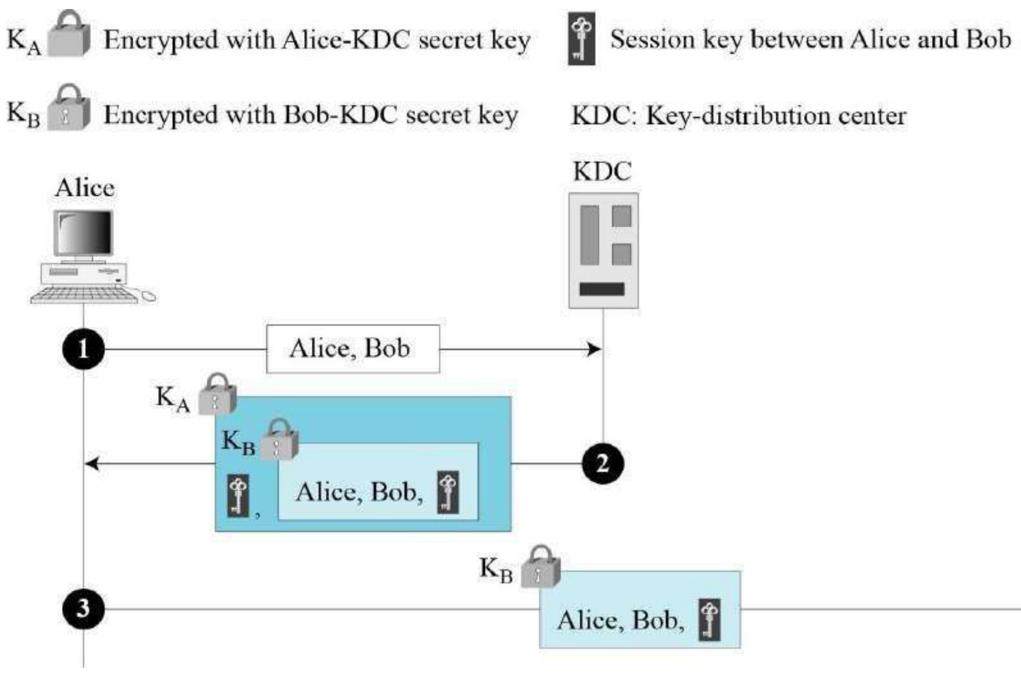
In this, KDCs are arranged in hierarchical model, the international KDC are at root, then national next and local KDCs at lower level.

Session Keys

A KDC creates a secret key for each member. This secret key can be used only between the member and the KDC, not between two members.
A session symmetric key between two parties is used only once.

Simple protocol Using a KDC

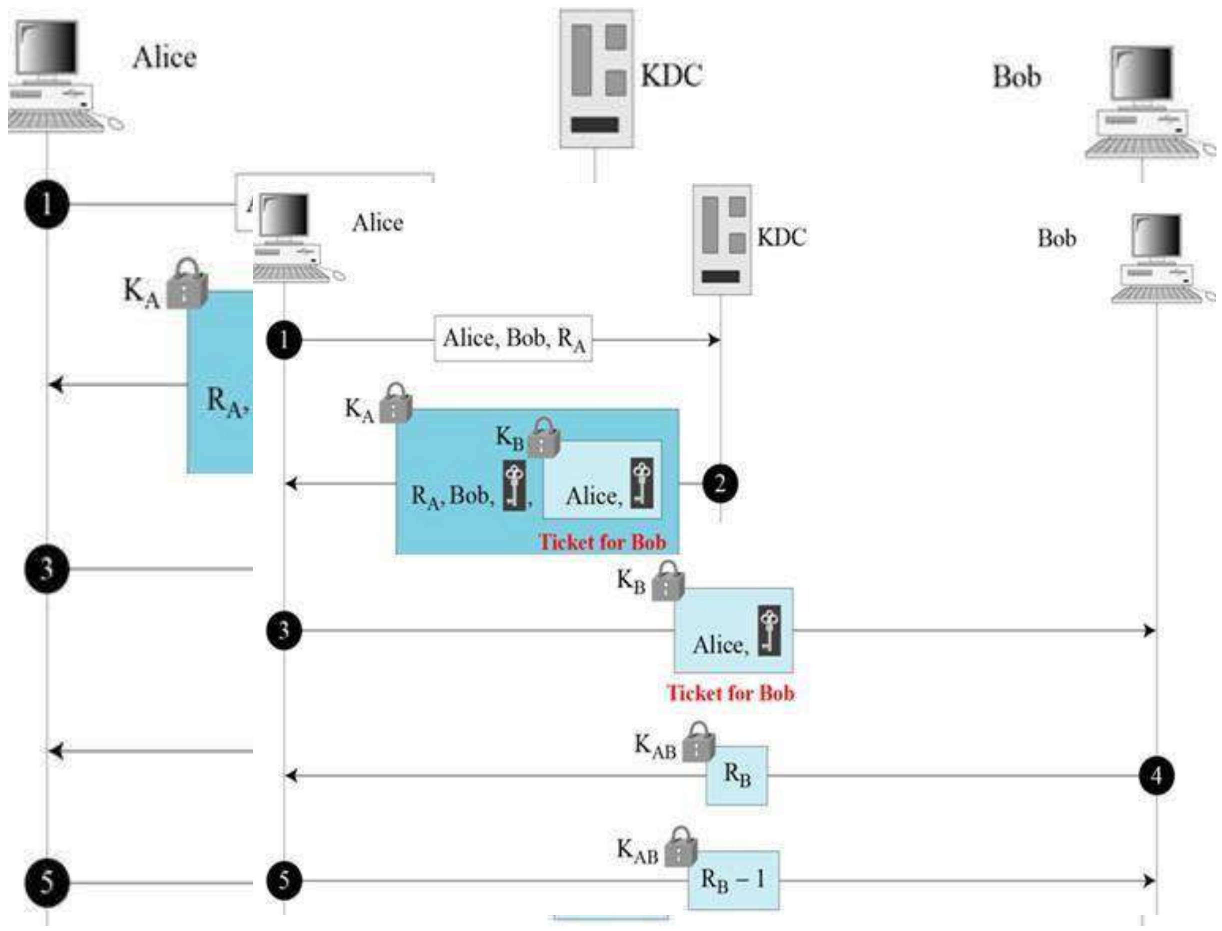
Figure shows first approach using KDC


K_A Encrypted with Alice-KDC secret key
Session key between Alice and Bob
K_B Encrypted with Bob-KDC secret key
KDC: Key-distribution center
Alice
KDC
Bob
1
Alice, Bob
K_A
K_B
Alice, Bob,
2
K_B
3
Alice, Bob,

1. Alice sends request to KDC
2. KDC creates ticket to Bob which is encrypted using Bob's key K_B. The ticket contains the session key (K_{AB}).
3. Alice extracts the Bob's ticket
4. Alice sends ticket to Bob. Bob opens the ticket and knows that Alice want to send message to him by using K_{AB}.

Drawback: Eve can use the replay attack at step 3.

Needham-Schroeder Protocol



1. Alice sends message to KDC that include her nonce, R_A
2. KDC sends encrypted ticket for Bob to Alice which contains session key
3. Alice sends Bobs ticket to him
4. Bob sends his challenge (R_B) to Alice which contains session key
5. Alice responds to Bobs challenge

KERBEROS

Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular. Several systems, including Windows 2000, use Kerberos.
Originally designed at MIT, it has gone through several versions.

KERBEROS Servers

Three servers are involved in the Kerberos protocol.

Authentication Server (AS)

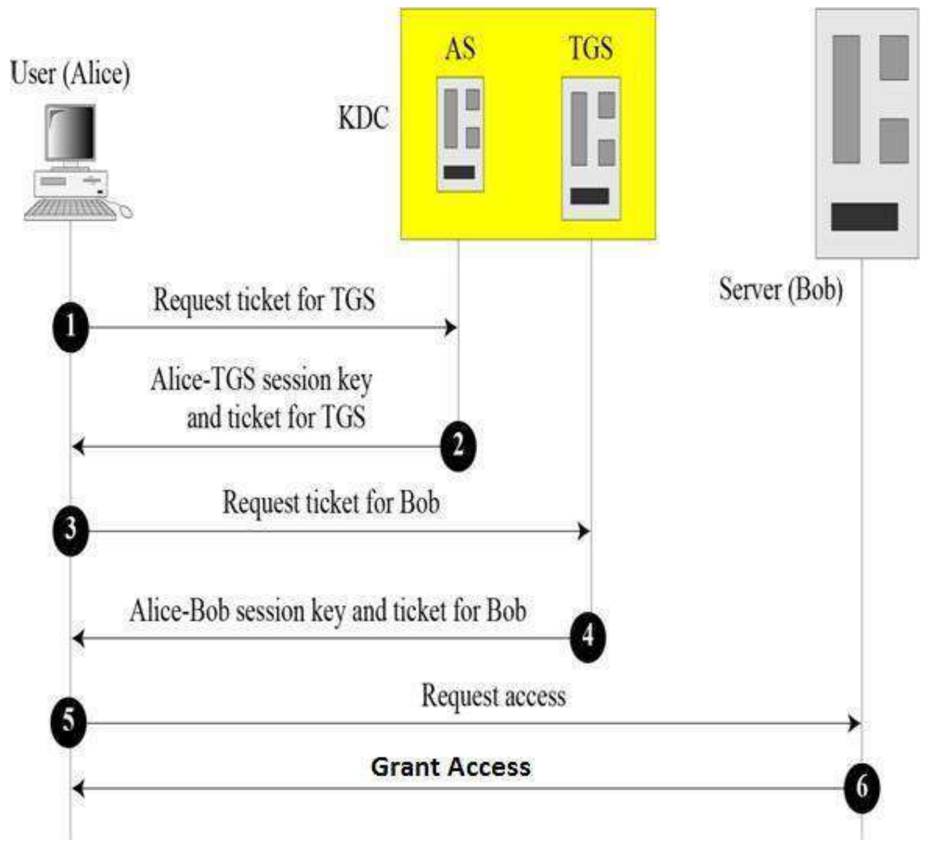
- ✓ The authentication server (AS) is the KDC in the Kerberos protocol.
- ✓ Each user registers with AS and is granted a user identity and a password.
- ✓ AS verifies the user, issues a session key to be used b/t Alice and TGS.
- ✓ and sends a ticket for TGS.

Ticket-Granting Server (TGS)

- ✓ The ticket-granting server (TGS) issues a ticket for the real server (Bob).
- ✓ Also provides the session key b/t Alice and Bob.
- ✓ Kerberos has a separated user verification from issuing of tickets.
- ✓ Alice can contact the TGS multiple times to obtained tickets for different real servers.

Real Server

- ✓ The real server (Bob) provides services for the user (Alice).
- ✓ Kerberos is designed for client-server programs.
- ✓ Kerberos is not used for person – to – person authentication

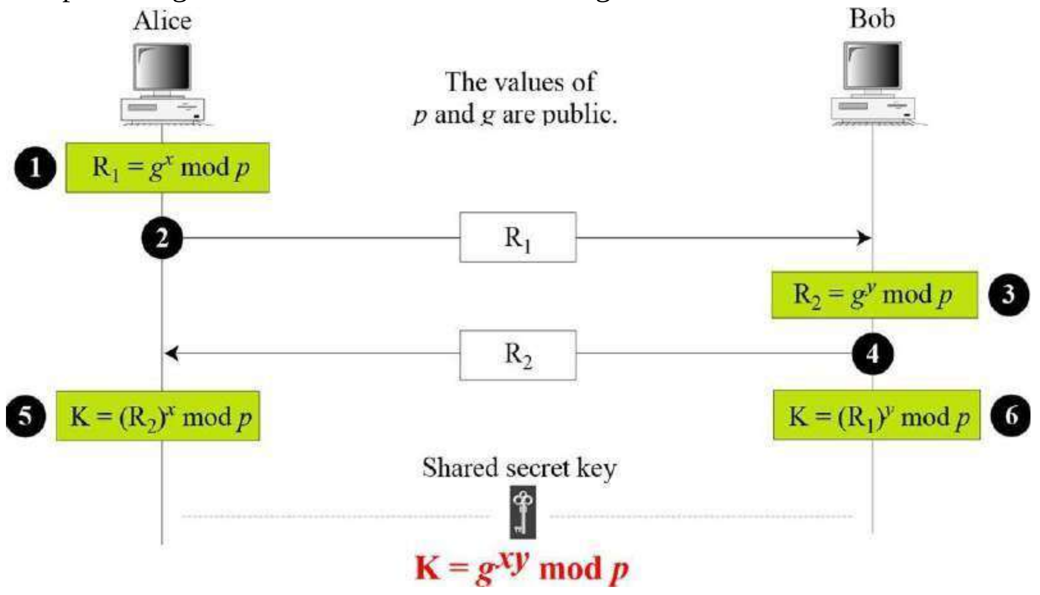


SYMMETRIC-KEY AGREEMENT

Alice and Bob can create a session key between themselves without using a KDC. This method of session-key creation is referred to as the symmetric-key agreement. Example: Diffie-Hellman Key Agreement

Diffie-Hellman Key Agreement

In this two parties are creating symmetric key without the need of a KDC.
Before establishing, the two parties need to choose two numbers **p and g**.
The p is a large number on the order of 300 digits.



Steps:

1. Alice chooses a large random integer number x and calculates R1=g^x mod p
2. Bob chooses another large number y and calculates R2=g^y mod p
3. Alice sends R1 to Bob and Bob sends R2 to Alice
4. Alice calculates key K=$(R2)^x$ mod p
5. Bob calculates key K=$(R1)^y$

mod p Where K is the symmetric key
for the session
The symmetric key in the Diffie-Hellman method is K=g^{xy} mod p

Diffie-Hellman Key Agreement- EXAMPLE

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume that $g = 7$ and $p = 23$. The steps are as follows:

1. Alice chooses $x = 3$ and calculates $R_1 = 7^3$ mod 23 = 21.
2. Bob chooses $y = 6$ and calculates $R_2 = 7^6$ mod 23 = 4.
3. Alice sends the number 21 to Bob.
4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key K = 43 mod 23 = 18.

6. Bob calculates the symmetric key K = 216 mod 23 = 18.
7. The value of K is the same for both Alice and Bob;

g^{xy} mod $p = 7^{18}$ mod 35 = 18.

PUBLIC-KEY DISTRIBUTION

In asymmetric-key cryptography, people do not need to know a symmetric shared key; everyone shields a private key and advertises a public key.
In public key key cryptography, everyone have access to every one's public key: public keys are available to the public.
So, public keys need to be distributed.

1. Public Announcement
2. Trusted Center
3. Controlled Trusted Center
4. Certification Authority

5. X.509
6. Public-Key Infrastructures (PKI)

Public Announcement

The normal method is to announce public keys publicly, but is not secure

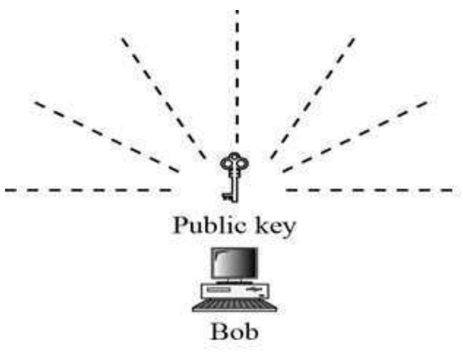
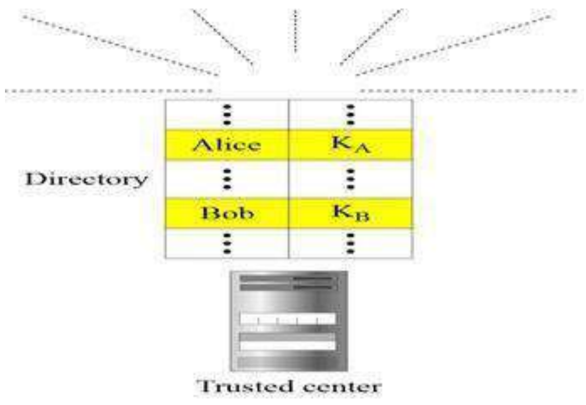


Figure *Announcing a public key*

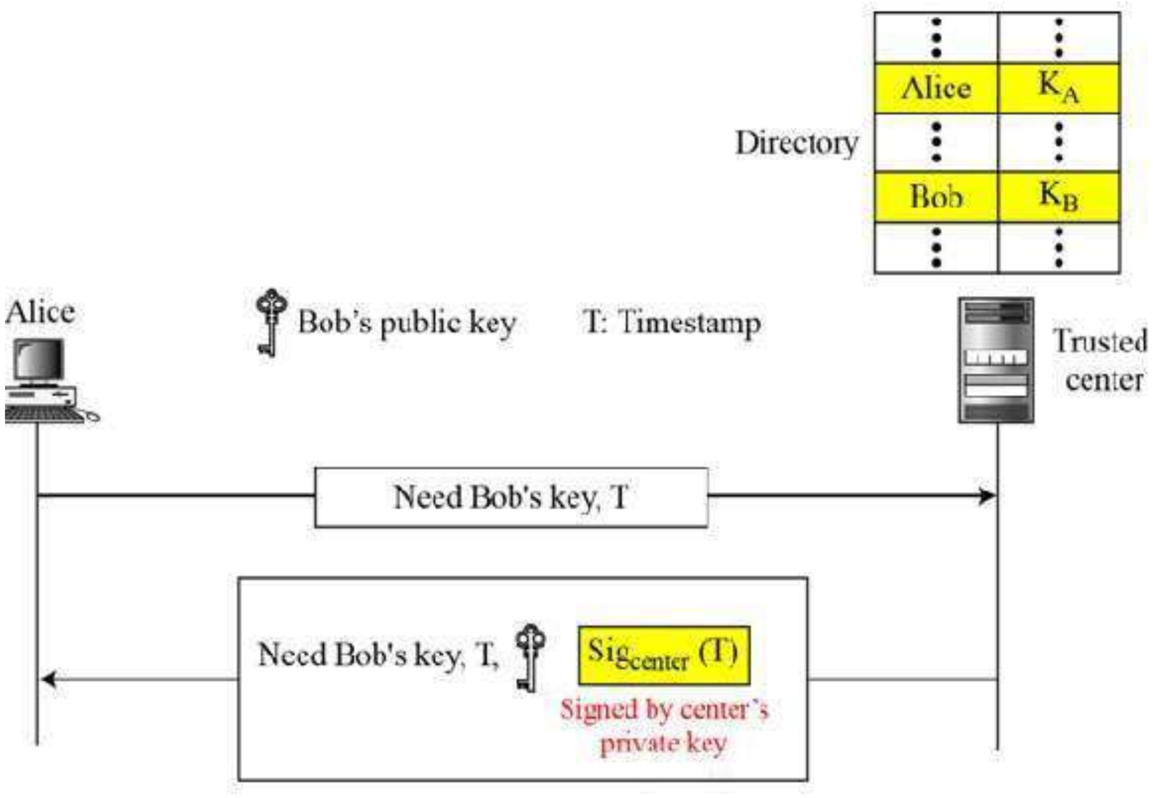
Trusted Center

A more secure approach is to have a trusted center retain a directory of public keys



Controlled Trusted Center

A higher level security can be achieved when there are added controls on



UNIT –V

Network Security-I

Security at application layer: PGP and S/MIME, Security at the Transport Layer: SSL and TLS

Security at Application layer and Transport Layer

Various business services are now offered online though client-server applications.
The most popular forms are web application and e-mail.
In both applications, the client communicates to the designated server and obtains services.

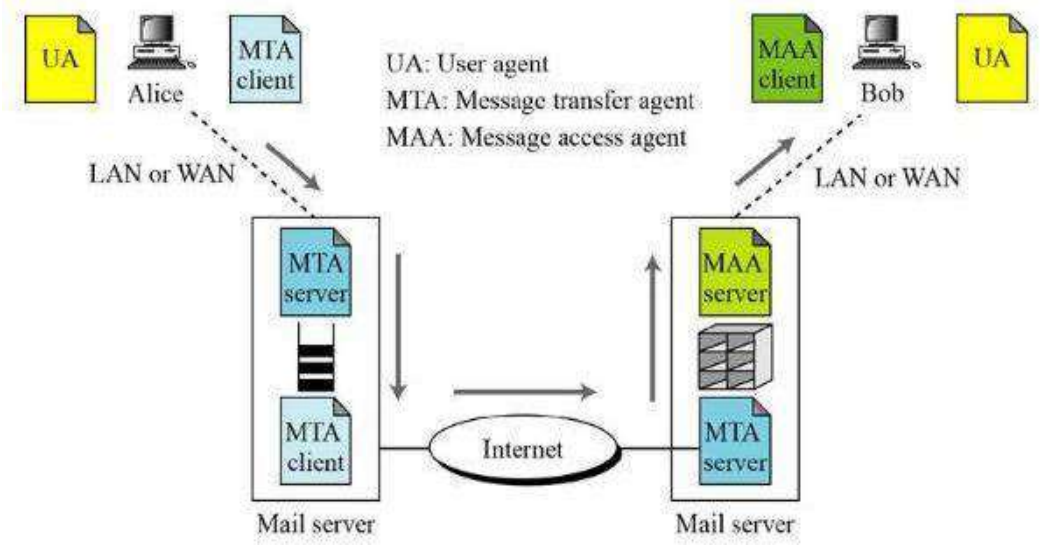
While using a service from any server application, the client and server exchange a lot of information on the underlying intranet or Internet. We are aware of fact that these information transactions are vulnerable to various attacks.
Network security entails securing data against attacks while it is in transit on a network.

E-mail Security

Nowadays, e-mail has become very widely used network application. Email is one of the most widely used and regarded network services. Currently message contents are not secure, may be inspected either in transit or by suitably privileged users on destination system.

E-mail Architecture:

Figure *E-mail architecture*



1. UA-User Agent is useful to prepare the messages
2. MTA-Message Transfer Agent is useful to send messages to mail server. This is the Push program
3. MAA-Message Access Agent is useful to receive messages from mail server. This is Pull program

PGP(Pretty Good Privacy)

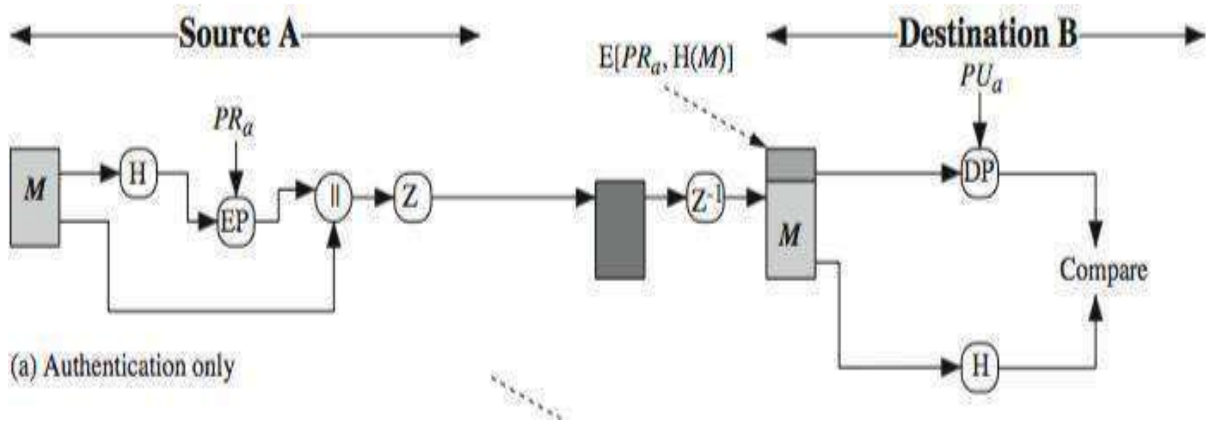
- Provides a confidentiality and authentication service that can be used for electronic mail and file storage applications
- Developed by Phil Zimmermann
- Selected the best available cryptographic algorithms as building blocks
- Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands
- Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks
- Entered into an agreement with a company to provide a fully compatible, low –cost commercial version of PGP

PGP Growth

It is available free worldwide in versions that run on a variety of platforms
• The commercial version satisfies users who want a product that comes with vendor support
• It is based on algorithms that have survived extensive public review and are considered extremely secure
• It has a wide range of applicability
• It was not developed by, nor is it controlled by, any governmental or standards organization
• Is now on an Internet standards track, however it still has an aura of an antiestablishment endeavor.

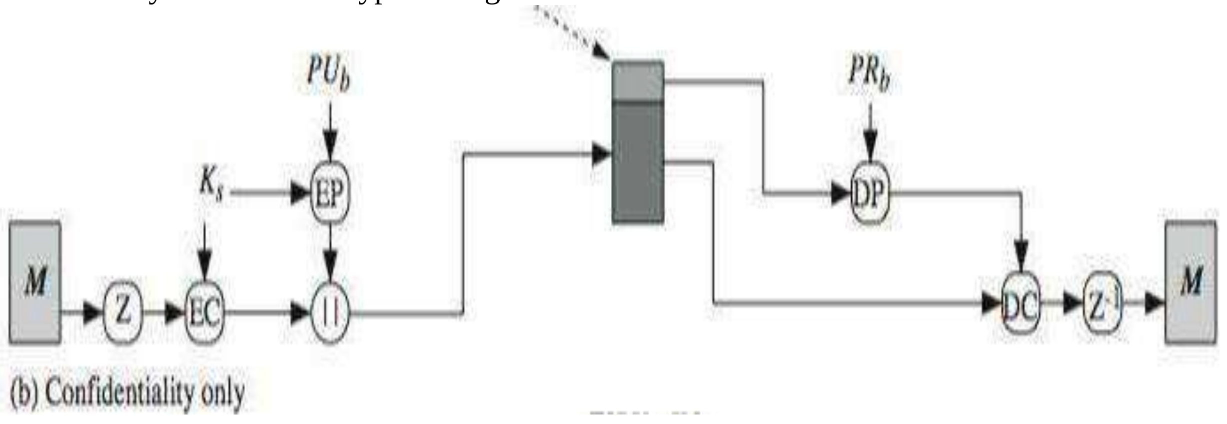
PGP Notation:
Ks = session key used in symmetric encryption scheme
PRa = private key of user A, used in public-key encryption scheme
PUa = public key of user A, used in public-key encryption scheme
EP = public-key encryption
DP = public-key decryption
EC = symmetric encryption
DC = symmetric decryption
H = hash function
|| = concatenation
Z = compression using ZIP algorithm
R64 = conversion to radix 64 ASCII format1

PGP Operation – Authentication:
1. sender creates a message
2. SHA-1 used to generate 160-bit hash code of message
3. hash code is encrypted with RSA using the sender's private key, and result is attached to message
4. receiver uses RSA or DSS with sender's public key to decrypt and recover hash code
5. receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic



PGP Operation – Confidentiality:

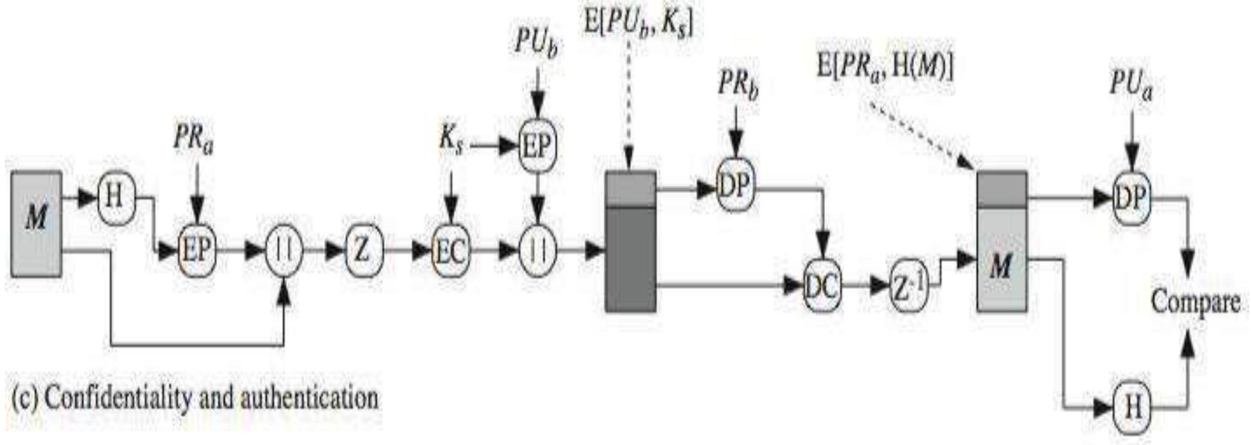
1. sender generates message and random 128-bit number to be used as session key for this message only
2. message is encrypted, using CAST-128 / IDEA/3DES with session key
3. session key is encrypted using RSA with recipient's public key, then attached to message
4. receiver uses RSA with its private key to decrypt and recover session key
5. session key is used to decrypt message



PGP Operation – Confidentiality & Authentication

Uses both services on same message

Create signature & attach to message o encrypt both message & signature o attach RSA encrypted session key



PGP Operation – Compression

As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.

The placement of the compression algorithm, indicated by Z for compression and Z^{-1} for decompression. So can store uncompressed message & signature for later verification & because compression is non deterministic uses ZIP compression algorithm

PGP Operation – Email Compatibility

- When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key).
- Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets.
- However, many electronic mail systems only permit the use of blocks consisting of ASCII text.
- To accommodate this restriction, PGP provides the service of converting the raw 8 -bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is radix-64 conversion.
- Each group of three octets of binary data is mapped into four ASCII characters. This format also appends

S/MIME (Secure/Multipurpose Internet Mail Extensions)

Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security. it appears likely that S/MIME will emerge as the industry standard for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many users. S/MIME is defined in a number of documents—most importantly RFCs 3370, 3850, 3851, and 3852.
S/MIME support in many mail agents eg MS Outlook, Mozilla, Mac Mail etc

To understand S/MIME, we need first to have a general understanding of the underlying e -mail format that it uses, namely MIME. We have to learn about RFC5322(internet Message Format)

RFC 5322:

- Defines a format for text messages that are sent using electronic mail
- Messages are viewed as having an envelope and contents
    - The envelope contains whatever information is needed to accomplish transmission and delivery
    - The contents compose the object to be delivered to the recipient
    - RFC 5322 standard applies only to the contents

The content standard includes a set of header fields that may be used by the mail system to create the envelope

The overall structure of a message that conforms to RFC 5322 is very simple. A message consists of some number of header lines (*the header*) followed by unrestricted text (*the body*). The header is separated from the body by a blank line. Put differently, a message is ASCII text, and all lines up to the first blank line are assumed to be header lines used by the user agent part of the mail system.

A header line usually consists of a keyword, followed by a colon, followed by the keyword's arguments; the format allows a long line to be broken up into several lines. The most frequently used keywords are *From*, *To*, *Subject*, and *Date*. Here is an example message:

Date: October 8, 2009 2:15:49 PM EDT

From: "William Stallings"
<ws@shore.net> Subject: The
Syntax in RFC 5322
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com

Hello. This section begins the actual message body, which is delimited from the message heading by a blank line.

Multipurpose Internet Mail Extensions (MIME):

An extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP) lists the following limitations of the SMTP/5322 scheme.

1. SMTP cannot transmit executable files or other binary objects.
2. SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations. The specification is provided in RFCs 2045 through 2049.

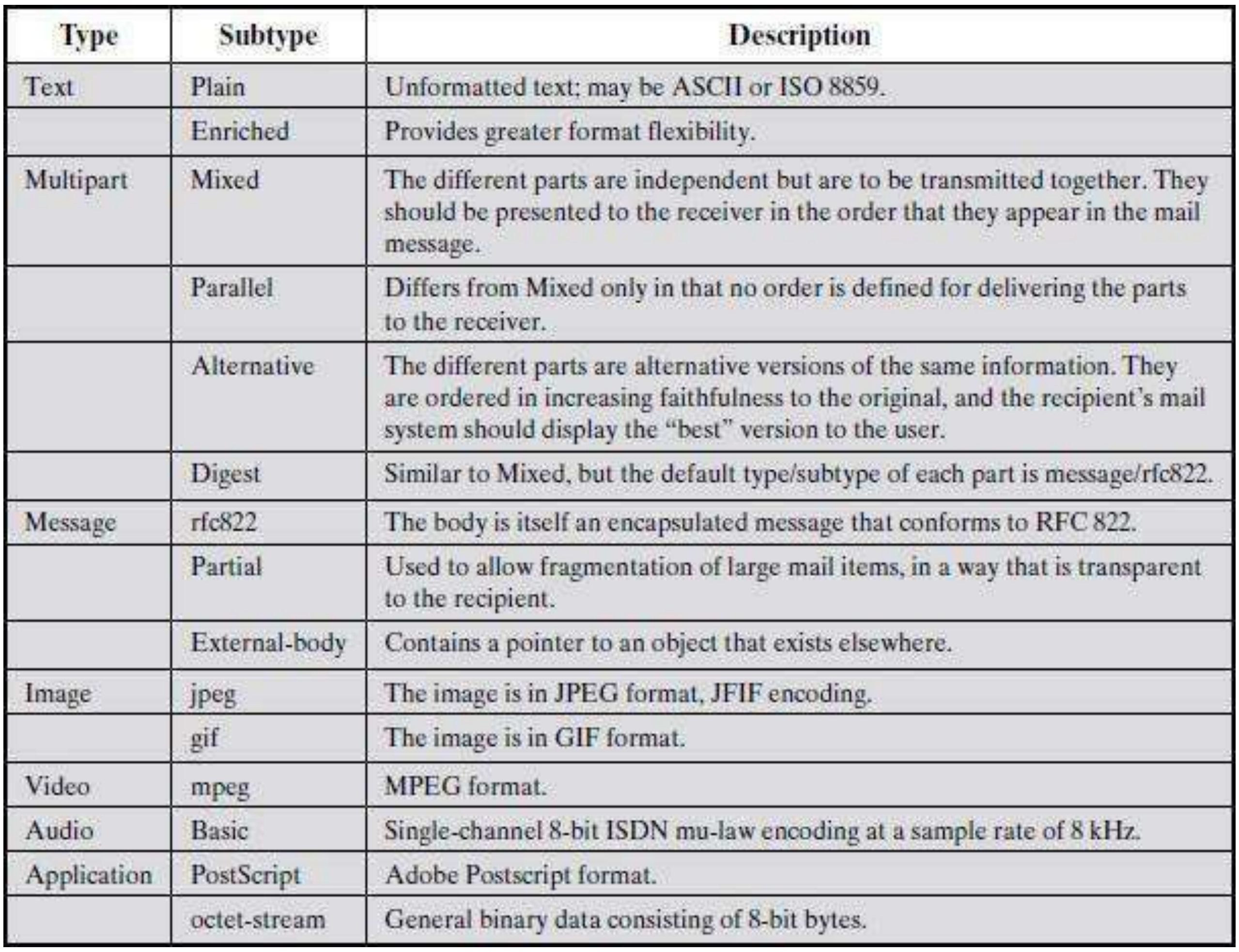
The MIME specification includes the following elements.

1. **Five new message header fields** are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.
2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

The Five Header Fields Defined in MIME: The five header fields defined in MIME are

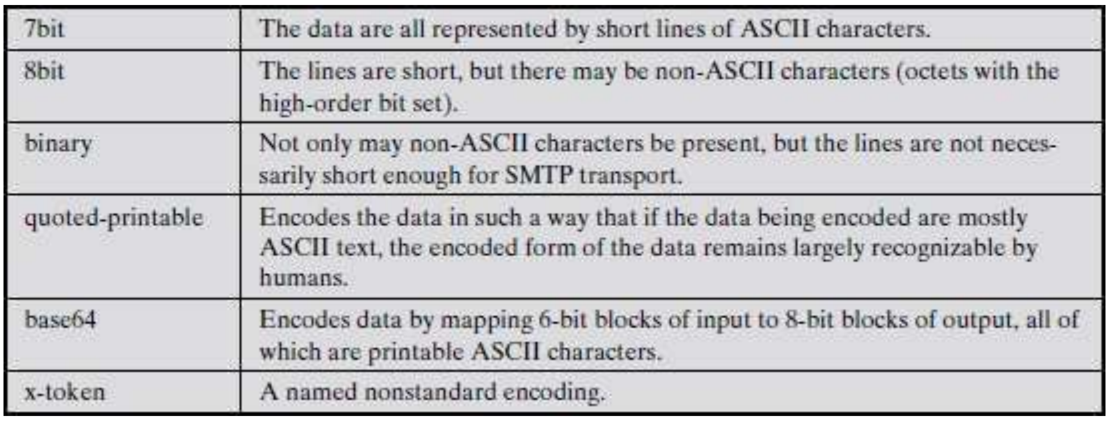
- **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME Content Types:



Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript format.
	octet-stream	General binary data consisting of 8-bit bytes.

MIME Transfer Encodings:



7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present, but the lines are not necessarily short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.
x-token	A named nonstandard encoding.

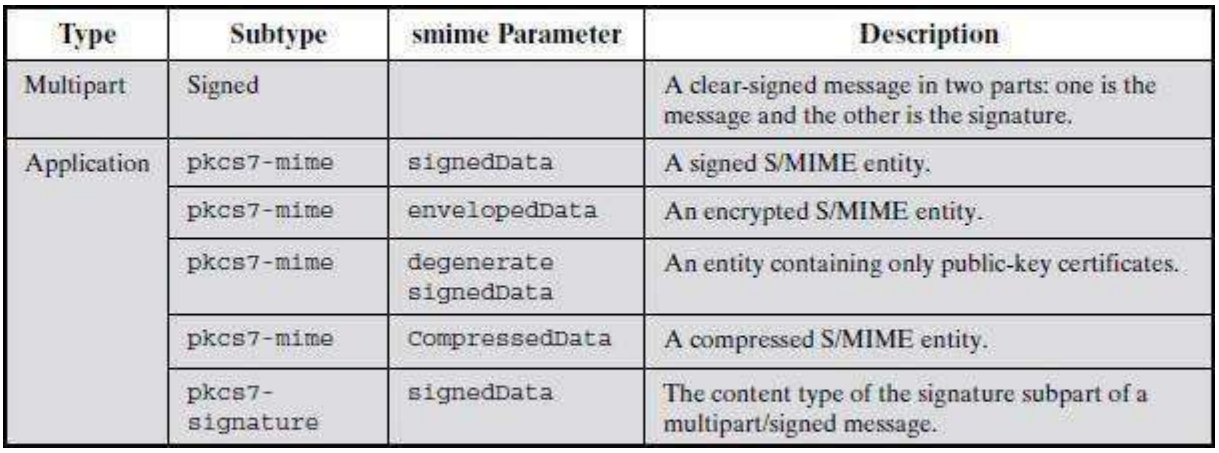
S/MIME Functionality: S/MIME provides the following functions.

- **Enveloped data:** This consists of encrypted content of any type and encrypted content encryption keys for one or more recipients.
- **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
- **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear - signed data may be encrypted.

S/MIME Messages:

- S/MIME secures a MIME entity with a signature, encryption, or both. forming a MIME wrapped
  Public-Key Cryptography Standards(PKCS) object have a range of content-types:
  enveloped data o signed data, clear-signed data o registration request,certificate only message

  S/MIME Content Types



Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs7-mime	signedData	A signed S/MIME entity.
	pkcs7-mime	envelopedData	An encrypted S/MIME entity.
	pkcs7-mime	degenerate signedData	An entity containing only public-key certificates.
	pkcs7-mime	CompressedData	A compressed S/MIME entity.
	pkcs7-signature	signedData	The content type of the signature subpart of a multipart/signed message.

S/MIME Certificate Processing:

- S/MIME uses public-key certificates that conform to version 3 of X.509
- The key-management scheme used by S/MIME is in some ways a hybrid between a strict X.509 certification hierarchy and PGP's web of trust
- S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists.
  The responsibility is local for maintaining the certificates needed to verify incoming signatures and to encrypt outgoing messages
- The certificates are signed by certification authorities

User Agent Role An S/MIME user has several key-management functions to perform

• **Key generation:** The user of some related administrative utility (e.g., one associated with LAN management) MUST be capable of generating separate Diffie-Hellman and DSS key pairs and SHOULD be capable of generating RSA key pairs. Each key pair MUST be generated from a good source of nondeterministic random input and be protected in a secure fashion. A use agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.
• **Registration:** A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.
• **Certificate storage and retrieval:** A user requires access to a local list of certificates in order to verify incoming signatures and to encrypt outgoing messages. Such a list could be maintained by the user or by some local administrative entity on behalf of a number of users.

VeriSign Certificates There are several companies that provide certification authority (CA) services. For example, Nortel has designed an enterprise CA solution and can provide S/MIME support within an organization. There are a number of Internet-based CAs, including VeriSign, GTE, and the U.S. Postal Service.

Enhanced Security Services : three enhanced security services have been proposed in an Internet draft. The three services are : **Signed receipts, Security labels, Secure mailing lists**

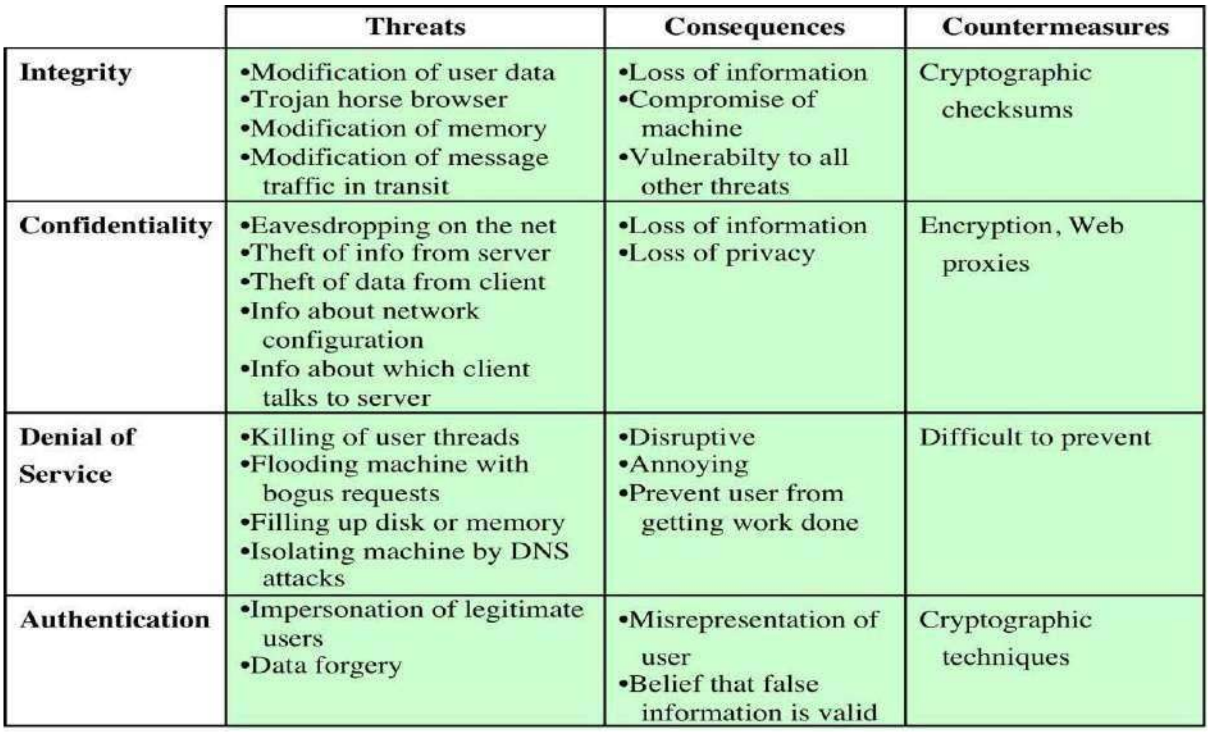
Transport Level Security:

Web security considerations:

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets
The following characteristics of Web usage suggest the need for tailored security tools:

- The Internet is two-way. Unlike traditional publishing environments—even electronic publishing systems involving teletext, voice response, or fax-back— the Web is vulnerable to attacks on the Web servers over the Internet.
- The Web is increasingly serving as a highly visible outlet for corporate and product information and as the platform for business transactions. Reputations can be damaged and money can be lost if the Web servers are subverted.
- Although Web browsers are very easy to use, Web servers are relatively easy to configure and manage, and Web content is increasingly easy to develop, the underlying software is extraordinarily complex. This complex software may hide many potential security flaws.
- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.
- Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

Web security Threats:



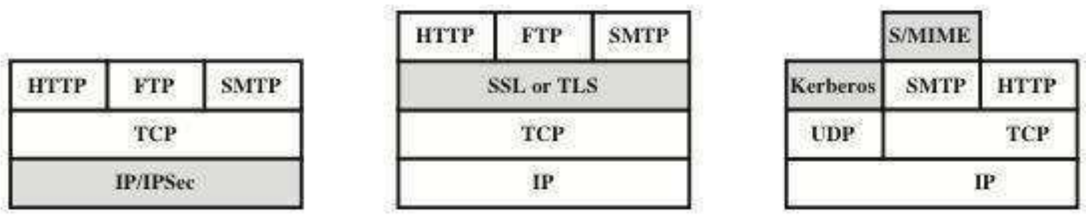
	Threats	Consequences	Countermeasures
Integrity	•Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit	•Loss of information •Compromise of machine •Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	•Eavesdropping on the net •Theft of info from server •Theft of data from client •Info about network configuration •Info about which client talks to server	•Loss of information •Loss of privacy	Encryption, Web proxies
Denial of Service	•Killing of user threads •Flooding machine with bogus requests •Filling up disk or memory •Isolating machine by DNS attacks	•Disruptive •Annoying •Prevent user from getting work done	Difficult to prevent
Authentication	•Impersonation of legitimate users •Data forgery	•Misrepresentation of user •Belief that false information is valid	Cryptographic techniques

Table. A Comparison of Threats on the Web

Web Traffic Security Approaches:

A number of approaches to providing Web security are possible.

1. One way to provide Web security is to use **IP security** (IPsec) (Figure(a)). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution. It includes filtering capability that filters the unwanted data.
2. Another relatively general-purpose solution is to implement security just above TCP (Figure (b)). The example of this approach is the **Secure Sockets Layer (SSL)** and the follow-on Internet standard known as **Transport Layer Security (TLS).** At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.
3. Application-specific security services are embedded within the particular application. Figure (c) shows examples of this architecture. The advantage of this approach is that the service can be tailored to the specific needs of a given application.



(a) Network Level (b) Transport Level (c) Application Level

Figure: relative location of security facilities in the TCP/IP Protocol stack **5.4.**

SSL (Secure Socket Layer):

SSL probably most widely used Web security mechanism, and it is implemented at the Transport layer.

SSL is designed to make use of TCP to provide a reliable end-to-end secure service.

Netscape originated SSL. Version 3 of the protocol was designed with public review and input from industry and was published as an Internet draft document. Subsequently, became Internet standard known as TLS (Transport Layer Security)

SSL Architecture:

- SSL is designed to make use of TCP to provide a reliable end-to-end
- secure service. SSL is not a single protocol but rather two layers of protocols.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

1. **Connection:** A connection is a transport that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. Every connection is associated with one session.
2. **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections.

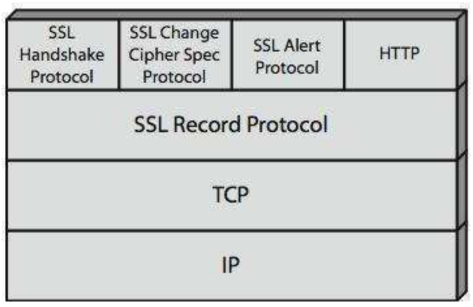


Figure: SSL Protocol stack

SSL Record Protocol:

SSL Record Protocol defines two services for SSL connections:

1. **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. The message is compressed before being concatenated with the MAC and encrypted, with a range of ciphers being supported as shown.
2. **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

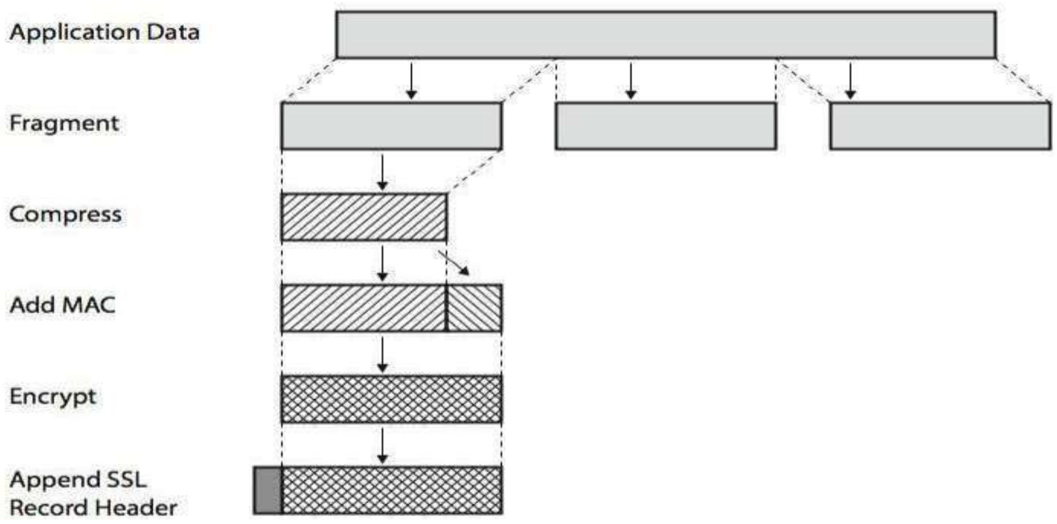


Figure: SSL Record Protocol Operation

Figure shows the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.

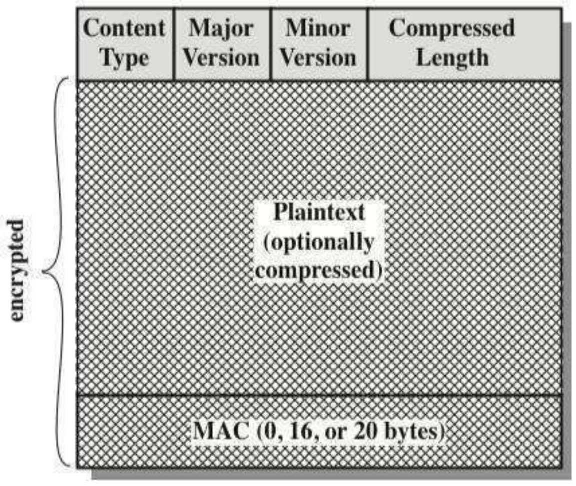


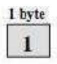
Figure: SSL Record Format

The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields:

- **Content Type (8 bits):** The higher-layer protocol used to process the enclosed fragment.
- **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
- **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
- **Compressed Length (16 bits):** The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is $2^{14} + 2048$.

ChangeCipherSpec Protocol:

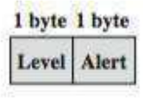
The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol. It is the simplest, consisting of a single message, which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.



(a) Change Cipher Spec Protocol

SSL Alert Protocol:

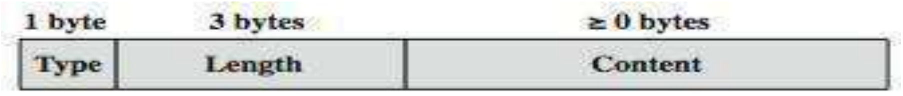
The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol consists of two bytes, the first takes the value warning (1) or fatal (2) to convey the severity of the message. The second byte contains a code that indicates the specific alert.



(b) Alert Protocol

SSL Handshake Protocol:

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by client and server.



(c) Handshake Protocol

The exchange can be viewed in 4 phases:

- **Phase 1. Establish Security Capabilities** - this phase is used by the client to initiate a logical connection and to establish the security capabilities that will be associated with it
- **Phase 2. Server Authentication and Key Exchange** - the server begins this phase by sending its certificate if it needs to be authenticated.
- **Phase 3. Client Authentication and Key Exchange** - the client should verify that the server provided a valid certificate if required and check that the server_hello parameters are acceptable
- **Phase 4. Finish** - this phase completes the setting up of a secure connection. The client sends a change_cipher_spec message and copies the pending CipherSpec into the current CipherSpec. At this point the handshake is complete and the client and server may begin to exchange application layer data.

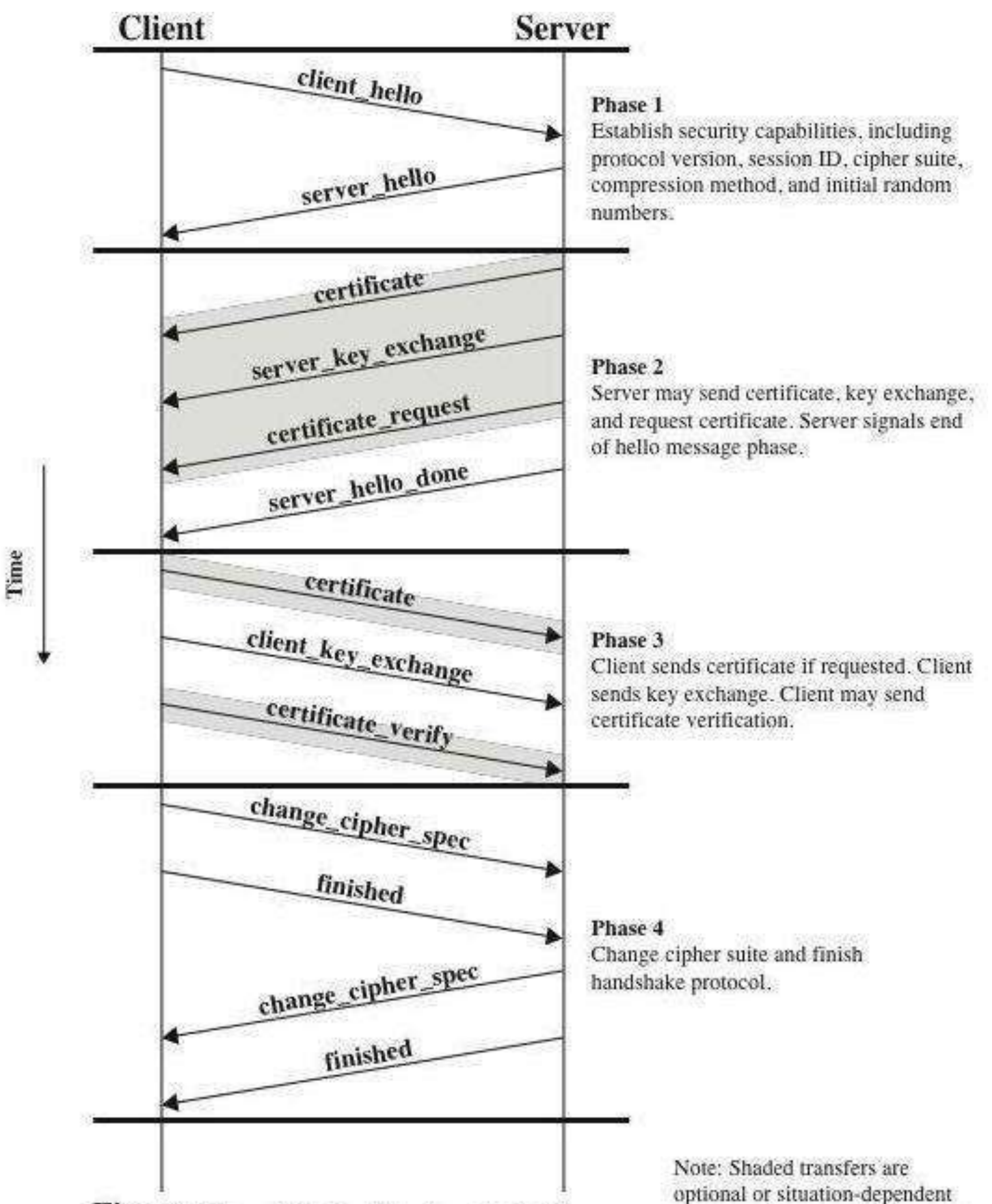

Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Figure Handshake Protocol Action

Transport Layer Security (TLS) Protocol

In order to provide an open Internet standard of SSL, Internet Engineering Task Force(IETF) released The Transport Layer Security (TLS) protocol in January 1999. TLS is defined as a proposed Internet Standard in RFC 5246.

Salient Features

TLS protocol has same objectives as SSL.

It enables client/server applications to communicate in a secure manner by authenticating, preventing eavesdropping and resisting message modification.

TLS protocol sits above the reliable connection-oriented transport TCP layer in the networking layers' stack.

The architecture of TLS protocol is similar to SSLv3 protocol. It has two sub protocols: the TLS Record protocol and the TLS Handshake protocol.

Though SSLv3 and TLS protocol have similar architecture, several changes were made in architecture and functioning particularly for the handshake protocol.

Comparison of TLS and SSL Protocols:

1. **Protocol Version −** The header of TLS protocol segment carries the version number 3.1 todifferentiate between number 3 carried by SSL protocol segment header.
2. **Message Authentication −** TLS employs a keyed-hash message authentication code (HMAC). Benefit is that H-MAC operates with any hash function, not just MD5 or SHA, as explicitly stated by the SSL protocol.
3. **Session Key Generation −** There are two differences between TLS and SSL protocol for generation of key material.
    1. Method of computing pre-master and master secrets is similar. But in TLS protocol, computation of master secret uses the HMAC standard and pseudorandom function (PRF) output instead of ad-hoc MAC.
    2. The algorithm for computing session keys and initiation values (IV) is different in TLS than SSL protocol.
4. **Alert Protocol Message −**
    1. TLS protocol supports all the messages used by the Alert protocol of SSL, except *No certificate* alert message being made redundant. The client sends empty certificate in case client authentication is not required.
    2. Many additional Alert messages are included in TLS protocol for other error conditions such as
       *record_overflow, decode_error*etc.
5. **Supported Cipher Suites −** SSL supports RSA, Diffie-Hellman and Fortezza cipher suites. TLS protocol supports all suits except Fortezza.
6. **Client Certificate Types −** TLS defines certificate types to be requested in a *certificate_request* message. SSLv3 support all of these. Additionally, SSL support certain other types of certificate such as Fortezza.
7. **CertificateVerify and Finished Messages −**

1. In SSL, complex message procedure is used for the*certificate_verify* message. With TLS, the verified information is contained in the handshake messages itself thus avoiding this complex procedure.
2. Finished message is computed in different manners in TLS and SSLv3.

8. **Padding of Data −** In SSL protocol, the padding added to user data before encryption is the minimum amount required to make the total data-size equal to a multiple of the cipher's block length. In TLS, the padding can be any amount that results in data-size that is a multiple of the cipher's block length, up to a maximum of 255 bytes.

Secure Shell Protocol (SSH):

The salient features of SSH are as follows −

- SSH is a network protocol that runs on top of the TCP/IP layer. It is designed to replace the TELNET which provided unsecure means of remote logon facility.
- SSH provides a secure client/server communication and can be used for tasks such as file transfer and e-mail.
- SSH2 is a prevalent protocol which provides improved network communication security over earlier version SSH1.

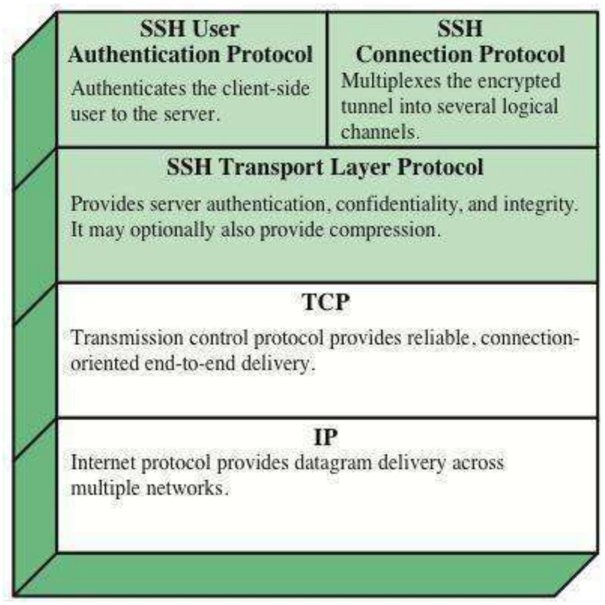

SSH User Authentication Protocol
Authenticates the client-side user to the server.
SSH Connection Protocol
Multiplexes the encrypted tunnel into several logical channels.
SSH Transport Layer Protocol
Provides server authentication, confidentiality, and integrity. It may optionally also provide compression.
TCP
Transmission control protocol provides reliable, connection-oriented end-to-end delivery.
IP
Internet protocol provides datagram delivery across multiple networks.

Figure: SSH Protocol stack

Transport Layer Protocol:
In this part of SSH protocol provides data confidentiality, server (host) authentication, and data integrity. It may optionally provide data compression as well.

- Server Authentication − Host keys are asymmetric like public/private keys. A server uses a public
key to prove its identity to a client. The client verifies that contacted server is a ―known‖ host from the database it maintains. Once the server is authenticated, session keys are generated.

 Session Key Establishment − After authentication, the server and the client agree upon cipher to be
used. Session keys are generated by both the client and the server. Session keys are generated before user authentication so that usernames and passwords can be sent encrypted. These keys are generally replaced at regular intervals (say, every hour) during the session and are destroyed immediately after use.
 Data Integrity − SSH uses Message Authentication Code (MAC) algorithms to for data integrity check. It is an improvement over 32 bit CRC used by SSH1.

User Authentication Protocol:

In this part of SSH authenticates the user to the server. The server verifies that access is given to intended users only. Many authentication methods are currently used such as, typed passwords, Kerberos, public-key authentication, etc.

Connection Protocol:

This provides multiple logical channels over a single underlying SSH connection

SSH Services:

SSH provides three main services that enable provision of many secure solutions. These services are briefly described as follows −

Secure Command-Shell (Remote Logon) − It allows the user to edit files, view the contents of directories, and access applications on connected device. Systems administrators can remotely start/view/stop services and processes, create user accounts, and change file/directories permissions and so on. All tasks that are feasible at a machine's command prompt can now be performed securely from the remote machine using secure remote logon.

Secure File Transfer − SSH File Transfer Protocol (SFTP) is designed as an extension for SSH-2 for secure file transfer. In essence, it is a separate protocol layered over the Secure Shell protocol to handle file transfers. SFTP encrypts both the username/password and the file data being transferred. It uses the same port as the Secure Shell server, i.e. system port no 22.

Port Forwarding (Tunneling) − It allows data from unsecured TCP/IP based applications to be secured. After port forwarding has been set up, Secure Shell reroutes traffic from a program (usually a client) and sends it across the encrypted tunnel to the program on the other side (usually a server). Multiple applications can transmit data over a single multiplexed secure channel, eliminating the need to open many ports on a firewall or router.

UNIT -VI:
Network Security-II : Security at the Network Layer: IPSec, System Security

1. IP SECURITY OVERVIEW

IPSec is an Internet Engineering Task Force (IETF) standard suite of protocols that provides data authentication, integrity, and confidentiality as data is transferred between communication points across IP networks.
IPSec provides data security at the IP packet level. A packet is a data bundle that is organized for transmission across a network, and it includes a header and payload (the data in the packet).

IPSec SECURITY FEATURES:
IPSec is the most secure method commercially available for connecting network sites.
IPSec was designed to provide the following security features when transferring packets across networks:
Authentication: Verifies that the packet received is actually from the claimed sender.
Integrity: Ensures that the contents of the packet did not change in transit.
Confidentiality: Conceals the message content through encryption.

IPSec ELEMENTS:
IPSec contains the following elements:
Encapsulating Security Payload (ESP): Provides confidentiality, authentication, and integrity.
Authentication Header (AH): Provides authentication and integrity.
Internet Key Exchange (IKE): Establish shared symmetric key. Provides key management and Security Association (SA) management.

APPLICATIONS OF IPSec:
IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet.
Examples of its use include the following:
- Secure branch office connectivity over the Internet
- Secure remote access over the Internet

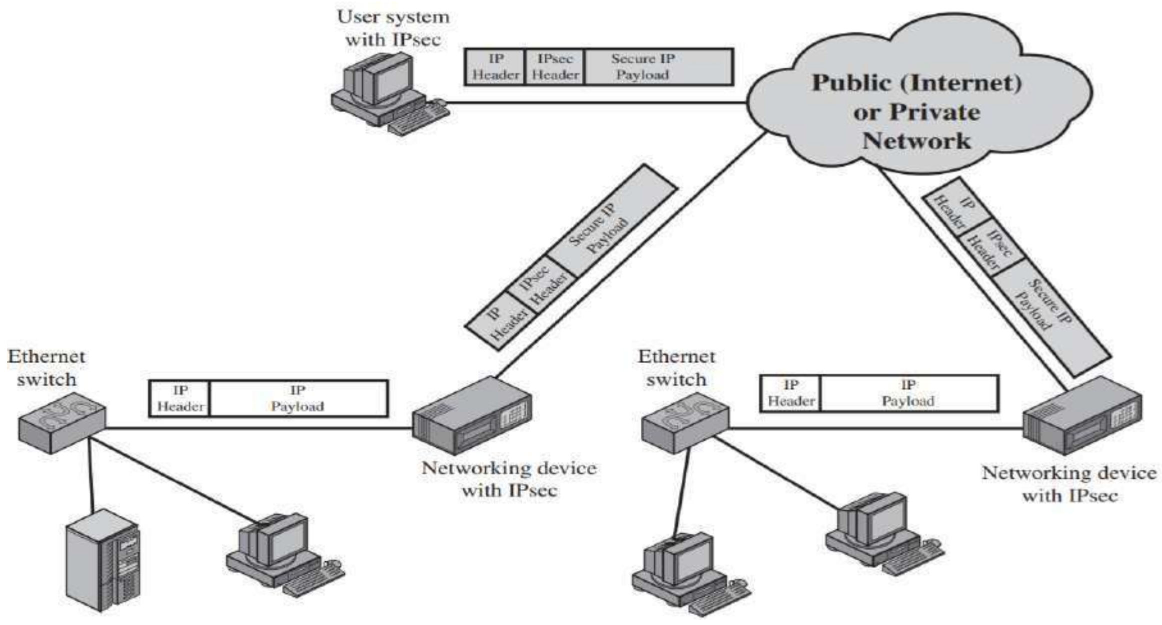
Establishing extranet and intranet connectivity with partners:
- IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.

Enhancing electronic commerce security:
- Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

BENEFITS OF IPSEC:
- IPSec provides strong security within and across the LANs.
- Firewall uses IPSec to restrict all those incoming packets which are not using IP. Since firewall is the only way to enter into an organization, restricted packets cannot enter.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications.
- There is no need to change software on a user or server system when IPSec is implemented in the firewall or router.
- Even if IPSec is implemented in end systems, upper- layer software, including applications, is not affected. IPSec can be transparent to end users.

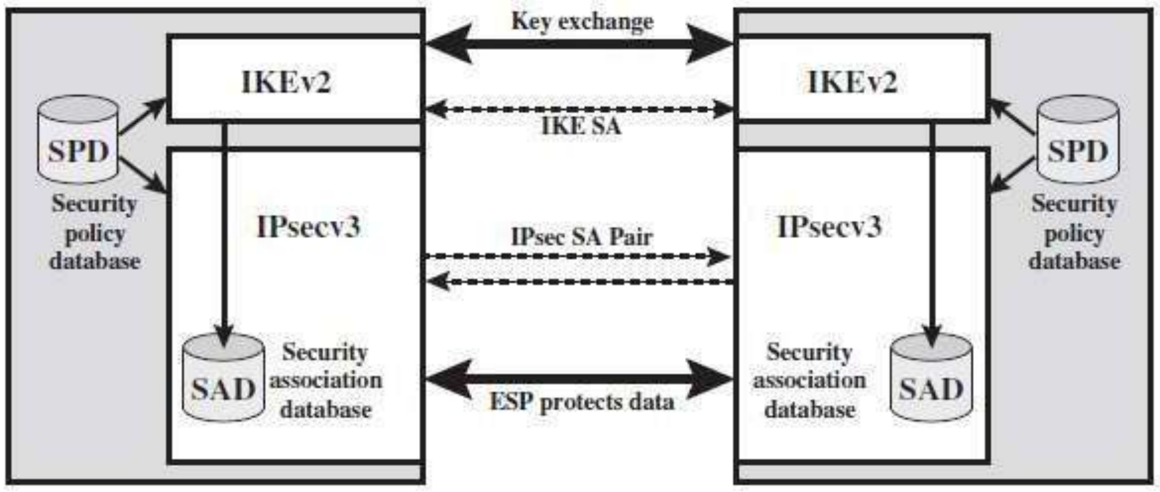
- IPSec can provide security for individual users if needed.

IPSec Scenario:



IPSec Architecture:

Architecture covers general concepts of security requirements, definitions, and mechanisms defining IPSec technology.



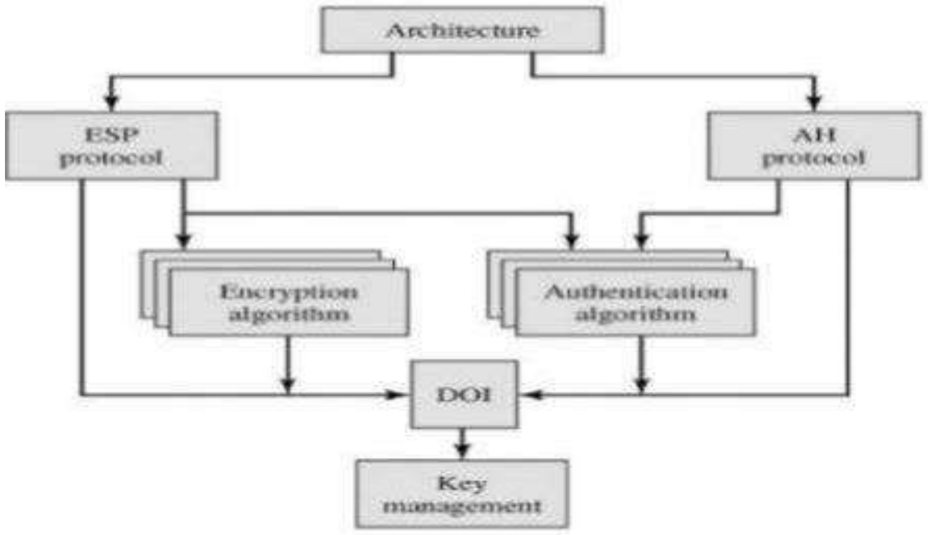


Figure : IPSec Architecture

Encapsulating Security Payload(ESP): The ESP header is designed to provide a mix of security services in IPv4 and IPv6. ESP may be applied alone, in combination with AH, or in a nested fashion. It consists of an encapsulating header and trailer used to provide encryption or combined encryption/authentication. Current specification is RFC 4303

Authentication Header(AH): An extension header to provide message authentication. Current specification is RFC 4302.

Encryption algorithms: Encryption algorithms encrypt data with a key. The ESP module in IPsec uses encryption algorithms.

Authentication algorithms: Authentication algorithms produce an integrity checksum value or digest that is based on the data and a key. The AH module uses authentication algorithms. The ESP module can use authentication algorithms as well.

Domain of Interpretation(DOI): DOI is the identifier which support both AH and ESP protocols. It contains values needed for documentation related to each other.

Key Management: It contains the document that describes how the keys are exchanged between sender and receiver.

Security Associations (SAs)

An SA is a relationship between communicating devices that describes how they will use security services to communicate securely.
If client wants to communicate with server, it has client Security Association, if Server wants to reply to client, it has server Security association.
These SAs are one way communications.
If two parties need to communicate, they must determine which algorithms (RSA, 3DES, MD5,

SHA…) and session keys are used. SA used by IPsec to track all these parameters for each session. You will need to configure SA parameters and monitor SAs on Cisco routers and the PIX Firewall.

- A separate pair of IPSec SAs are set up for AH and ESP transform.
- Each IPSec peer agrees to set up SAs consisting of policy parameters to be used during the IPSec session.
- The SAs are unidirectional for IPSec so that peer 1 will offer peer 2 a policy.
- If peer 2 accepts this policy, it will send that policy back to peer 1. This establishes two one-way SAs between the peers.
- Two-way communication consists of two SAs, one for each direction.
- Each SA consists of values such as destination address, a security parameter index (SPI), the IPSec transforms used for that session, security keys, and additional attributes such as IPSec lifetime.

A security association is uniquely identified by three parameters:

- **Security Parameters Index (SPI):** A bit string assigned to this SA and having local significance only. SPI is located in AH and ESP headers. SPI enables the receiving system under which the packet is to process.
- **IP Destination Address:** It is the end point address of SA which can be end user system or a network system.
- **Security Protocol Identifier:** security protocol identifier indicates whether the associations is an AH or ESP.

All the SAs are maintained in Security Association Database(SAD)

SA Parameters:

Sequence Number Counter: A 32-bit value used to generate the Sequence Number field in AH or ESP headers.

Sequence Counter Overflow: A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA.

Anti-Replay Window: Avoid duplicate of packets

AH Information: Authentication algorithm, keys, key lifetimes, and related parameters being used with AH.

ESP Information: Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations).

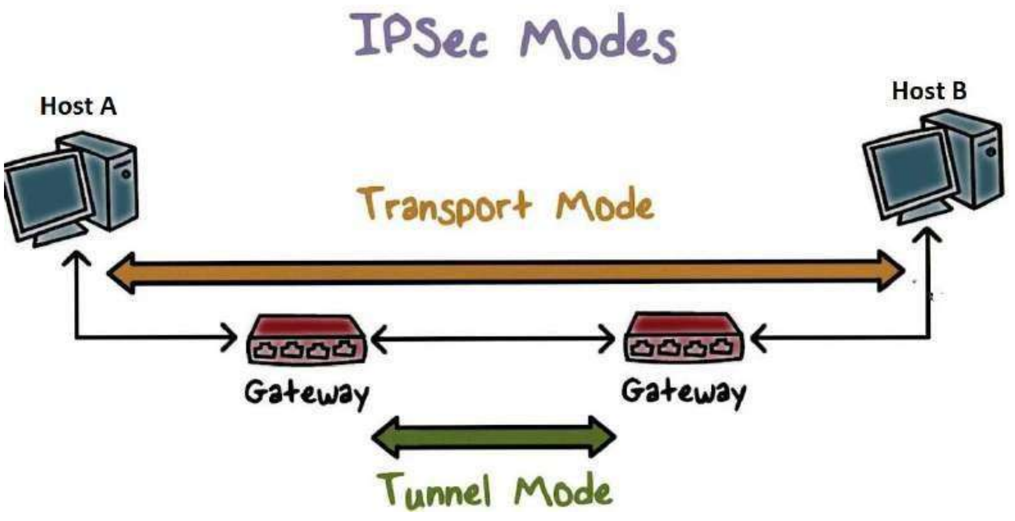
Lifetime of This Security Association: A time interval or byte count after which an SA must be replaced with a new SA or terminated.

IPSec Protocol Mode: This parameter represents the type of mode used for IPSec implementation. The mode may be a Tunnel or transport.

Transport and Tunnel Modes in IPsec

IPSec operates in two modes:

1)Tunnel Mode
2) Transport Mode



Tunnel Mode:
With tunnel mode, the entire original IP packet is protected by IPSec. This means IPSec wraps the original packet, encrypts it, adds a new IP header and sends it to the other side.
Original IP Header not visible to attacker(if it is using ESP).
Attacker does not know which hosts are talking.

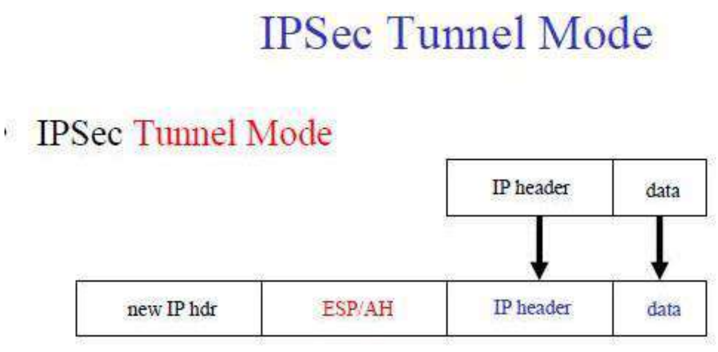


Figure : IPSec Tunnel mode

Tunnel mode is most commonly used between gateways, end-system to Gateways.

Transport Mode:

IPSec Transport mode is used for end-to-end communications, for example, for communication between a client and a server or between a workstation and a gateway (if the gateway is being treated as a host).

When using the transport mode, only the IP payload is encrypted. AH or ESP provides protection for the IP payload. The original IP header is not changed,
So the passive attackers can see who is talking.

- IPSec **Transport Mode**

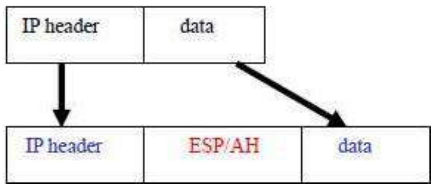


Figure : IPSec Transport Mode

AUTHENTICATION HEADER (AH)

The Authentication Header provides support for data integrity and authentication of IP packets.
Data integrity service insures that data inside IP packets is not altered during the transit.
The authentication feature enables an end system to authenticate the user or application and filter traffic accordingly. It also prevents the **address spoofing attacks**
AH is implemented in one way only i.e Authentication along with Integrity.
AH provides authentication for as much of the IP header as possible, but cannot all be protected by AH.
AH also includes an IPSec sequence number, which provides protection against replay attacks because this number is also included in authenticated data and can be checked by the receiving party.
Data privacy is not provided by AH.

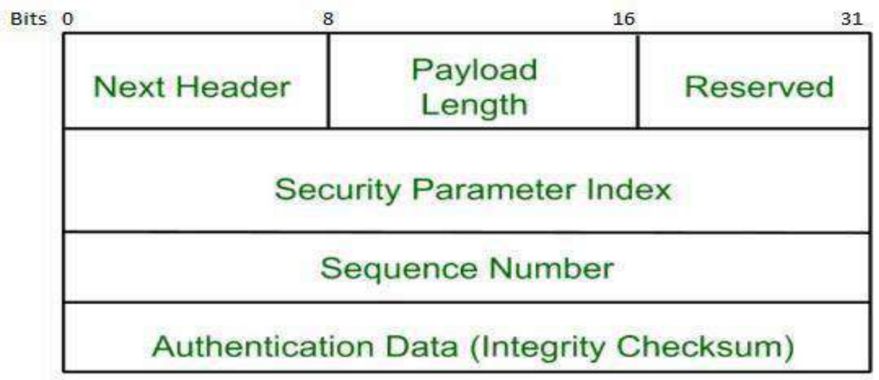


Figure : Authentication Header Format

1. Next Header: Identifies the type of header that immediately following the AH.
2. Payload Length: Length of Authentication Header in 32-bit words.
3. Reserved: For future use.
4. Security Parameters Index: Identifies a security association.
5. Sequence Number: A monotonically increasing counter value.
6. Authentication Data (variable): A variable-length field that contains the Integrity Check Value (ICV), or MAC, for this packet.

Encapsulating Security Payload(ESP):

Security services can be provided between a pair of communicating hosts, between a pair of communicating security gateways, or between a security gateway and a host. The ESP header is inserted after the IP header and before the next layer protocol header (transport mode) or before an encapsulated IP header (tunnel mode). ESP can be used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and (limited) traffic flow confidentiality. The set of services provided depends on options selected at the time of Security Association (SA) establishment and on the location of the implementation in a network topology.

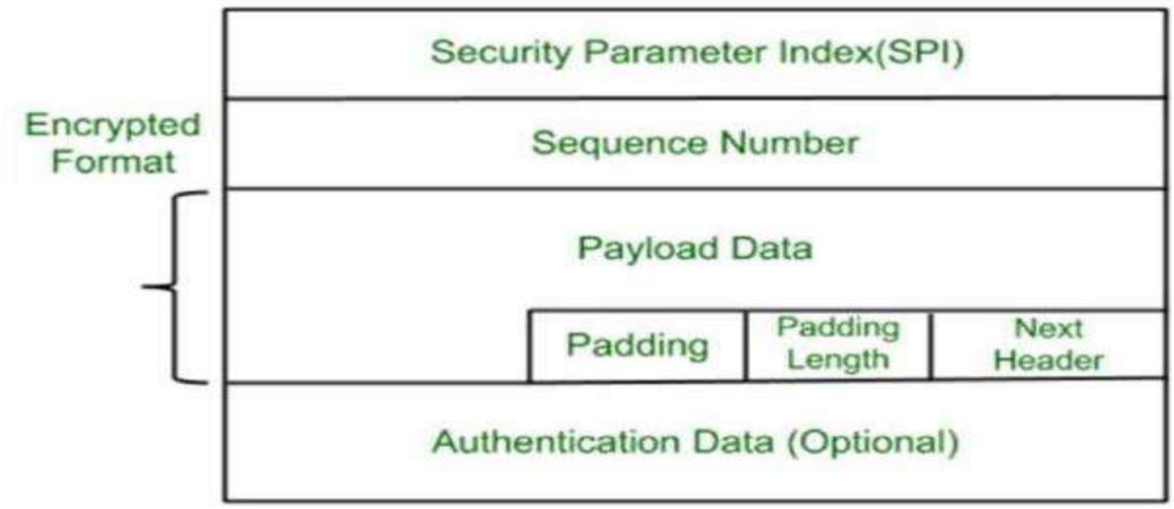


Figure : ESP Format

1. Security Parameters Index : Identifies a security association.
2. Sequence Number : A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
3. Payload Data : This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
4. Padding (0-255 bytes):Extra bits or spaces are added to the message in order to maintain confidentiality
5. Pad Length : Indicates the number of pad bytes immediately preceding this field.
6. Next Header : means the next payload or next data
7. Authentication Data (variable): contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.

Security Policy(SP)

A Security Policy is a set of rules that define the type security applied to a packet when it is to be sent or when it has arrived. It defines the network traffic at the IP layer.

IPSec protects your private network from internet attacks through end-to-end security.

IPSec policy is determined primarily by the interaction of two databases, the Security Association Database(SAD) and the Security Policy Databases(SPD)

IPSec policies must be carefully designed, configures, coordinated and managed to ensure that IPSec communication is successful.

Security Policy Database (SPD)

IPSec Policies are maintained in the Security Policy Database (SPD).

IPSec Policies defines which traffic to be protected, how it is to be protected, and with whom to protect it.

The sending host determines what policy is appropriate for the packet, depending on various "Selectors" by checking in the Security Policy Database (SPD).

"Selectors" can include Source and Destination IP Addresses, Name (User ID ir a System Name), Transport Layer Protocols (TCP or UDP) or Source and Destination Ports.

The Security Policy Database (SPD) indicates what the policy is for a particular packet. If the packet requires IPsec processing, it will be it is passed to the IPsec module for the required processing.

KEY MANAGEMENT of IPSec

The key management portion of IPSec involves the determination and distribution of secret keys

typical requirement is four keys for communication between two applications: transmit and receive pairs for both AH and ESP.

Keys are managed by

- Manual: A system administrator manually configures each system with its own keys and with the keys of other communicating systems. This is suitable for small, relatively static environments.
- Automated: An automated system enables the on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system.

The default automated key management protocol for IPSec is referred to as ISAKMP/Oakley .

Key management protocol – Elements

1. Oakley Key Determination Protocol

2. Internet Security Association and Key Management Protocol (ISAKMP)

<u>Oakley Key Determination Protocol:</u>

- Oakley is a key exchange protocol based on the Diffie-Hellman algorithm but providing added security.
- Oakley is generic in that it does not dictate specific formats.

The Diffie-Hellman algorithm has **two attractive features:**

1. Secret keys are created only when needed.

2. The exchange requires no preexisting infrastructure other than an agreement on the global parameters. However, there are a number of weaknesses to Diffie-Hellman, as pointed out in

3. It does not provide any information about the identities of the parties.

4. It is subject to a man-in-the-middle attack

It is computationally intensive. As a result, it is vulnerable to a clogging attack, in which an opponent requests a high number of keys. Oakley is designed to retain the advantages of Diffie-Hellman while countering its weaknesses.

Features of Oakley:

The Oakley algorithm is characterized by five important features:

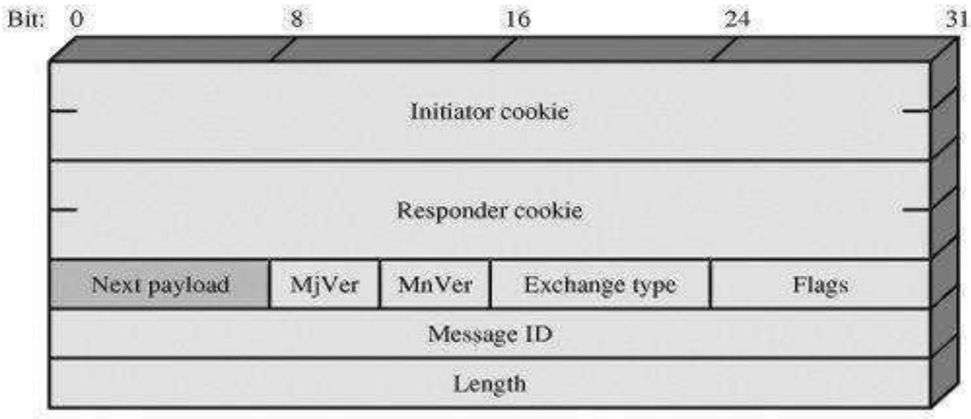
- It employs a mechanism known as cookies to thwart clogging attacks.
- It enables the two parties to negotiate a group; this, in essence, specifies the global parameters of the Diffie-Hellman key exchange.
- It uses nonces to ensure against replay attacks.
- It enables the exchange of Diffie-Hellman public key values.
- It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks.

Internet Security Association and Key Management Protocol (ISAKMP):

ISAKMP provides a framework for Internet key management and provides the specific protocol support, including formats, for negotiation of security attributes.

ISAKMP Header Format:

An ISAKMP message consists of an ISAKMP header followed by one or more payloads. All of this is carried in a transport protocol. The specification dictates that implementations must support the use of UDP for the transport protocol.



(a) ISAKMP header

It consists of the following fields:
1. Initiator Cookie (64 bits): Cookie of entity that initiated SA establishment, SA notification, or SA deletion.
2. Responder Cookie (64 bits): The cookie of entity that is responding to an SA establishment request, SA notification, or SA deletion. On the first message, the responder cookie iszero.
3. Next Payload (8 bits): Indicates the type of the first payload in the message
4. Major Version (4 bits): Indicates major version of ISAKMP in use.
5. Minor Version (4 bits): Indicates minor version in use.
6. Exchange Type (8 bits): Indicates the type of exchange.
7. Flags (8 bits): Indicates specific options set for this ISAKMP exchange.
8. Message ID (32 bits): Unique ID for this message.
9. Length (32 bits): Length of total message (header plus all payloads) inoctets.

ion has a high false alarm rate.