$$a_i \equiv x^{\phi(n)\cdot\phi(n)} \pmod n \quad (\text{product } i=1 \text{ to } \phi(n),\ a_i \pmod n)$$

$$1 \equiv x^{\phi(n)} \pmod n.$$

CHINESE REMAINDER THEOREM:

The CRT is used to solve a set of congruent equations with one variable but different moduli, which are relatively Prime as shown below:

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\vdots$$
$$x \equiv a_K \pmod{m_K}$$

Step-1: The CRT states that the above equations have a unique solution. If the moduli are relatively prime.

Step-2: To find x, we have to follow.

find $m = m_1 * m_2 * \cdots * m_K$

find $m_1 = \frac{m}{m_1}$; $m_2 = \frac{m}{m_2}$ ---- $M_K = \frac{m}{m_K}$.

Find the multiplicative inverse of $m_1, m_2, \cdots m_K$ using the Corresponding moduli

$(m_1, m_2, \cdots m_k)$

call the inverse as $m_1^{-1}, m_2^{-1}, \cdots m_k^{-1}$.

Step-3: The solution to the simultaneous equation is.

$$x = \left[(a_1 * m_1 * m_1^{-1}) + (a_2 * m_2 * m_2^{-1}) + (+ \cdots + (a_m * m_K * m_K^{-1})\right] \pmod m$$

Eg: $x \equiv 2 \pmod 3$

$x \equiv 3 \pmod 5$

$x \equiv 2 \pmod 7$

Soln: $m = 3 * 5 * 7 = 105$

$m_1 = 105/3 = 35$; $m_2 = \frac{105}{5} = 21$; $m_3 = \frac{105}{7} = 15$.

m_1^{-1} . 35×2 . $1 = 2$

m_2^{-1} 21×1 $1 = 1$

m_3^{-1} $1 = 1/7$ $1 = 1$

(ii) $\phi(13) = 13 - 1 = 12$

The number of elements listed which are relatively prime to 'n' in a set of notations. is denoted with Z_n^*

Eg: find Z_4^*

$\{1, 3, 5, 9, 11, 13\}$

$\phi(4) = \phi(2 * 7)$

$= (\phi 2) * \phi(7)$

$= 1 * 6 = 6.$

Euler's theorem:

stmt: Euler's theorm state that for every 'a' and 'n' that are relatively prime then $a^{\phi(n)} \equiv 1 \pmod n$

Proof: The above stmt is true. if 'n' is the prime number, because $\phi(n) = n - 1$ & fermat's theorem holds.

Now, we have to prove the above statement also holds for any integer. Re-call $\phi(n)$ is the number of positive integers $< n$ that are relatively prime to n.

Consider the set of search integers as

$$R = \{a_1, a_2, \ldots, a_{\phi(n)}\}$$

Now, multiply each element by 'x', modulo 'n'

$$S = \{x a_1 \bmod n, x a_2 \bmod n, \ldots, x a_{\phi(n)} \bmod n\}$$

Name the above set as 'S'.

The set 'S' is a permutation of 'R', by the following reasons.

* Because 'a_i' is relatively prime to 'n' and 'x' is relatively prime to 'n' then 'xa_i' also relatively prime to 'n'.

There are no duplicates in 'S'.

∴ There fore,

$$a_1 * a_2 * \cdots * a_{\phi(n)} = x a_1 \bmod n * x a_2 \bmod n * \cdots * x a_{\phi(n)} \bmod n$$

$$\prod_{i=1}^{\phi(n)} a_i = \prod_{i=1}^{\phi(n)} x a_i \bmod n.$$

$$\left(\prod_{i=1}^{\phi(n)} a_i\right) \bmod n = \left(\prod_{i=1}^{\phi(n)} x a_i \bmod n\right) \bmod n$$

$$\left(\prod_{i=1}^{\phi(n)} a_i\right) \bmod n = \left(\prod_{i=1}^{\phi(n)} x a_i\right) \bmod n.$$

Fermat's Theorem:

Stmt: If 'p' is a prime number & 'a' is a +ve Integer, divisible by not then $a^{p-1} \equiv 1 \pmod{p}$

Proof: we know that if all the elements of Z_p are multiplicated by 'a' & module 'p' then results consist. The set of elements of Z_p in some order.

The "p-1" elements are $\{1, 2, \ldots, p-1\}$

multiply these number together.

$(a \bmod p) * (2a \bmod p) * \cdots * ((p-1)a \bmod p) \equiv [a * 2a * \cdots * (p-1)a] \pmod{p}$

Q: Find the result of $4^{18} \bmod 19$.

$4^{18} \bmod 19$

$4^{19-1} \bmod 19$

a p

$4^{19-1} \equiv 1 \pmod{19}$

$4^{19-1} \bmod 19 = 1 \bmod 19$

$4^{18} \bmod 19 = 1$

Euler's Totient function ($\phi(n)$):

Totient function is represented by $\phi(n)$.

This function is defined as the number of +ve integer less than 'n' & relatively prime to 'n'.

Note: $\phi(1) = 1$ $\phi(10) = 4$

$\phi(10) = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$= \{\underline{1}, 2, \underline{1}, 2, 5, 2\underline{1}, 2, \underline{1}\}$

$= 4.$

$\phi(n)$ properties:

* Euler's totient function (or) phi function having the following Properties.

i) $\phi(p) = p-1$ if 'p' is a prime number.

ii) $\phi(m*n) = \phi m * \phi n$ if 'm' & 'n' are relatively prime numbers

iii) $\phi(p^e) = p^e - p^{e-1}$, where 'p' is a Prime number.

iv. $\phi(1) = 1$

Eg: i) $\phi(2 \cdot 10) = 2^1 * 3 * 5$

by applying formula

$\phi(2^1) * \phi(3) * \phi(5) = [2^1 - 2^0] * 2 * 4$

$= 2 * 2 * 4 = 16$

Extended Eucledian Algorithm:

Purpose: To find multiplicative inverse of 'a' in 'Z_n' where $a \in Z_n$

Step-1: Take to variables r_1 & r_2 initialize $r_1 = n$, $r_2 = b$

Step-2: Take another 2 variables namely t_1 & t_2 as $t_1 = 0$, $t_2 = 1$

Step-3: until ($r_2 > 0$) (or while $r_2 > 0$)) do the following steps.

Step-4: Calculate $q = r_1 / r_2$

$r = r_1 - (q * r_2)$

Step-5: Swap r_1 & r_2 values with 'r'

$r_1 = r_2$

$r_2 = r$

Step-6: $t = t_1 - (q * t_2)$

$t_1 = t_2$

$t_2 = t$

Step-7: If ($r_1 == 1$) then $b^{-1} = t_1$

Step-8: Multiplicative inverse of 'b' is b^{-1}

$n = 26$ $b = 11$ $r_1 = 26$ $r_2 = 11$

$r_2 > 0$

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5
1	4	3	1	-2	5	-7
3	3	1	0	5	-7	26
	1	0		-7	26	

$t = t_1 - (q * t_2)$

$= 0 - (2 * 1)$

$= 0 - 2 = -2$

$t = 1 - (2 * (-2))$

$= 1 + 4 = 5$

$t = -2 - (1 * 5)$

$= -2 - 5 = -7$

$t = 5 - (3 * (-7))$

$= 5 + 21 = 26$

$b^{-1} = -7 + 26$ $b = 11$ $b^{-1} = 19$

$= 19$ $\Rightarrow 19 \times 11 = 209 \div 26$

$\Rightarrow 26 \overline{)209} (8$

208

(1)

Since $r_1 = 1$ multiplicative inverse of numbers.

$n = 5$ $b = 2$.

q	r_1	r_2	r	t_1	t_2	t
2	5	2	1	0	1	-2
2	2	1	0	1	-2	5
	1	0		-2	5	

$b^{-1} = -2$

$= -2 + 5$

$= 3$

Properties of Z_n:

$(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$

$(a-b) \bmod n = (a \bmod n - b \bmod n) \bmod n$

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

Eg: $a = 8 \quad b = 7 \quad n = 5$

i) $15 \bmod 5 = 0 \Rightarrow ((8 \bmod 5) + (7 \bmod 5)) \bmod 5 = (3+2) \bmod 5$

ii) $1 \bmod 5 = 1 \Rightarrow 1 \bmod 5 = 1 \qquad = 5 \bmod 5 = 0$

iii) $56 \bmod 5 = 1 \Rightarrow (3 * 2) \bmod 5 = 6 \bmod 5 = 1$

Note:

$10^x \bmod n = (10 \bmod n)^x$

Eg: $10^4 \bmod 5 = (10 \bmod 5)^4 = (0)^4 = 0$

Additive inverse:

In modular arithmetic if two number $a, b \in Z_n$ are said to be additive inverses of each other if $a+b$ is congruent ($\equiv$) 0 $(\bmod n)$

$a + b \equiv 0 \pmod n$

$\Rightarrow (a+b) \bmod n = 0 \bmod n$

$\Rightarrow (a+b) \bmod n = 0$

$n = 5, \quad Z_n = \{0, 1, 2, 3, 4\}$

$a = 0 \quad (a+b) \bmod 5$

$b = 1 \quad (0+1) \bmod 5$

$1 \bmod 5 = 1$

Multiplicative inverse:

In modular arithmetic $a, b \in Z_n$, are said to be multiplicative inverses of each other if $a * b \equiv 1 \pmod 1$

$\Rightarrow (a * b) \bmod n = 1 \bmod n$

$(a * b) \bmod n = 1$

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

$a = 2, \quad b = 3$

$(a * b) = 6 \bmod 5 = 1$

Problem

1) Find the multiplicative inverse of 8 in Z_{10}

$Z_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Extended Euclidean algorithm

GCD - 8 : 1, 2, 4, 8

10 : 1, 2, 5, 10 $\quad \therefore$ GCD : 2

* Now, we are going to discuss about some of the modulo arithmetic Properties.

Set of Residue (Z_n):

The result of modulo operation with modulus n, is always an integer b/w '0' & 'n-1'

Eg: $Z_6 = \{0, 1, 2, 3, 4, 5\}$

$Z_2 = \{0, 1\}$

$Z_n = \{0, 1, 2, \ldots n-1\}$.

Congruence:

In cryptography we often used the concept of congruence in place of equality that congruence operator is '($\equiv$)'

Eg: 2 mod 10 = 2 and 22 mod 10 = 2

then $2 \equiv 22 \pmod{10}$.

Note:

If $a \equiv b \pmod{n}$ then we can expand as a mod n = b mod n & viceversa.

Residue class:

A residue class [a] (or) $[a]_n$ is a set of integers congruent to modulo n. If 'n' is the module value then we can find n-1 residue classes.

Eg: modulus $5 = Z_5 = \{0, 1, 2, 3, 4\}$.

$[0] = \{\ldots, -15, -10, -5, 0, 5, 10, 15, 20, \ldots\}$

$[1] = \{\ldots, -14, -9, -4, 1, 6, 11, 16, 21, \ldots\}$

$[2] = \{\ldots, -13, -8, -3, 2, 7, 12, 17, 22, \ldots\}$

$[3] = \{\ldots, -12, -7, -2, 3, 8, 13, 18, 23, \ldots\}$

$[4] = \{\ldots, -9, -4, -1, 4, 9, 14, 19, 24, \ldots\}$

Operations of Z_n:

i) addition - $(a+b) \bmod n = c$

ii) Subtraction - $(a-b) \bmod n = c$

iii) multiplication - $(a*b) \bmod n = c$

Here a, b > 0 (any +ve integers) and

$c \in Z_n = \{0, 1, 2, \ldots, n-1\}$

Eg: a = 8, b = 4, n = 5, $Z_5 = \{0, 1, 2, 3, 4\}$

(a+b) mod n (8+4) mod 5 12 mod 5 in Z_5

(a-b) mod n (8-4) mod 5 4 mod 5 in Z_5

* prime number play a critic role in number theorey, in particular many cryptography techniques.

* Any integer ($a>1$) can be factorized in a unique way as.

$a = P_1^{a_1} * P_2^{a_2} * \cdots P_n^{a_n}$, where $P_1, P_2, P_3, \cdots P_n$ are Prime numbers and 'a_i' are +ve integers.

* A prime number can be represented as set of all prime numbers any +ve integer 'a' can be return uniquely in the following form

$$a = \prod_{p \in P} a^{a_p} \text{ where } a_p > 0$$

$$13 \mid 11011 \quad 7 \mid 847 \quad 11 \mid 121 \quad 11 \mid 11 \quad 1 \qquad = 11^2 \times 13 \times 7$$

Note:

* The integer '12' is represented as

$12 = \{a_2 = 2, a_3 = 1\} = 2^2 \times 3^1 = 4 \times 3 = 12.$

* The integer '91' is represented as

$91 = \{a_7 = 1, a_{13} = 1\}$

$= 7^1 \times 13^1$

$= 91.$

* $11011 = \{a_7 = 1, a_{13} = 1, a_{11} = 2\}$

$= 7^1 \times 13 \times 11^2$

$= 11011$

GCD (greatest Common Divisor):

The GCD of 60, 24

$$24 \mid 60 \mid 2 \quad (48) \quad 12 \mid 24 \mid 2 \quad (24) \quad (0)$$

GCD = 12.

Relatively prime numbers:

If a, b are 2 +ve integers numbers whose GCD is 1. Those numbers are called relatively prime numbers.

Eg: 8 = 1, 2, 4, 8

15 = 1, 3, 5, 15

∴ GCD = 1

8, 15 are relatively prime number because both of term having a common factor which is 1.

Modular Arithmatic:

* The modular arithmatic concept mainly depends on modulo operation, which will produce remainder.

* In modular arithmatic the remainder is also called as

residue

5) $C \rightarrow V$: $Ticket_V \parallel authenticalor_C$

6) $V \rightarrow C$: $E_{K_{C,V}}[TS_{5+1}]$ (mulual authentication)

$Ticket_V = E_{K_V}[K_{C,V} \parallel ID_C \parallel AD_C \parallel ID_V \parallel TS_4 \parallel lifetime\ 4$

$Authenticalor = E_{K_{C,V}}[ID_C \parallel AD_C \parallel TS_5$

Number Theory

Divisibility rule: A number which is divisible by another number.

$a = (q * b) + remainder$

Eg: 23 divided by 5

Quotient = 4 $\quad\therefore\quad 23 = (4*5)+3$

remainder = 3 $\quad\quad = 20 + 3$

$23 = 23$

Divisibility rule:

Let a, b are two integers (a>b). If a is divided by b, we can write the divisibility rule as $a = (b*q)+R$, where

q is the Quotient and R is the remainder.

Properties of divisibility:

* If 'a' divides (/) 1, then $a = \pm 1$
* If a/b & b/a, then $a = \pm b$
* Any $b \neq 0$ divides '0'.
* If 'a' divides 'b' & 'b' divides 'c' then 'a' divides 'c'.
* If 'b' divides 'g' & b/h then $b/(mg+nh)$, where m,n are arbitory integers.

$b/g \Rightarrow g = bm_1 \rightarrow (1)$

$b/h \Rightarrow h = bn_1 \rightarrow (2)$

$b/(mg+nh) \Rightarrow mg+nh = m(bm_1)+n(bn_1)$

$\Rightarrow \frac{b(mm_1+nn_1)}{b} = mm_1+nn_1$

$-11 = (-2)*7+3$ remainder = 3 Quotient = -2

$-23 = (-4)*6+1$ R = 1, Q = -4.

Prime number:

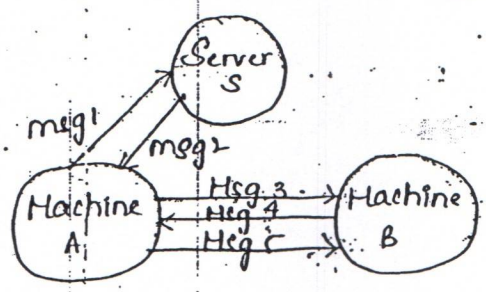
* A number which is divisible by 1 & itself is called as prime number

* An integer (p>1) is a prime number if and only if its only divi

Network Security

Authentication protocol

Kerberos: * Derived by MIT institute.
* based on Needham-Schroeder — Symmetric key, asymmetric key



Kerberos Version 4:

1) $C \longrightarrow As : ID_c \| P_c \| ID_V$
2) $AS \rightarrow C$: Ticket
3) $C \rightarrow V : ID_c \| Ticket$

$Ticket = E_{K_V}[ID_c \| AD_c \| ID_V]$

Here C = client
As = Authentication Server
V = Server
ID_c = Identifier of user on 'c'
ID_V = Identifier of V
P_c = Password of user on c
AD_c = Network address of c
K_V = Secret encryption key shared by As & V

Kerberos Version 6:

1) $C \rightarrow As : ID_c \| ID_{tgs} \| TS_1$
2) $As \rightarrow C : E_{K_c}(K_{c,tgs} \| ID_{tgs} \| TS_2 \| life\ time_2 \| Ticket_{tgs})$
3) client $\rightarrow$ TGS : $ID_V \| Ticket_{tgs} \| authenticator_c$
4) $TGS \rightarrow C : E_{K_{c,tgs}}[K_{c,V} \| ID_V \| TS_4 \| Ticket_V]$

$Ticket_{TGS} = E_{K_{tgs}}[K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| life\ time_2]$

$Ticket_V = E_{K_V}[K_{c,V} \| ID_c \| AD_c \| ID_V \| TS_4 \| life\ time_4]$

$Authenticator_c = E_{K_{c,tgs}}[ID_c \| AD_c \| TS_3]$

Verification:

$w = (s')^{-1} \bmod q$

$u_1 = [(H(H'))w] \bmod q$

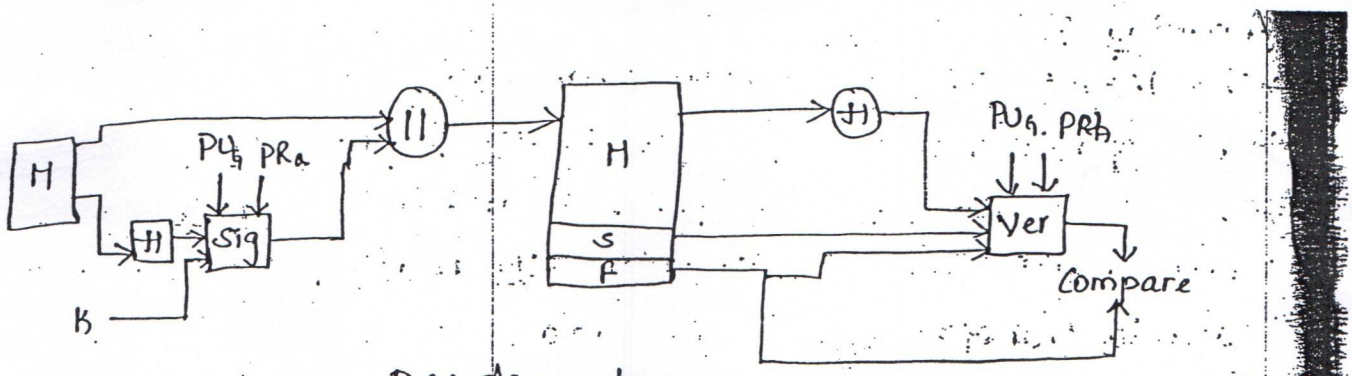
$u_2 = (r')w \bmod q$

$v = [(g^{u_1} \cdot y^{u_2}) \bmod p] \bmod q$

Test:

$v = r'$

CIDR: class Inter Domain routing



DSS Approach

Global public key Components:

* Take 'p' as prime number, where $2^{k-1} < p < 2^k$
for $512 \leq k \leq 1024$ and k is a multiple of '64'
i.e; the bit length is between 512 and 1024 bits in increments of 64 bits.
* Take 'q' as divisor of (p-1) of bit length 160 bits
* $g = h^{(p-1)/q} \bmod p$ where h is an integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p = 1$

Signing:

$$r = (g^k \bmod p) \bmod q$$

$$s = [k^{-1}(H(M)) + xr] \bmod q$$

Signature = (r, s) Concatenate

User's private key:

'x' a random integer with $0 < x < q$

User's public key:

$$y = g^x \bmod p$$

K → user's per message secret key.

K - a random integer with $0 < K < q$

Advantage :-

→ The distant coverage is more than 802.11
→ This operate with the speed of 66 GHz frequency range.
→ 802.16 is useful for both personal and business purpose..
→ Due to this wireless LANs only GSM mobile phones which are used to transfer multimedia data.
→ This is the main cause for third Generation (3G) mobile network revolution.

Bluetooth :-

Bluetooth is an invention of SIG (Special Intrest Group) which is used to connect communication devices using short Range. The motivater to invent this wireless technology is - L.M. Ericsson. He was supported by "Hewlett packard" (H.P) computer peripherals without wire. Eventhough IEEE approved the first personal area network (PAN) standard. In 2002 the bluetooth SIG is still active busy with the improvements. The wireless Lan bluetooth technology finally approved in 2006 with a single standards.

Bluetooth Architecture :- The basic unit of bluetooth system is called piconet. This contains 7 slave node within a distance of 10 m. An interconnected piconets is called Scatternet. The main device which transfers data is called master. The master node is centralised in the piconet, it can transfer the data by establishing interconnection b/w other piconet of slave (receives).

Applications of bluetooth :-

Bluetooth is supported by cordless telephony, synchronization, general object exchange (client-Server technology), fax machine, headset etc.

W-LAN's :-

The no. of mobile computing and communication devices grow so that the demand to connect them to the outside world. To acheive this type of mobility and networks communication there are so many protocols have been evolved. Some of them are infrared, bluetooth etc. These networks are numbered by IEEE, as 802.11.

The 802.11 technology maintains a protocol stack. This stack contains all the wireless protocols like Infrared FHSS, DSSS

The physical layer of wireless LAN contains FHSS (frequency Hopping Spread Spectrum) which uses 79 channels at the speed of 2.4 GHz. whereas Infrared uses only one channel at 1 mbps to 2mbps speed. The another useful protocol is DSSS (Direct Sequence Spread Spectrum) is having 11 channels with the speed 2 Mbps.

broad band wireless :-

The broad band wireless is used at high speed internet LAN's which are erecting a big antenna at outside. Those customers using these wireless Lans should install at their roofs to get the signals in an easiest way.

So many people from industry realized that the broadband wireless standard was the key element to develop their business They establish air interface for fixed broadband wireless access System. This is also called as wireless local loop (or) wireless MAN. They have given a number 802.16 as per the instructions of IEEE

Say from 10mbps persecond to 100 mbps fortunately switched ethernet containing high speed backs plane and 32 plug in cards with 10 BaseT Twisted pair connectors to a single host computer.

2) Fast ethernet :- To pump up the speed various industry groups proposed fast ethernet as optional LANs the fast ethernet sometimes called as FDDI (Fibre Distributed Data Interface) and uses fibre channel. In three cases the station management was too complicated, which needs complex chips and high prices.

The primary reasons to use Ethernet are :-

1) Backward Compatibility with existing ethernet LAN's.

2) The new protocol other than ethernet may lead to unforeseen problems.

Giga bit Ethernet :-

It was quickly ratified by IEEE in 1998 and named as 802.3z It is 10 times faster than fast ethernet and also compatible with the latest internet technologies to use gigabit ethernet we should use direct network connection between two computers (or) in multiconnected networking environment a device called Hub (or) Switch needed.

The 802.32 Considered two features to exploit the giga bit ethernet features :-

1) Carrier extension
2) Frame Bursting
3)

The Gigabit ethernet uses 5000 mts distance fiber optic cable its highest data transmission ethernet cable (1000 base Lx).

Computer Networks UNIT-II

Ethernet :-

After takening another channel allocation solution specifically in LAN and MAN networks is introduced as Ethernet. This is derived by IEEE standards. people and named it as 802.3.

All the ethernet protocols are identical accept in the usage of geographical distance and transmission speed.

Ethernet Cabling :- There are four types of cables are commonly used for ethernet. They are :-

Name	Cable	Max. segmentation	No. of nodes
1) 10 Base 5	Thick Coax	500m	100
2) 10 Base 2	Thin Coax	185m (no hub needed)	30
3) 10 Base T	Twisted pair	100m	1024 (no hub needed)
4) 10 Base F	Fiber optics	2000m	1024

In the above cables the ethernet prefers to use 10 Base F which is expensive but it has excellent noise immunity it also runs of upto kilometers and also provides good security.

Ethernet Performance :-

Ethernet is an excellent channel allocation system under the heavy conditions with constant load, It uses backoff algorithm which is ap complicated, It makes the data mostly collision free and also improves channel efficiency.

It is proved that the ethernet is probably worthful and also carry large amounts of data through the networks. the average no. of frames in each minute will vary (change) to the next minute, due to that discovery network traffic is difficult. This also makes the ethernet performance as useful.

Types of Ethernet :-

1) Switched ethernet : If more stations are added to an ethernet the traffic will grow up. one copy out is to a higher speed

H [1, 2, ---- L] → Blocks of padded and appended message

F [0, B, C, D) to F (79, B, C, D) → 80 processing functions.

K [0], K [1], ---- K [79] → 80 processing constants.

H_0, H_1, H_2, H_4 → 5 initialized buffers.

Step 1: Pseudo code:

(<<< → Circular shift)

```
for loop on k = 1 to L
    [w(0), w(1), --- w(15)] = H [K]
    for t = 16 to 79 do
        w(t) = [w(t-3) XOR w(t-8) XOR w(t-14) XOR w(t-16)]
                <<<
    A = H0, B = H1, C = H2, D = H3, E = H4
    for t = 0 to 79 do
    temp = A<<<5 + f (A, B, C, D) + E + w(t) + k(t); E = D; D = C;
                C = B<<<30; B = A; A = TEMP;
    END for loop
    H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D; H4 = H4 + E
END for loop.
```

Step-3 : processing functions :

→ SHA-1 requires 80 processing functions defined as

1) $f(t,B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$ $(0 <= t <= 19)$

2) $f(t,B,C,D) = B \oplus C \oplus D$ $(20 <= t <= 39)$

3) $f(t,B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$ $(40 <= t <= 59)$

4) $f(t,B,C,D) = B \oplus C \oplus D$ $(60 <= t <= 79)$

Step-4 : Processing Conitants :

→ SHA-1 requires 80 processing conitant words defined as follows.

$K(t) = 0X5A827999$ $(0 <= t <= 19)$

$k(t) = 0XBED9EBA1$ $(20 <= t <= 39)$

$K(t) = 0X8F1BBCDC$ $(40 <= t <= 59)$

$K(t) = 0XCA62C1D6$ $(60 <= t <= 79)$

Step-5 : Initializing Buffers :

→ SHA-1 requires 160 bits of 5 buffers each with 32 bits.

$H_0 = 0X67452301$

$H1 = 0XEFCDAB89$

$H2 = 0X98BADCFE$ [0x represents Hexadecimal number]

$H3 = 0X10325476$

$H4 = 0XC3D2E1F0$ [0_o → octal]

[0_b → Binary]

Step-6 : Processing Messages

→ After making the message as 512 multiple, in this step we divide the entire message into 512 bit blocks. This is the main task of SHA-1 algorithm which loops through the padded and appended message in 512 bit blocks.

III unit

SHA 1 (Secure Hasing Algorithm):

→ This algorithm is an enhancement for message digest-5 algorithm. This is also developed by RON Rivest.

→ It generates 160 bit unique Hash code, which is 32-bits extra of MD5 output.

→ This 160 bit Hash code will give more security than MD-5 algorithm when compared.

→ The process also very similar to MD-5 algorithm with respect to no. of steps.

→ The following are the steps following by SHA-1 algorithm.

(Note:
Different types of Message digest algorithm

1) MD (MD_4 and MD_5)
2) SHA-1
3) RTPEMD (160)
4) Whirlpool
5) Tiger
6) Gost (3411)
7) SHA-3 [Best algorithm to secure Hash code])

Step-1: Padding:

→ The input message / pain text is padded with a1 and many zeros as necessary to bring the message length to 64 bits fewer than an multiplier of 512.

Step-2: Appending length:

→ 61-bits are appended to the end of padded message. These bits hold the binary-format from 2^{61} combinations of a binary number. This makes the Original message length to multiplier of 512.

The SET process:

1) The customer opens an account:

→ The customer opens a credit card account (such as master card/visa) with a bank (issuer) that supports electronic payment mechanisms & the SET protocol.

2) The customer receives a certificate:

→ After the customer's identity is verified (with the help of details such as passport, business documents etc), the customer receives a digital certificate from a CA.

3) The merchant receives a certificate:

→ A merchant that wants to accept a certain brand of credit cards must possess a digital certificate.

4) The customer places an Order:

→ In this task the customer browses the list of items available searches for specific items, select one (or) more of them & places the order.

→ The merchant sends back details such as the list of items selected, their quantities, prices, total bill etc back to the customer for his record, with the help of an order form.

5) The merchant is verified:

→ The merchant also sends its digital certificate to the customer.

→ This assures the customer that he is dealing with a valid merchant.

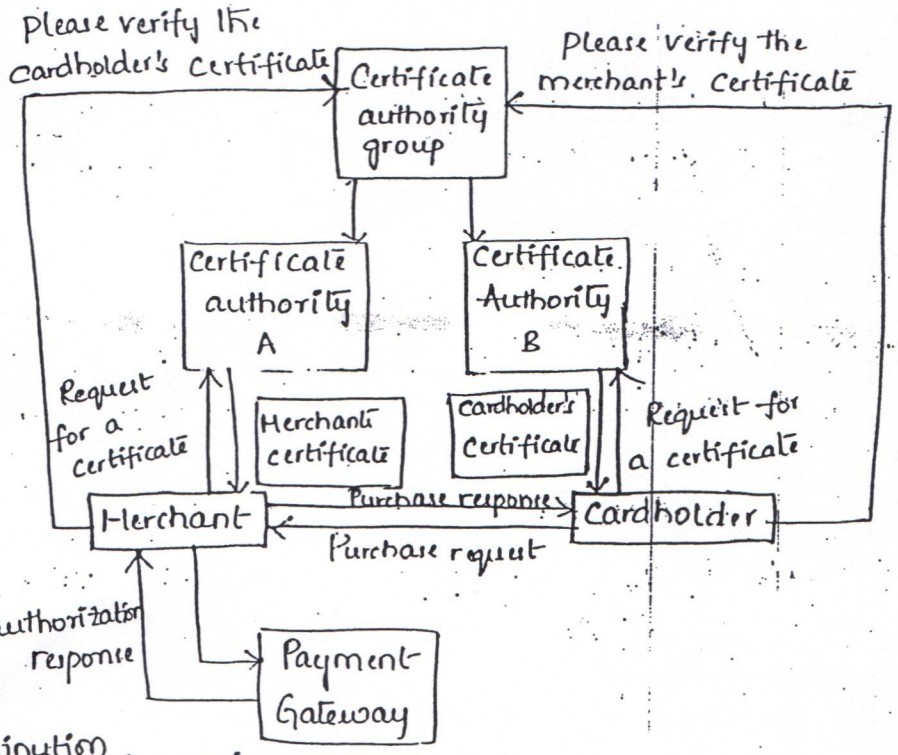
6) The order & payment details are sent:

→ The customer sends both the order & payment details to the merchant along with the customer's digital certificate.

→ The order confirms the purchase transaction with reference to the items mentioned in the order form.

→ The payment contains credit card details.

→ The payment information is encrypted that the merchant cannot read it. → Continued before -



→ continuation of acquirer:

→ Instead an acquirer provides the merchant an assurance (with the help of the issuer) that a particular cardholder account is activate and that the purchase content doesn't exceed the credit limit etc.

→ The acquirer also provides electric funds transfer to the merchant account.

Payment Gateway:

→ This is a task can be taken up by the acquirer.

→ The payment gateway acts as an interface b/w SET and the existing card payment networks for payment authorizations.

→ The payment gateway processes the payment message on behalf of the merchant.

→ The merchant exchanges SET message with the payment gateway over the Internet.

→ The payment gateway connects to the acquirer's system using a dedicated network line.

Certification Authority (CA):

→ This is an authority that is trusted to provide publickey certificates to cardholders, merchants & payment gateway.

→ CA's are very important to the success of SET.

The merchant requests payment authorization:

→ The merchant forwards the payment details sent by the customer to the payment gateway via the acquirer and requests the Payment gateway to authorize the payment.
(i.e; ensures that the credit card is valid and the credit limits are not exceeded)

The payment gateway authorizes the payment:

→ using the credit card information received from the merchant, the Payment gateway verifies the details of the customer's credit card with the help of the issuer & either authorizes (or) rejects the payment.

The merchant confirms the order:

→ The payment gateway authorizes the payment, the merchant sends a confirmation of the order to the customer.

The merchant provides goods/services:

→ The merchant now transfers the goods/services as per the customer's order.

The merchant requests payment:

→ The payment gateway receives a request from the merchant for making the payment.

→ The payment gateway interacts with the various financial institutions such as the issuer, acquire to effect the payment from the customer's account to the merchant's account.

→ The second byte specifies the actual error.

Severity	cause
Byte 1	Byte 2

Secure Electronic Transaction (SET):

→ The SET is an open encryption & security specification that is designed for protecting credit card transactions on the Internet.

SET Participants:

Card holder:

→ using the Internet, Consumers and corporate purchasers interact with merchants for buying goods and services.

→ A Card holder is an authorized holder of a payment card such as master card / visa that has been issued by an issuer.

Merchant:

→ A merchant is a person / an organisation that wants to sell goods (or) services to cardholders.

→ A merchant must have a relationship with an acquirer for accepting payments on the Internet.

Issuer:

→ The issuer is a financial institution (such as bank) that provide a payment card to cardholder.

→ The most critical point is that the issuer is the ultimately responsible for the payment of the cardholder's debt.

Acquirer:

→ This is a financial institution that has a relationship with merchant for processing card authorizations & payments.

→ The reason for having acquirers is that merchants accept credit cards of more than one brand, but are not intrested in dealing with so many bank card organizations / issuers.

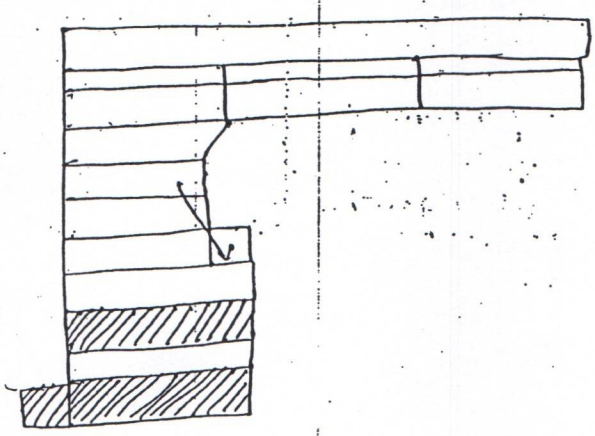
→ The customer's certificate assures the merchant about the customer's identity

→ This protocol provides 2 services to an SSL connection.

i) Confidentiality: This is achieved by the using the secret key that is defined by the handshake protocol.

ii) Integrity: The handshake protocol also defines a shared secret key (HAC that is used for assuring the message integrity.

→ The operation of the record protocol is shown in the following figuare:



→ The SSL record protocol takes an application message as input.

→ first, it fragments it into smaller blocks, optionally compresses each block, adds MAC, encrypts it, adds a header and gives it to the transport layer.

The alert protocols

→ when either the client or the server detects an error, the detecting party sends an alert message to the other party.

→ If the error is fatal, both the parties immediately close the SSL connection

→ Both the parties also destroy the session identifiers, secrets and keys associated with this connection before it is terminated

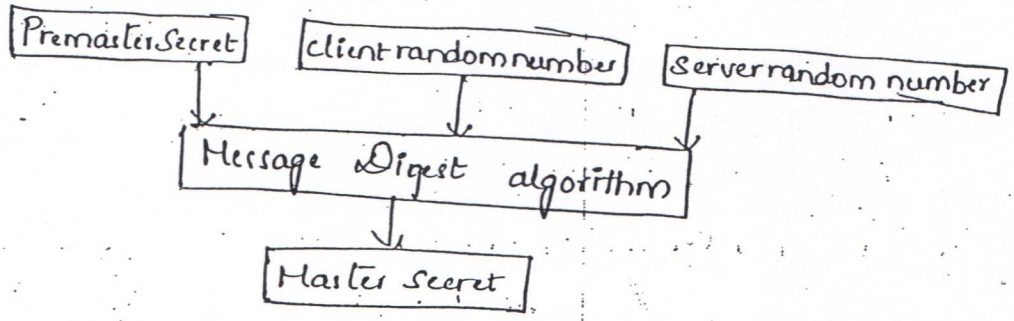
→ Other errors which are not so severe, the parties handles the errors without terminating the connection.

→ Each alert message consists of 2 bytes.

→ the first byte signifies the type of error.

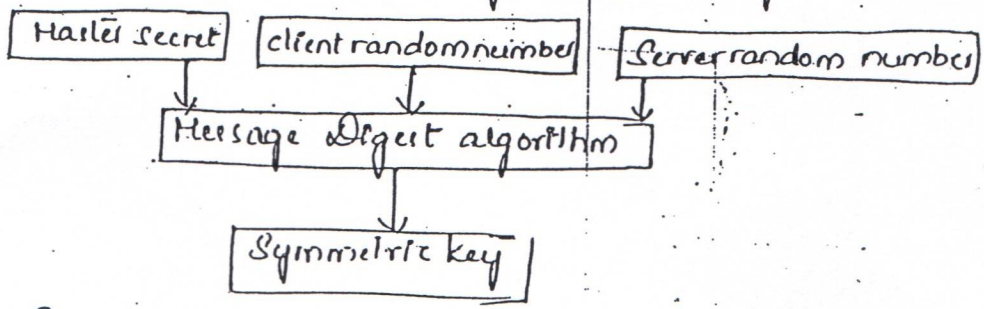
→ if it is a warning, this byte contains 1

→ if the error is fatal this byte contains 2



→ This master secret is 48-byte value and is used to generate keys and secrets for encryption & HAC computations.

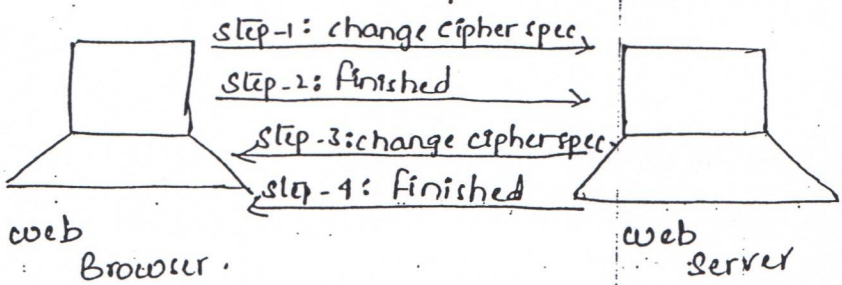
→ The shared secret keys are generated using this master secret.



Phase-4: finish:

→ The client initiates this fourth phase of the SSL handshake, which file Server ends.

→ This phase contains 4 steps.



The first step is a confirmation from the client that all is well from its end, which it strengthens with the finished message. The Server sends identical message to the client.

Record protocol:

This protocol comes into picture after a successful handshake completed b/w the client & Server.

After the client & the Server have authenticated each other & decided what algorithms to use for secure information exchange

Server hello done:

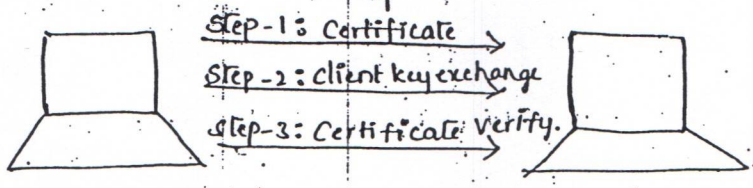
→ This message indicates to the client that the Server's portion of the hello message is complete.

phase-3: client authentication & key exchange:

→ The client initiates this third phase and all the messages that are sent by the client only.

→ Here, the Server system only receives the messages.

→ This phase contains 3 steps:



Certificate:

→ This step is performed only if the Sender demands for the client's Certificate.

→ If the client doesn't have Certificate then it sends a no Certificate message, instead of a Certificate message.

→ It is then up to server do decide if it wants to still continue (or) not.

Client key exchange:

→ This step allows the client to send information to the server.

→ This information is related to the Symmetric key that both the parties will use in the session.

→ Here, the client creates, 48 byte pre-master Secret and encrypts with the Server's public key & sends to the Server.

Certificate Verify:

→ This step is necessary only if the Server had demanded the client's Certificate.

→ For this purpose, in this optional step, the client combines the pre-master Secret with the random numbers exchanged by the client & the Server earlier after having them together using MD5 (or) SHA-1 and signs the result with it, private key.

Session id: If the session id value sent by the client was non-zero, the server uses the same value otherwise, the server creates a new session id and puts it in this field.

Cipher : Contains a single cipher, which the server selects from the list sent earlier by the client.

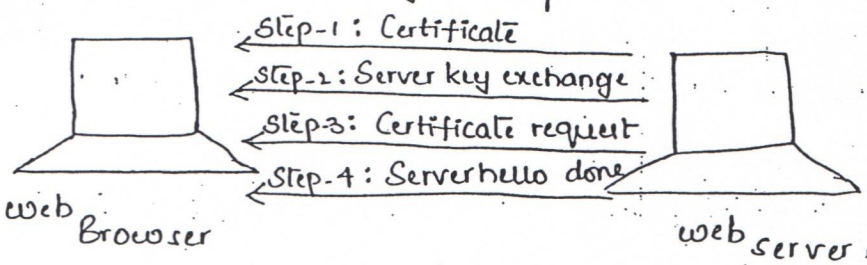
Compression method : contains a compression algorithm, which the server selects from the list sent earlier by the client.

phase-2: Server Authentication & key Exchange:

→ The server initiates this second phase and all the messages that are sent by the server only.

→ Here, the client system only receives the messages.

→ This phase contains four steps:



Certificate:

→ The server sends its digital certificate and this will help the client to authenticate the server using the server's public key from the server's certificate.

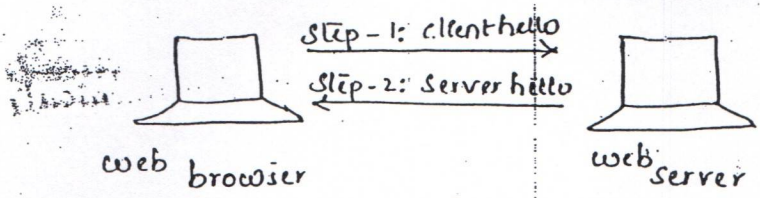
Server key Exchange:

→ It is optional. It is used only if the server does not send its digital certificate to the client.

→ The server sends its public key to the client if the certificate is not available.

Certificate request:

→ The server can request the client's digital certificate and the server may not always expect the client to be authenticated therefore this is optional.



→ The process starts with a client hello message from the client to the server

→ It consists of the following parameters.

Version: This field identifies the highest version of SSL that the client can support.

Random number: It contains 2 sub-fields:

i, A 32-bit data time field that identifies the current system date and time on the client computer.

ii, A 28-byte random number generated by the random no generator software built inside the client computer.

Session id: i, If the field contains a '0' value then the client wants to create a new connection with the server.

ii, If this field contains a non-zero value then there is already a connection b/w the client & the server.

Cipher: This list contains a list of the cryptographic algorithms supported by the client (eg: RSA, Diffie-Hellman etc)

Compression method: This field contains a list of the compression algorithms supported by the client.

→ The client sends the client hello message to the server and waits for server's response.

→ The server sends back a server hello message to the client

→ The server hello message consists of the following fields:

Version: The field identifies the lower of the version suggested by the client and the highest supported by the server.

Random number: This field has the same structure as the random number field in the client.

Handshake protocols

→ This protocol is used by the client & the server to communicate using an SSL enabled connection.

→ The handshake protocol consists of a series of messages b/w the client and the server.

→ Each of the message has the following format.

Type	length	content
1 byte	3 bytes	3 or more bytes.

Type: This field indicates one of the ten possible message types.

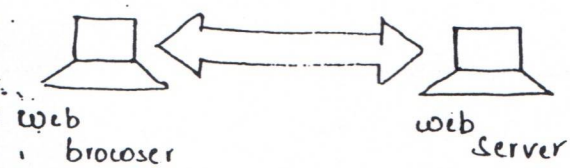
Length: This field indicates the length of the message in bytes.

Content: This field contains the parameters associated with this message, depending on the message type.

Message Type	Parameter
1. Hello request	None
2. Client hello	version, RNo, Session id, Cipher, Compression method
3. Server hello	version, RNo, session id, Cipher, Compression metho
4. Certificate	X.509 v3 certificate

→ The handshake protocol is actually made up of four phases. These phases are:

1. Establish Security capabilities
2. Server authentication & key exchange
3. Client authentication & key exchange
4. Finish.



Phase 1: Establish Security capabilities:

This is used to initiate a logical connection and establish the security capabilities associated with that connection.

→ This consists of 2 messages i) client hello

Web Security

Secure Socket Layer (SSL):

→ The ssl protocol is an Internet protocol is used for secure exchange of information b/w a web browser and a web server.

→ It provides two-basic security services: authentication & confidentiality

→ logically, it provides a secure pipe between the webbrowser and the webserver.

→ SSL major versions are 2, 3 and 3.1.

The position of the SSL in TCP/IP protocol:

→ This ssl is placed in b/w application layer and Transport layers of TCP/IP protocol.

→ At the Sender side the application layer transmits the information to the ssl layer, here it will do encryption and adds its own header to the encrypted data and that is transfered to transport layer.

→ Now the process is same as in normal TCP/IP

→ At the receiver side the transport layer transmits the data to the ssl layer, her it will do some decryption process and it passes the Original information to the receiver side system application layer

Application layer
Secure Socket layer
Transport layer
Network layer
Data link layer
Physical layer

How ssl works:

→ SSL has 3 sub-protocols namely (i), Handshake protocol
(ii), Record protocol
(iii), Alert protocol.

Next payload → This 8-bit field indicates the type of the first Payload of the message.

Major Version → This 4-bit field identifies the major ISAKMP Protocol version as used in the current exchange.

Exchange Type → This 8-bit field indicates the type of exchange.

Flags → This 8-bit field indicates the specific set of options for this ISAKMP exchange.

Message ID → This 32-bit field identifies the unique id for this message.

Length → This 32-bit field specifies the total length of the messag including the header and the payloads in octets.

Payload types:

→ ISAKMP Specifies different payload types.

Eg: SA payload – It is used to start establishment of an SA.

Proposal payload – It contains the information used in during the SA establishment.

Key exchange Payload – for exchanging keys using mechanisms such as Oakley, Diffie-Hellman, RSA etc.

Exchange Types:

There are 5 exchange types defined in ISAKMP.

1. Base exchange → The transmission of the key and authentication material.
2. Identity protection exchange → It is used to protect the identities of the user.
3. Authentication only exchange → It is used to perform mutual authentication.
4. The aggressive exchange → It is used to minimize the no. of exchanges because of hiding the user's identities.
5. The information exchange → It is used for one way transmission of information for SA management.

Message-1:

→ X sends a cookie and the keys generated by Diffie-Hellman and signs this block with its private key.

Message-2:

→ when Y receives message 1, it verifies the signature of X using the public key of X.

→ when Y is satisfied that the message indeed came from X, it prepares an acknowledgement message for X, containing a cookie sent by X.

→ Y also prepares its own cookie and Diffie-Hellman public key and along with some other information, it signs the Package with its private key.

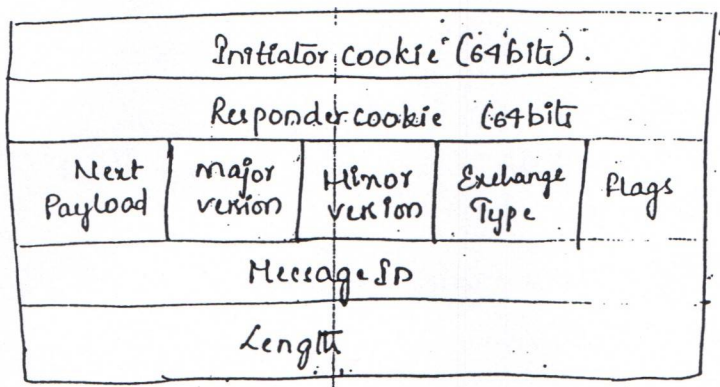
Message-3:

→ upon receipt of message 2, X verifies it using the public key of Y, when X is satisfied about it, it sends a message back to Y to inform that it has received Y's public key.

ISAKMP:

→ The entire block is encapsulated inside a transport segment.

→ The format of this protocol is shown in the following figure.



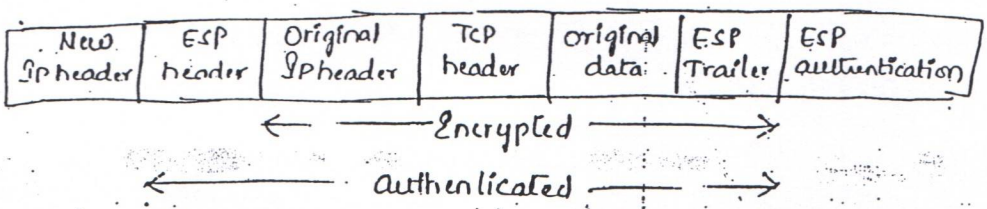
Fields — Description

Initiator cookie → This is a 64-bit field containing the cookie of the entity that initiates the SA establishment or deletion.

Responder cookie → This is also a 64 bit field containing the cookie of the responding entity (initially it is 0)

Esp Tunnel mode:

→ In the ESP tunnel mode, the entire original IP packet is encryp and is inserted b/w the new IP header and original IP header.



IP sec Key Management:

→ with out a proper key management setup, IP sec can't exist.
→ This key management in IPsec consists of two aspects:
  (i) key agreement
  (ii) key distribution
→ we require four keys if we want to make use of both AH and ESP two keys for AH (one for transmission and one for receiving) and two keys for ESP.
→ The protocol used in IPsec for key management is called as IS AKMP / oakley.
→ The Internet security association and key management protocol defines the procedures for establishing, modifying and deleting SA's
→ oakley is based on the Diffie-Hellman key exchange protocol, with a few variations.

Oakley key determination protocol:

→ This protocol is used to retain the advantages of Diffie-Hellman and to remove the drawbacks.
→ Oakley supports three authentication mechanism.
  (i) Digital Signature (MD, Encryption with Sender's private key)
  (ii) Public key Encryption (Encrypting the data such as Sender's user id with the receiver's public key)
  (iii) Secret-key Encryption (A key derived By using some Outer mechanism
→ The oakley protocol provides for a number of message types.
  Eg: Aggressive key exchange.
→ It consists of 3 messages exchanges b/w 2 parties by which

Field	Description
Payload data	This variable length field contains the transport layer segment (transport mode) or IP Packet (tunnel mode), which is protected by encryption.
Padding	This field contains the padding bits, if any, that are used by the encryption algorithm.
Padding-length	This 8-bit field identifies the no. of padding bytes in the immediately preceeding field.
Next Header	This 8-bit field identifies the type of encapsulated data in the payload.
Authentication data	This variable length field contains the authentication data called as ICV per datagram.

Modes of Operation:

Esp Transport mode:

→ Transport mode Esp is used to encrypt the IP data and optionally authenticated.

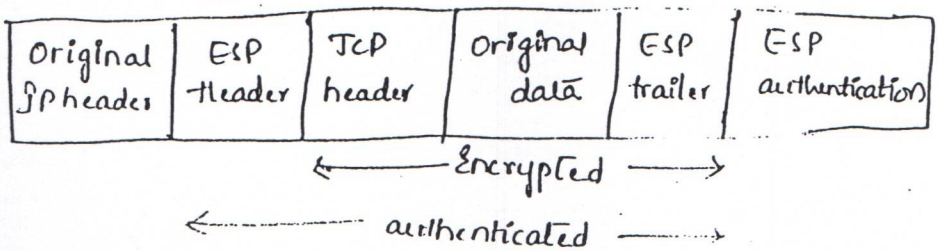
→ Here the Esp is inserted into the IP Packet immediately before the transport layer header and an ESP trailer (containing the fields padding, Padding length and next header) is added after the IP packet.

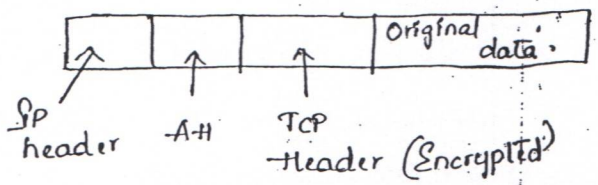
→ If authentication is also used the Esp authentication data field is added after the Esp trailer.

→ The entire transport layer segment and the ESP trailer are encrypted.

→ The entire cipher text, along with the ESP header is authenticated

→ These steps are shown in the following figure.



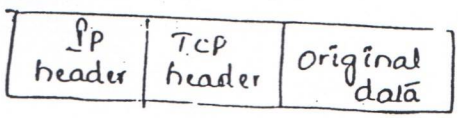


(b) After applying AH

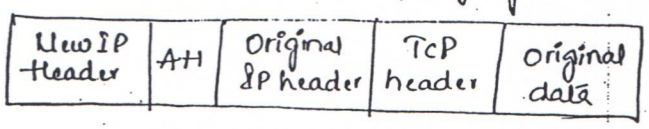
AH Tunnel mode:

→ In the tunnel mode, the entire original IP packet is authenticated and the AH is inserted b/w the original IP header and new outer IP header.

→ The Inner IP header contains the original source and destination IP addresses, where as the Outer IP header contains different IP addresses (Eg: IP addresses of the firewalls (or) other security gateways)



(a) Before applying AH.



(b) After applying AH.

Encapsulating Security Payload: (ESP)

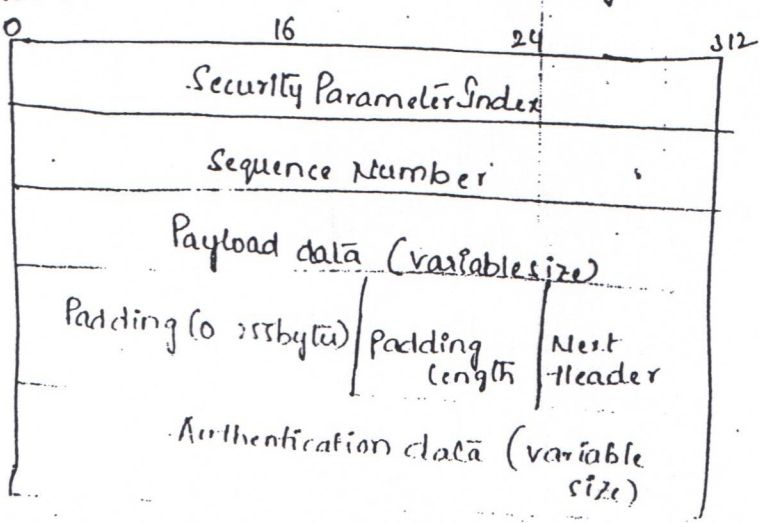
Esp format:

→ Esp protocol provides confidentiality and integrity of messages

→ Esp is based on symmetric key cryptography techniques.

→ Esp can be used in tunnel (or) it can be combined with AH.

→ The ESP Packet contains four fixed length fields and three variable length fields.



Anti replay Service:

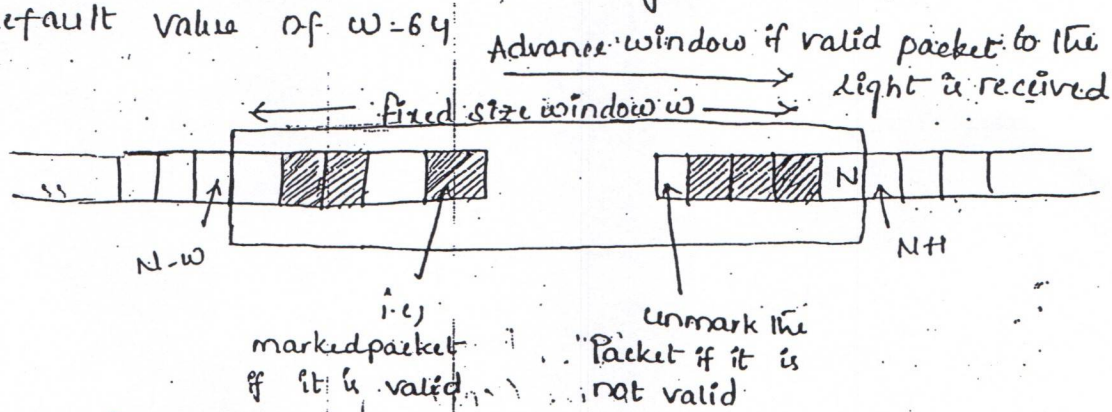
→ when a new SA is intialized, the Sender intializes a Sequence number counter to 0.

→ Each time that a packet is sent on this SA, the Sender increments the counter by 1 and places the value in Sequence number field.

→ If it exceed $2^{32}-1$, then the Sender should terminate this SA and establish a new SA with a new key.

→ At the receiver side the processing is different.

→ The receiver maintains a sliding window of size w, with the default value of w=64



w → Specifies the size of the window

N → Specifies the maximum highest Sequence number so far received for a valid packet.

N is always at the right edge of the window.

→ If the received packet falls with in the window and is new, the MAC is checked. If the packet is authenticated the window is advanced so that this Sequence number is the right edge of the window, and the Corresponding slot in the window is marked.

→ If the received packet is to the left of the window or if authentication fails, the packet is discarded and this is an audiable event.

AH modu of Operation:

AH Transport mode:

In the transport mode, the position of the AH is b/w the original IP header and the Original TCP header of the IP packet.

New IP header	TCP Header	original data

(a) Before applying AH

Authetication Header (AH):

AH format:

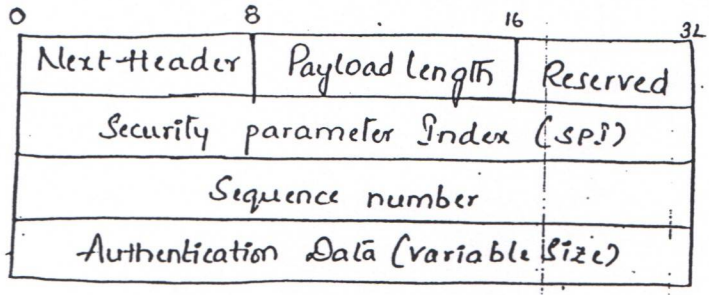
→ The Authentication Header provides support for data integrity and authentication of IP packets.

→ The data integrity service ensures that data inside IP Packets is not altered.

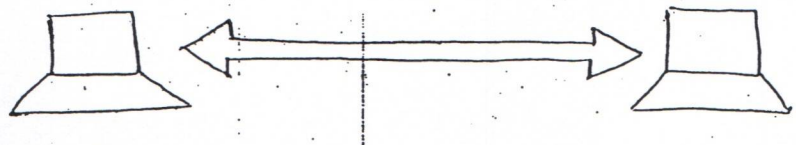
→ The authentication service enables an end user (or) a Computer System to authenticate the user (or) the application at the other end and decide to accept (or) reject packets accordingly.

→ Internally, AH is based on the Mac protocol, which means that the 2 communicating parties must share a secret key in order to use AH.

→ The AH structure is shown in the following figure.



Fields	Description
Next Header	– This 8-bit field identifies the type of header that immediately follows the AH.
Payload length	– The 8-bit field is used to solve the length of the AH in 16 bit words minus 2
Reserved	– This 16-bit field is reserved for future use
Security parameter Index (SPI)	– This 32-bit field is used for Source & destination addresses IP sec protocol used (AH to ESP), to uniquely identify the Security association (SA) for the traffic to which a datagram belongs.
Sequence number	– This 32-bit field is used to present replay attacks.
Authentication data	– This variable length field contains the authentication data, called as the Integrity check value (ICV) for the datagram. The ICV is calculated using



SA : 1 Traffic direction : Incoming Protocol : AH	SA : 1 TD : I P : AH
SA : 2 Traffic direction : Outgoing Protocol : AH	SA : 2 TD : O P : AH
SA : 3 Traffic direction : Incoming Protocol : ESP	SA : 3 TD : I P : ESP
SA : 4 Traffic direction : Outgoing Protocol : ESP	SA : 4 TD : O P : ESP

→ The SAD contains the following information

SAD fields	Description
1. Sequence number Counter	– It is a 32 bit field to generate the Sequence no. which is used in AH & ESP
2. Sequence Counter Overflow	– This flag indicates whether the over flow of the Sequence (or) not and present futher transmission of packets on this SA.
3. Anti replay window	– A 32 bit field which is used to detect whether any packet is replaying (or) not
4. AH authentication	– AH authentication algorithm & required key.
5. ESP Encryption	– Encryption algorithm and key details
6. IPsec protocol mode	– Transport mode / Tunnel mode
7. Path maximum transfer unit (MTU)	– the max size of an IP model
8. Life time	– Specifies the life of the SA

Transport Modes

→ In transport mode it doesn't hide the actual source and destination addresses.

→ They are visible in plaintext while in transmission.

The Internet key Exchange (IKE) protocols

→ This protocol is also used in IP sec for the key management Procedures.

→ Thus IKE is the initial phase of IPsec, where the algorithms & keys are decided.

→ After the IKE phase, the AH and ESP protocols take over.

→ The O/p of the IKE phase is known as Security Association (SA)

→ SA is an agreement blw the communicating process parties about the factors such as the IP sec protocol version in use, mode of Operation (transport mode / tunnel mode), Cryptographic algorithms, Cryptographic keys, Lifetime of keys etc.

→ Once this is done, both major protocols of IPsec (ESP) make use of SA for their actual Operation.

→ If both AH and ESP are used, each Communicating party requires 2 sets of SA, one for AH and another for ESP.

→ SA is a Simplex i.e; unidirectional, Therefore we need two sets of SA per communicating party i.e; one for incoming transmission and another for outgoing transmission.

→ Thus, if 2 communicating parties use both AH and ESP, each of them would require four sets of SA.

→ Both communicating parties must allocate some storage area for SA information at their end.

→ This is known as Security Association Database (SAD) and is predefined which is used by ipsec.

→ On the receiver side, the IPsec processes the AH first and then decryption can be done.

→ Both AH and ESP can be used in one of 2 modes.

(i) Tunnel mode

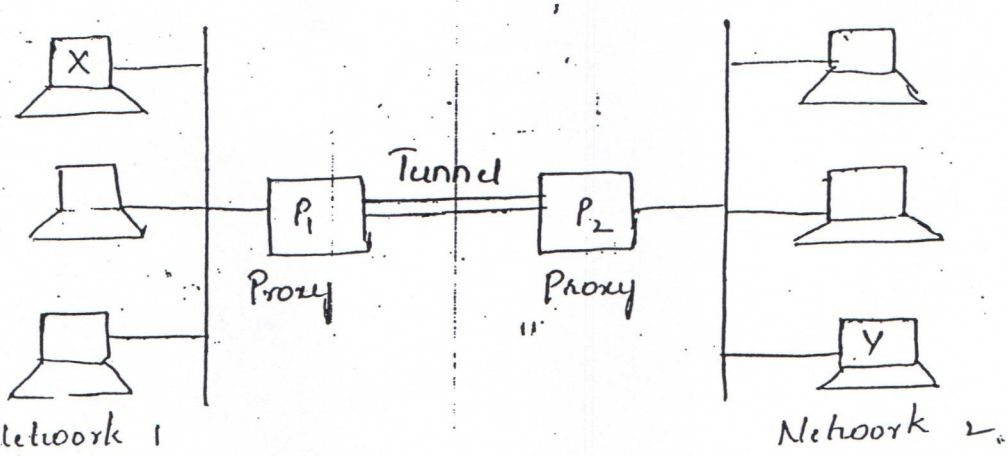
(ii) Transport mode.

Tunnel Mode:

→ Suppose X and Y are two systems wants to communicate each other using IPsec tunnel mode.

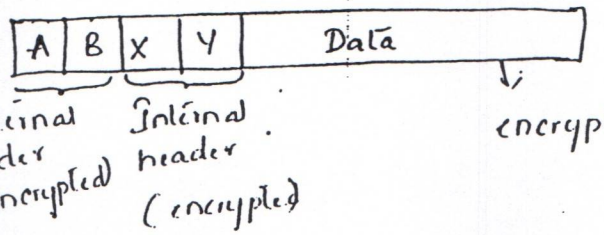
→ First they identify their respective provides P_1 and P_2 and a logical encrypted tunnel is established b/w P_1 and P_2

→ X sends its transmission to P_1. The tunnel carries the information to P_2. P_2 forwards this to Y.

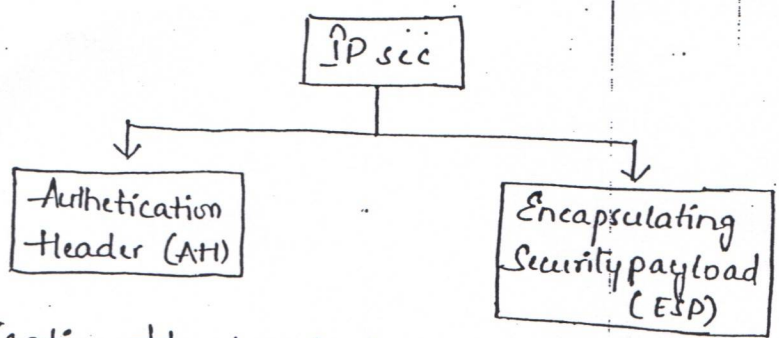


→ Here we will use 2 sets of IP headers (i) Internal
(ii) External

→ The internal header which is encrypted contains the source and destination addresses as X and Y, where as the external IP header contains the addresses of P_1 and P_2.



→ we know IP packets consists of two portions
  (i) IP header
  (ii) Actual data
→ IP sec features are implemented in the form of additional IP headers called as extensions headers to the default IP headers
→ IP sec offers two main services (i) Authentication
  (ii) Confidentiality.
→ Each of these requires its own extension header.
→ To support these 2 services IP sec defines two IP extension headers one for authenticatim & another for confidentiality.

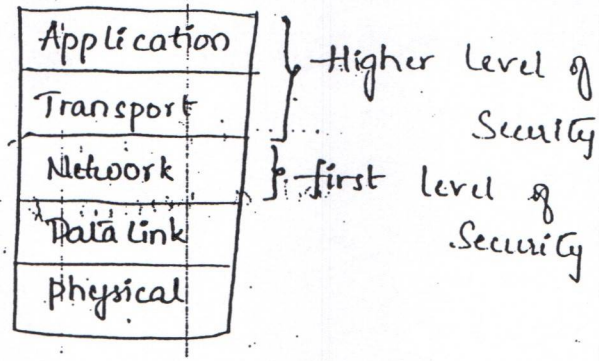


Authetication Header (AH):
→ The IP sec AH is a header is an IP packet, which contains a cryptographic checksum (Similar to message digest/hash) for the contents of the packet.
→ The AH is simply inserted b/w the IP header and the Packet contents.
→ No changes are required to the data contents of the packet.
→ The Security resides completely in the contents of the AH.

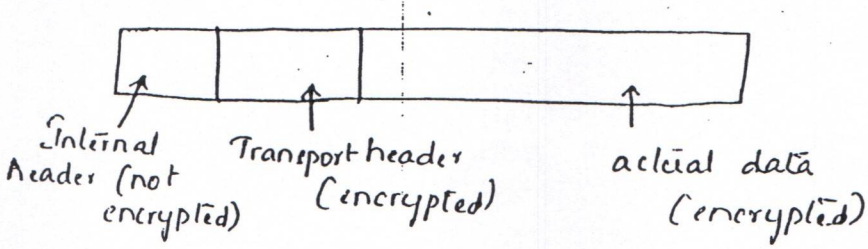
Encapsulating Security payload (ESP):
→ ESP protocol provides data Confidentiality.
→ It also defines a new header to be inserted into the IP Packet
→ ESP protocol changes the contents of the packet into an unreadable format.
→ The ESP header will be inside the AH.
  i.e; Encryption happens first, and then will

IP Security:

→ The IP packets contain data in plaintext form.
→ Therefore any one can access them, read their contents and even change them.
→ we know the mechanism for higher-level security such as SSL, PGP, S/MIME etc.
→ These are used in Application and Transport Layers of TCP/IP Protocol.
→ If we can provide security at the network layer itself then we need not rely much on the higher level security mechanisms.
→ we will use this layer higher level mechanisms as additional security measures.



→ The protocol which is used to provide security at the IP level called as IP Security (IP sec)
→ The Overall idea of IP sec is to Encrypt and seal the transport & application layer data during transmission
→ The logical format of the message after IP sec Processing is looks like



→ The sender and receiver look at IP sec as another layer in the TCP/IP Protocol stack and lies in b/n the transport and the network layer.

One-way Authentication:

→ One-way authentication involves a single transfer of information from one user to another user

→ user A sends its identity and a message for B.

→ The message includes a timestamp t_A, a nonce X_A, identify of B signed with A's public key.

→ Time stamp consists of an optional generation time and an expiration time

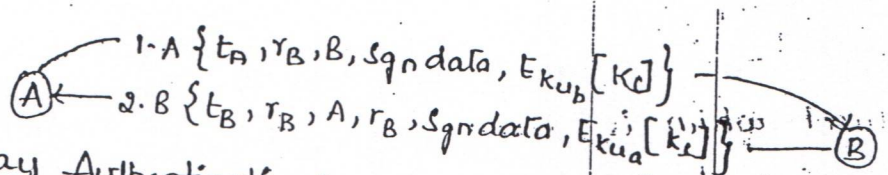
→ It also includes the Original data that is signed by A, and a session key which is encrypted with B's public key.

(A) ⟶ A {t_A, r_A, B, Signdata, E_{ku_b}[ ]} ⟶ (B)

Two-way Authentication:

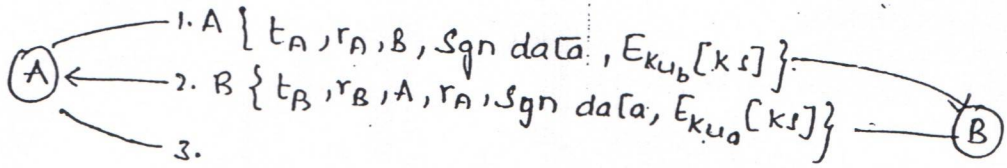
→ In addition to the above list, in this procedure the receiver B sends the identity of B and a message for A.

→ The message includes, time stamp by B, nonce by B, is signed by B, Signed data, and a version key encrypted with A's public key.



Three way Authentication:

→ In this procedure, the final message from A to B is signed copy of nonce r_B.



→ In the above figuare if A wants to access the B's certificate than it has to take all the certificates in the following manner.

$X\langle\langle W\rangle\rangle W\langle\langle V\rangle\rangle V\langle\langle Y\rangle\rangle Y\langle\langle Z\rangle\rangle Z\langle\langle B\rangle\rangle$

→ Suppose if B wants to access the A's Certificate.

$Z\langle\langle Y\rangle\rangle Y\langle\langle V\rangle\rangle V\langle\langle W\rangle\rangle W\langle\langle X\rangle\rangle X\langle\langle A\rangle\rangle$.

Revocation of Certificates:

→ Each certificate includes a period of validity.
→ A new certificate is issued just before the expiration of old ones
→ for the following reasons the revocation of the Certificate can be done

(i) Suppose if the secret key is assumed to be known
(ii) The user is no longer certified by the CA
(iii) The CA's certificate is assumed to be known.

→ Each CA must maintain a list containing of all revoked bit not expired certificates issued by the CA, including both these issued to users and to other CA's.
→ Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes the issuers name the date the list was created, the date the next CRL is scheduled and any early entry for each revoked certificate.
→ Because Serial no.s are unique with in a CA, the Serial numbers is sufficient to identify the certificate.
→ when a user receivers a certificate in a message, the user must determine whether the certificate has been revoked.

Authentication procedures:

→ X.509 pr includes 3 alternative authentication procedures to share the information
→ All these procedures use public key signatures.
→ it is assumed that 2 parties know each other's public key, either by obtaining each other's certificates from the directory/because the certificate is included in the initial message from each side.

Obtaining a user's certificate:

→ user certificate generated by a CA have the following characteristics

- Any user wants to access the CA's public key, it is possible only for the users they are certified by CA.
- No party other than the CA can modify the certificate.

→ Eg: Suppose we have 2 system A and B. A has obtained a certificate from the CA X_1 and B has obtained a certificate from the CA X_2.

→ If A doesn't have the public key of X_2 then B's certificate issued by X_2 is useless to A.

→ If the two CA's have securely exchange their own public keys, the following procedure will enable A to obtain B's public key:

1. A obtains the certificate of X_2 signed by X_1 from the dictonary
2. Now A then go back to the directory & receives the B's certificate which is signed by X_2

→ This can be noted in the following manner

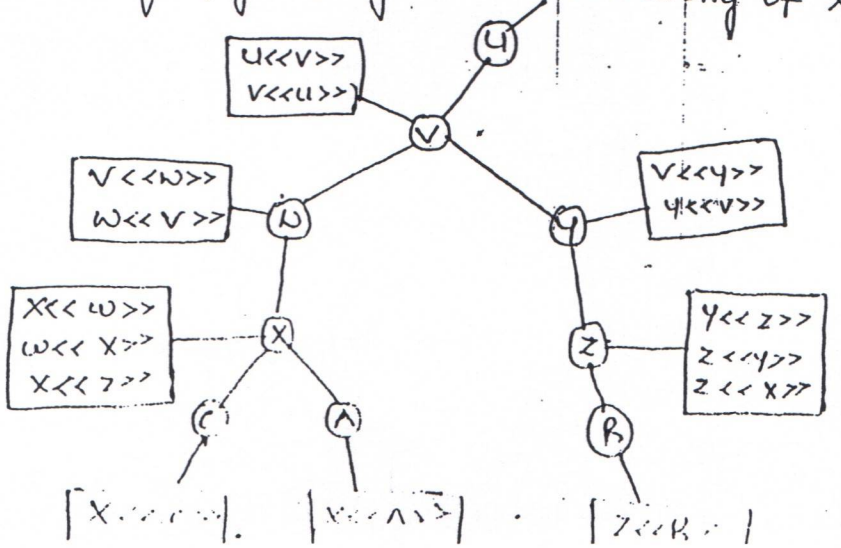
If 'A' wants to obtain the certificate of 'B'

$X_1 << X_2 >> X_2 << B >>$

If 'B' wants to obtain the certificate of A.

$X_2 << X_1 >> X_1 << A >>$.

→ The following figure gives the hierarchy of X.509 standard &



Period of validity: Consists of two values dates, the first and last dates of the certificate.

Subject name: The name of the user to whom this certificate refer ise; this certificate certifies the public key of the subject who holds the corresponding private key.

Subject's public-key information: The public key of the subject plus an identifier of the algorithm for which this key is to be used and any parameter needed

Issuer unique identifier: It is an optional bit string which is used to identify uniquely the issuing CA.

Subject unique identifier: An optional bit string which is used to identify uniquely the subject.

Extensions: A set of one or more extensions fields. These were added in version 3.

Signature: It covers all of the other fields of the certificate it contains the hash code of the other fields encrypted with the CA's private key.

→ This X.509 standard uses the following notation to define a certificate:

$$CA \ll A \gg = CA\{V, SN, AI, CA, T_A; A; A_P\}.$$

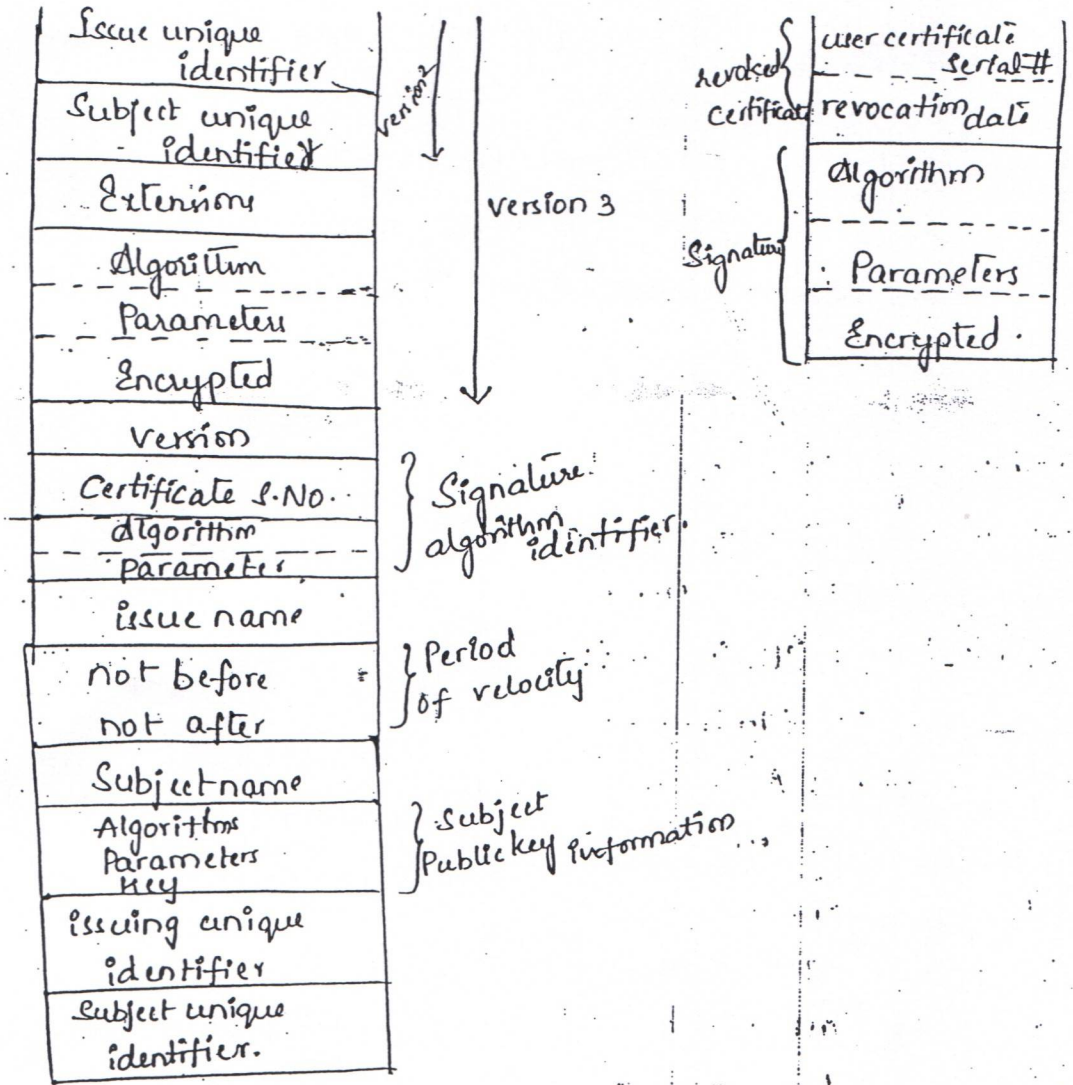
where

$Y \ll X \gg$ = The certificate of user X issued by certification authority Y.

$Y[I]$ = The signing of I by Y. It consists of I with an encrypted hash code appended.

→ The CA signs the certificate with its secret key.

→ If the corresponding public key is known to user, then that user can verify that a certificate signed by the CA is valid.



→ The elements in the digital certificate are

Version: The default version is — Version 1

If the unique identifier (or) Subject unique identifier are present — Version 2

If one (or) more extensions are present — Version 3

Serial Number: It is an Integer value to provide unique identify for the certificate.

Signature algorithm: The algorithm used to Sign the certificate identifier: and the parameter associated with it.

Issure name: X 500 name of the CA that created and Signed this certificate.

→ If authentication is the primary concern then

$A \rightarrow B : M \,||\, E_{KR_a}[H(M)]$

→ To provide more security than use the following

$A \rightarrow B : E_{KU_b}[M \,||\, E_{KR_a}[H(M)]]$

→ we will also provide some time stamp to this approach

$A \rightarrow B : M \,||\, E_{KR_a}[H(M)] \,||\, E_{KR_{as}}[T \,||\, ID_A \,||\, KU_a]$

X.509 Authentication Service:

→ It is a directory service in which a database will be maintained about the users.

→ It also includes the information about the mapping b/w user name to network addresses.

→ It is based on the use of public key Cryptography and digital signatures.

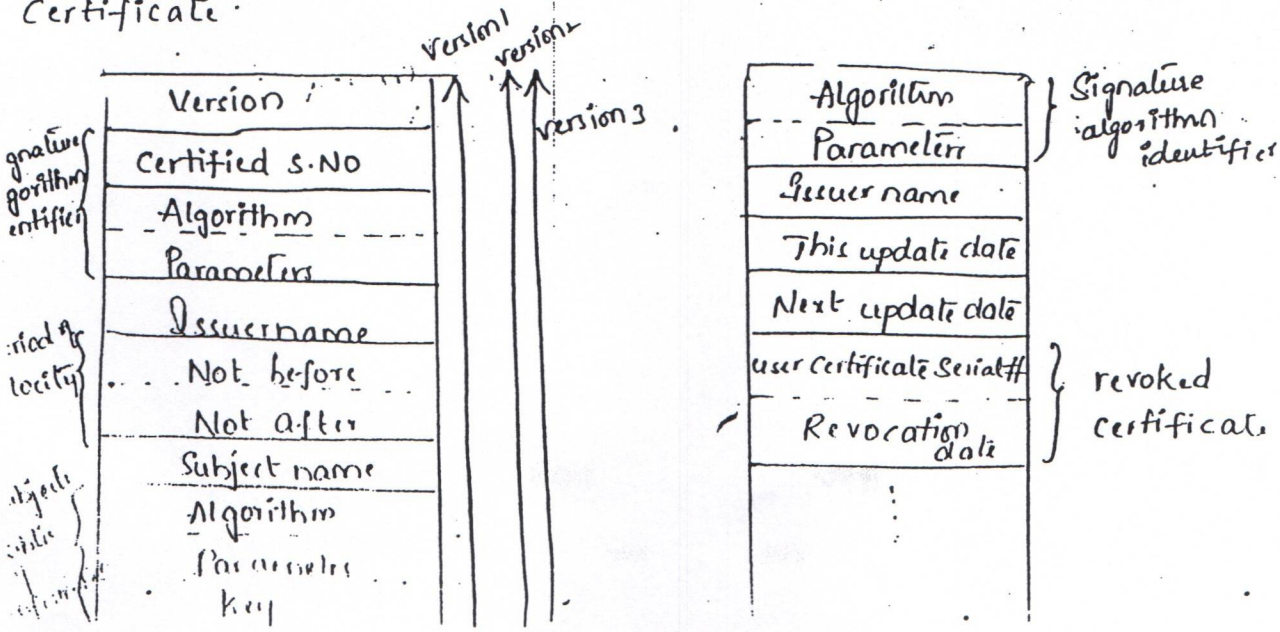
→ Most of the times it uses RSA & for digital signatures it uses hash functions.

→ This service depends upon the digital certificates.

Certificates:

→ In this scheme each user uses the public key certificates : those created by the certification authority (CA) and placed in the directory by CA / by user.

→ The following figure shows the general format of a certificate.



1) $A \rightarrow KDC : ID_A \| ID_B$

2) $KDC \rightarrow A : E_{KR_{auth}} [ID_B \| ku_b]$

3) $A \rightarrow B : E_{ku_b} [Na \| ID_A]$

4) $B \rightarrow KDC : ID_B \| ID_A \| E_{ku_{auth}} [Na]$

5) $KDC \rightarrow B : E_{KR_{auth}} [ID_A \| ku_a] \| E_{ku_b} [E_{KR_{auth}} (Na \| K_s \| ID_A \|$

6) $B \rightarrow A : E_{ku_a} [E_{KR_{auth}} (Na \| k_s \| ID_A \| ID_B] \| N_b]$

7) $A \rightarrow B : E_{K_s} [N_b]$

One-way Authentication protocol:

→ In this approach it is not necessary for the sender and receiver to be online at the same time

→ Eg: E-mail

Convential Encryption Approach:

→ The protocol will contain the following steps.

1) $A \rightarrow KDC : ID_A \| ID_B \| N_1$

2) $KDC \rightarrow A : E_{ka} [K_s \| ID_B \| N_1 \| E_{k_b} [K_s \| ID_A]]$

3) $A \rightarrow B : E_{k_b} [K_s, ID_A] \| E_{K_s} [M]$

→ Here, the sender A doesn't require the response from the receiver B.

→ whenever B is free, he Opens this message and extract the informan,

→ Suppose if the sender wants to send the message with tonestamp then it is also possible.

Public-key Encryption Approach:

→ These approaches require that either, the sender know the receptents public key (confidintiality) or the receptent-know the sender's public key(authentication) or both.

→ In addition, the public key algorithm must be applied once/twice to what may be a long message.

→ If confidentiality is the primary concern then

$A \rightarrow B : E_{ku_b} [K_s] \| E_{K_s} (M)$.

Public key Encryption Approaches

→ Here an authentication server is used to issue the public key certificates of the each party.

1) $A \rightarrow As : ID_A \| ID_B$
2) $As \rightarrow A : E_{KR_{as}}[ID_A \| KU_a \| T] \| E_{KR_{as}}[ID_3 \| KU_b \| T]$
3) $A \rightarrow B : E_{KR_{as}}[ID_A \| KU_a \| T] \| E_{KR_{as}}[ID_B \| KU_b \| T] \| E_{KU_b}[E_{KR_a}[K_s \| T]$

→ Here the session key is chosen by A.
→ Here it require the synchronization of the clocks.
→ woo and Lam had proposed another protocol which uses the nonce to authenticate the parties:

1) $A \rightarrow KDC : ID_A \| ID_B$
2) $KDC \rightarrow A : E_{KR_{auth}}[ID_B \| KU_b]$
3) $A \rightarrow B : E_{KU_b}[N_a \| ID_A]$
4) $B \rightarrow KDC : ID_B \| ID_A \| E_{KU_{auth}}[N_a]$
5) $KDC \rightarrow B : E_{KR_{auth}}[ID_A \| KU_a] \| E_{KU_b}[E_{KR_{auth}}[N_a \| K_s \| ID_a]]$
6) $B \rightarrow A : E_{KU_a}[E_{KR_{auth}}[N_a \| K_s \| ID_B] \| N_b]$
7) $A \rightarrow B : E_{K_s}[N_b]$

→ Here A wants to communicate with B for that purpose A asks the KDC that to send the authentication of B.
→ KDC will provide the public key of B to A in an encrypted format.
→ A generates the nonce & send it to B along with the identification of A encrypted with B's public key.
→ B asks the KDC whether the system A is authenticated / not by sending the A's nonce and A's identification to KDC
→ KDC issues the public key of A with an encrypted format and it also sends the selection key k_1 to B.
→ Now B and A shares the seetion key.
→ An advanced version this protocol was also developed i.e;

→ To Overcome this problem, Denning had proposed another protocol which contains the time stamp for the messages.

→ It is the modification of the Needham protocol and it contains the following steps:

1) $A \longrightarrow KDC : ID_A \| ID_B$

2) $KDC \rightarrow A : E_{Ka}[K_s \| ID_B \| T \| E_{Kb}[K_s \| ID_A \| T]]$

3) $A \rightarrow B : E_{Kb}[K_s \| ID_A \| T]$

4) $B \rightarrow A : E_{Ks}[N_1]$

5) $A \rightarrow B : E_{Ks}[f(N_1)]$

→ If the clocks if each party are synchronized then this protocol will be applicable

→ Suppose if sender's clock is a head of the receiver's clock then an opponent note down the sender's clock time and it sends the fault message to B when the receiver's clock shows the sender's time

So, to avoid this Needham has proposed another protocol which will give more security to the time stamp also.

The protocol will contain the following steps.

1) $A \rightarrow B : ID_A \| N_a$

2) $B \rightarrow KDC : ID_B \| N_b \| E_{Kb}[ID_A \| N_a \| T_b]$

3) $KDC \rightarrow A : N_b \| E_{KB}[ID_A \| K_s \| T_b] \; E_{Ka}[ID_B \| N_a \| K_s \| T_b]$

4) $A \rightarrow B : E_{Kb}[ID_A \| K_s \| T_b] \| E_{Ka}[N_b]$

→ Avoiding the need to contact the authentication server repeatedly

→ using the following protocol the Sender A communicates with the received B without consulting the KDC.

1) $A \rightarrow B : E_{Kb}[ID_A \| K_s \| T_b], N_a'$

2) $B \rightarrow A : N_b', E_{Ks}[N_a']$

3) $A \rightarrow B : E_{Ks}[N_b']$

→ Here synchronization of the clocks is not necessary

→ only 1 system will take care of the time stamp.

→ N_a' and N_b' are new nonces which enables the system that the message is not replaying

→ For this each party keep a back of this Sequence number which is difficult.
→ Because of this Overload the following general approaches are used.

Time stamps:
→ Each party accepts the messages only if the message contains a time stramp.
→ for this the clocks of each participant must be in Synchronize fashion.

Challenge/Responses
→ party A expecting a fresh message from B, first A Sends a nonce (challenge) to B & require a Subsequent message (response) from B which includes the nonce function value.

Convential Encryption Approaches
→ Between the Sender & receiver a third party KDC is present to issue the master keys & Session keys.
→ The protocol can be Summarized as follows.

1) $A \rightarrow KDC : ID_A \| ID_B \| N_1$
2) $KDC \rightarrow A : E_{Ka}[K_s \| ID_B \| N_1 \| E_{K_b}[K_s \| ID_A]]$
3) $A \rightarrow B : E_{K_b}[K_s \| ID_A]$
4) $B \rightarrow A : E_{K_s}[N_2]$
5) $A \rightarrow B : E_{K_s}[f(N_2)]$

→ Here the Secret key K_a is shared b/w A and KDC, and the Secret key K_b is shared b/w KDC and B.
→ The purpose of this protocol is to distribute a session key K_s security of A and B.
→ Here, we have problem in step 3
→ Suppose if an Opponent 'C' has Simply recorded this step 3, be then he knows the ID_A, K_s and K_b by applying some calculations.
→ Then he act as a original Seder 'A' and sends the meesage to B.
→ Here B Sends the information to 'C' instead of A.
→ This protocol was discovered by Newton and Sefiroedes.

Authentication protocols:

→ we have 2 protocols to provide authentication.

(i) Mutual authentication

(ii) One-way authentication.

Mutual authentication:

→ It is used to satisfy the parties mutually about their identities and to exchange session key.

→ In key exchange we have 2 problems i.e confidentiality and timeliness.

→ To prevent masquerade and to prevent compromise of session keys, essential identification and session key information most be communicated in encrypted form.

→ The second issue, timeliness, is important because of the threat of message replay.

→ The following will given the list of replay attacks..

Simple replay:

→ The Opponent simply copies a message and replays it later.

Repeatition that can be logged:

→ An opponent can replay a toneistamped message within the valid time.

Repeatition that can't be detected:

→ The Original message have been compressed and thus did not arrive at its destination; only the replay message arrives.

Backward replay without modification:

→ This is replay back to the message sender.

→ This attack is possible if convential encryption is used and the sender cannot easily recognize the difference b/w messages sent and messages received.

→ one approach to overcome this problem is to attach a sequence number to each message used in an authentication exchange.

→ A new message is accepted only if its sequence number is in the proper order

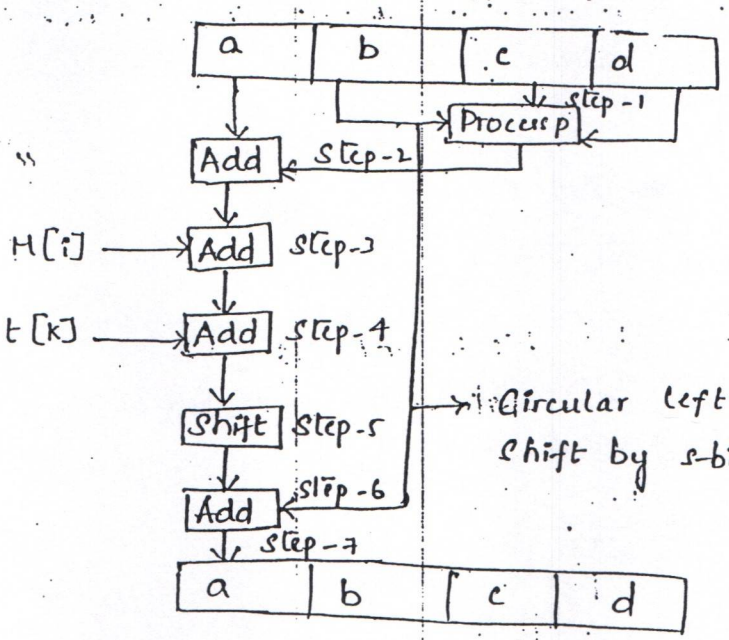
→ In all the four rounds only 1st step is different, other remains same

→ In each round we have 16 input sub-blocks named as M[0] each contain 32-bits.

→ 't' is an array of constants. It contains 64 elements with element consisting of 32-bits & we respect these elements as t[1], t[2], --- t[64]

→ we use 16 elements in each round.

→ The following figuare shows the steps involved in each round.



→ we can mathematically express a single MD5 Operation as follows

$$a = [[a + P(b,c,d) + M[i] + t[k] <<< s] + b]$$

→ In 4 rounds, only the process step will differ.

Round	Process P
1	(b AND c) OR [(NOT b) AND d]
2	$(b \wedge d) \vee (c \wedge \bar{d})$
3	$b \oplus c \oplus d$
4	$c \oplus (b \vee \bar{d})$

[AND - $\wedge$
OR - $\vee$
$\bar{d}$ - not d]

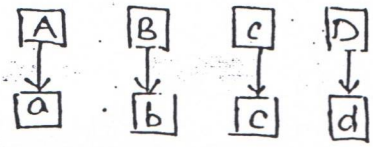
→ As a last step of MD5 algorithm we should do XOR operation b/w all 128 bit O/p blocks of 512 bit I/p blocks. The result of this XOR operation will produce a 128 bit length of message digest which ends

Step-5: process the blocks:

Step-5.1:

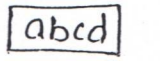
→ Copy four chaining variables into four corresponding variables a, b, c and d.

→ Thus we now have a=A, b=B, c=C, d=D.

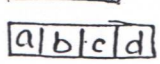


→ Actually the algorithm considers the combination of a, b, c and d as a 128-bit single register abcd.

→ This register holds the information about the intermediate and final results.



Abstracted view

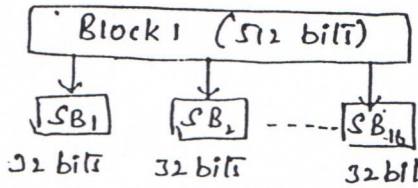


Internal view.

Step-5.2:

→ Divide the current 512-bit block into 16 sub-blocks

→ Each sub-block contains 32-bits.



Step-5.3:

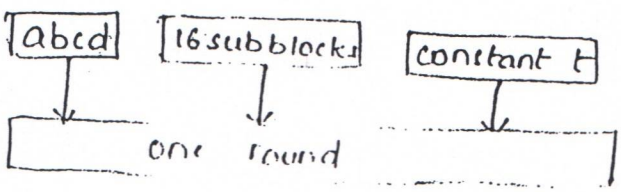
→ Now we have four rounds

→ In each round the inputs are

(a) all the 16-sub blocks

(b) the variables a, b, c, d.

(c) some constants designed as t.

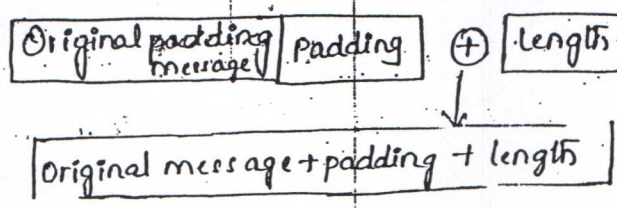


Step-2: Append length:

→ After padding bits are added, the next step is to calculate the original length of the message and add it to the end of the message i.e; after padding.

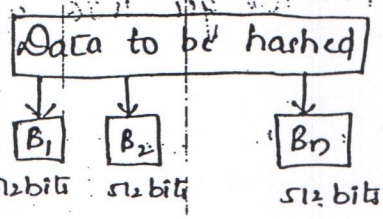
→ Eg: If the Original message consisted of 1000 bits and we add a 472 bits to make the length is considered as 1000 but not 1472 for this step only.

→ The length of the Original message exceeds 2^{64} bits then only last 64-bits of the length are used.



→ This now becomes the message whose digest will be calculated.

Step-3: Divide the input into blocks each block contains 512 bits:



Step-4: Initialize chaining variables:

→ In this step, four variables called chaining variables are initialised & represented as A, B, C and D.

→ Each of these is a 32-bit number.

→ The initial hexadecimal values of these chaining variables are

A → 01 23 45 67

B → 89 AB CD EF

C → FE DC BA 98

D → 76 54 32 10

MD5 [Message Digest]:

Introduction:

→ MD5 is a message digest algorithm developed by Rivest.
→ The Original message digest algorithm was called as MD
→ He invented next MD_2, MD_3, MD_4 & MD_5.
→ MD5 is quite fast and it produces 128-bits of message digest.

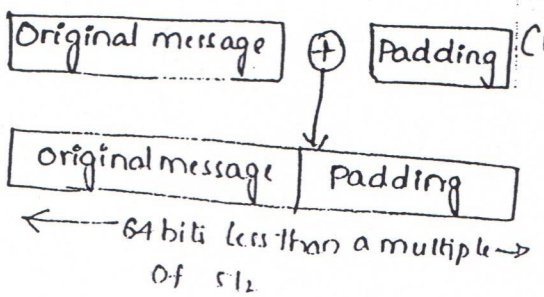
How MD5 works:

Step-1: padding:

→ The 1st step in MD5 is to add the padding bits to the Original message.
→ The main aim of this step is to make the length of the Original message equal to a value, which is 64 bits less than an exact multiple of 512.

Eg: If the Original message contains 1000 bits we add a padding of 472 bits to make the length of the message 1472 bits.
[If we add 64 bits to 1472 bits we get 1536 which is a multiple of 512. $1536 = 512 \times 3$]

→ Thus after padding, the Original message will have a length of 448 bits (64 bits less than 512), 960 bits (64 bits less than 1024), 1472 bits (64 bits less than 1536) etc.
→ The padding consists of a single 1-bit followed by as many 0-bits as required.
→ The padding is always added to the Original message.
→ The padding length is any value b/n 1 & 512 (∵ b/n → between)



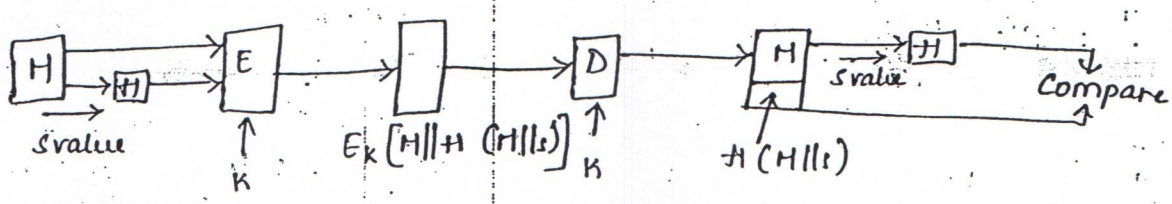
→ This techique uses a hash function but no encryption for message authentication.

→ In this case the two parties will share a secret value 's'.

→ A computer the hash value over the concatenation of M and and appends the resulting hash value to M.

→ on the receiver side it can recompute the hash value to verify

ii)



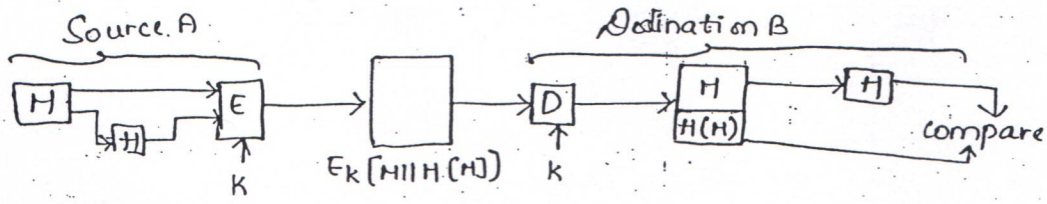
→ In this case the confidentiality can be added by encrypting the entire message plus the hash code.

→ Some of the hash algorithms are

i) MD5 [Message Digest]

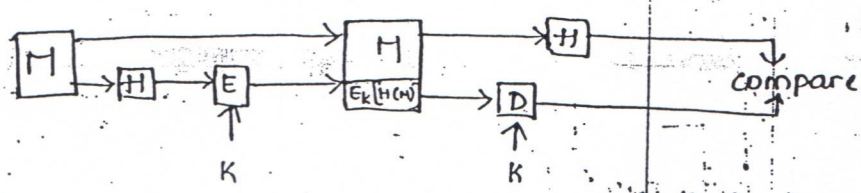
ii) SHA-1 (Secure Hash Algorithm)

(i)



→ Here the message plus the hash code is encrypted using symmetric encryption.

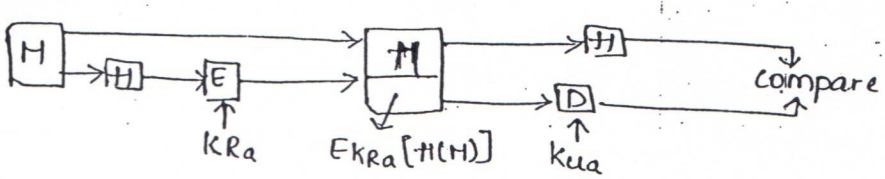
(ii)



→ In this case only the hash code is encrypted using symmetric encryption.

→ This reduces the processing burden for those applications that do not require confidentiality.

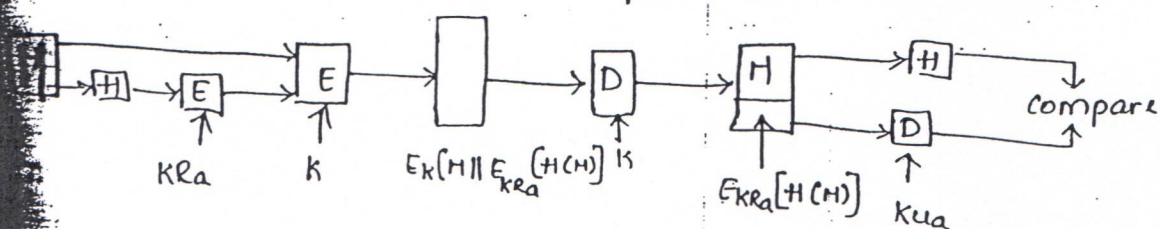
(iii)



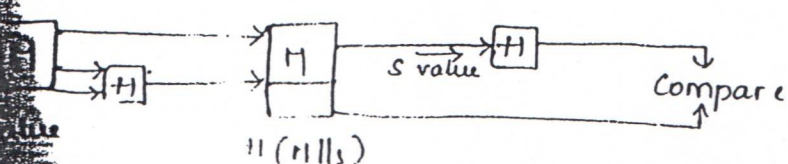
→ In this case also only the hash code is encrypted using public key encryption & using the sender private key.

→ In this case the ciphertext is in the form of digital signature.

(iv) → If confidentiality as well as digital signature is desired, then the message plus the public key encrypted hash code.

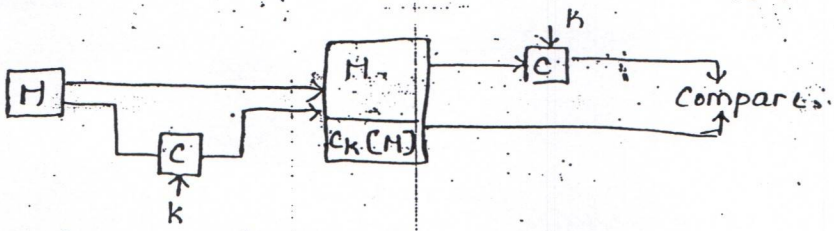


…an be encrypted using a symmetric secret key
…is is a common technique.

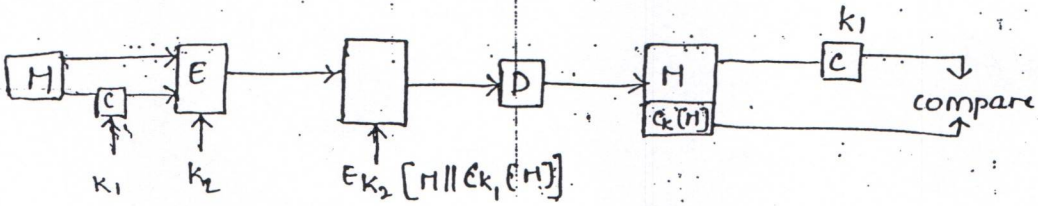


$MAC = C_K[H]$ where H = input message
C = MAC function
K = shared secret key
MAC = Message authentication code,

→ we calculate the MAC in the following Manner.

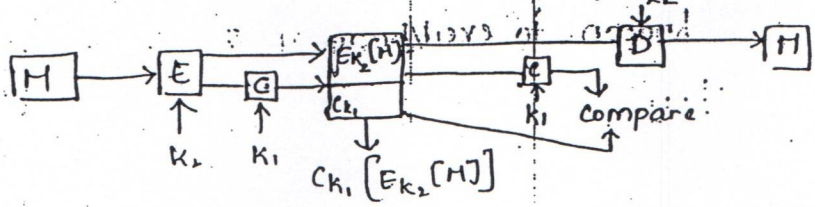


(a) Message Authentication.



(b) Message authentification & Confidentiality.

→ Here authentication tied to plaintext.



(c) Message authentification is Confidentiality.

→ Here authentification tied to cipher text.

Hash function:

→ This is another authentication technique

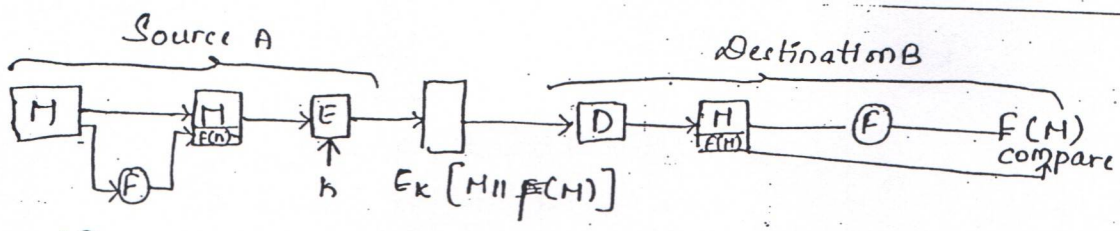
→ Hash function accepts a variable size of Message H as input and produces a fixed-size output referred to as a hash code $H(H)$

→ It doesn't use any secret key like MAC function.

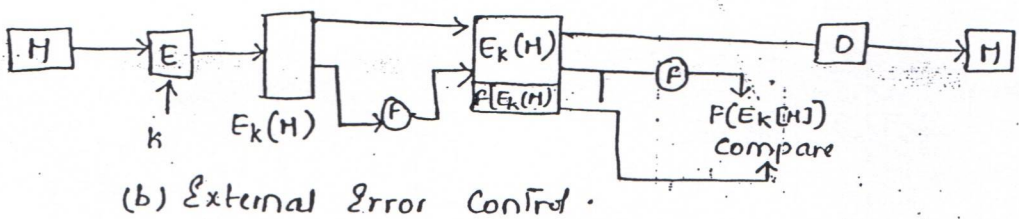
→ It is only a function which is applied to the input message.

→ The hash code also known as a message digest / a hash value

→ we have so many ways in which a hash code can be used to provide message authentication.



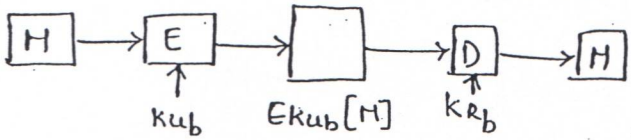
(a) Internal Error Control.



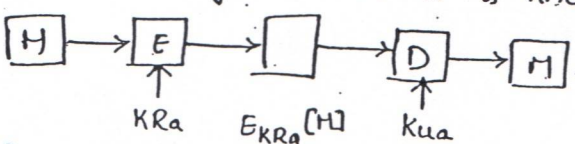
(b) External Error Control.

Public key Encryption:

→ The straight forward method of public encryption it provides confidentiality but not authentication

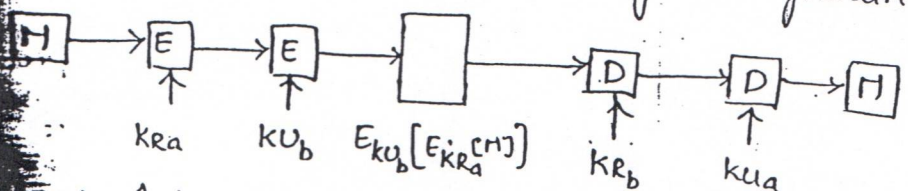


→ In this technique the intruder may trap the message using the public key, because it is known to everybody at the sender's side



→ In this scheme at the destination side the intruder may trap the message.

→ So, to avoid this problem we have another method which provides confidentiality, authentication & digital signature.



Message Authentication Code (MAC):

→ This is an alternative authentication technique involves the use of secret key to generate a small fixed-size block of data, known as cryptographic check sum / MAC.

→ When A has a message to send to B, it calculates the MAC as a function of the message and the key.

Authetication function:

→ We have 3 functions in order to produce an authentication

- Message Encryption
- Message authentication code (MAC)
- Hash function.

Message Encryption:

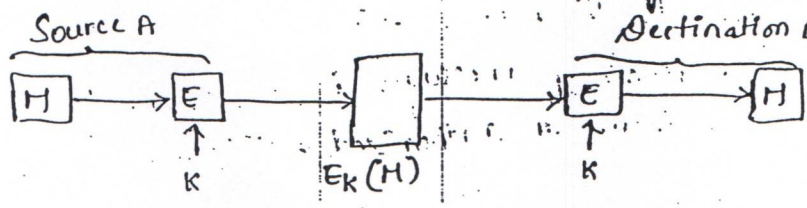
→ Message Encryption is itself can provide a measure of authentification.

→ The analysis differs for symmetric & public key encryption schemes

Symmetric encryption:

→ A message M transmitted from source A to destination B is encrypted using a secret key 'K' shared by A & B.

→ If no other party knows the key then confidentiality is provided otherwise it leads to trapping.



→ To make this scheme secure we use an error-detecting code / frame check sequence (FCS) to each message before encryption.

→ A prepares a plaintext M and then provides this message as the input to a function F that produce an FCS.

→ Now this both message & FCS block is encrypted.

→ At the destination, B decrypts the incoming block & separates the message & FCS.

→ And B applies, the same function F to repradare the FCS

→ If the calculated FCS is equal to the incoming FCS then the message is considered authentic.

UNIT-III

Message Authentication and Hash functions

Authetication Requirements:

→ If the communication of network, the following attacks can be identified.

(1) Disclosure:

→ Release the message contents to any person (or) process not possessing the appropriate cryptographic key.

(2) Traffic Analysis:

→ Discovery of the pattern of traffic b/w parties.

→ In a connection-oriented application, the frequency & duration of connections could be determined.

→ And in a connection less environment, the number & length of messages b/w parties could be determined.

(3) Masquerade:

→ Insertion of messages into the network from a fault source.

→ This includes the creation of messages by an apparent that are purported to come from an authorized entity.

(4) Content modification:

→ changes to the contents of a message including insertion, deletion and modification etc.

(5) Sequence modification:

→ Any modification to a sequence of messages b/w parties including insertion, Deletion and reordering.

(6) Timing modification:

→ Delay / replay of messages.

(7) Source repudiation:

→ Denial of transmission of message by source.

(8) Destination Repudation:

→ Denial of receipt of message by destination

→ Message authentication is a procedure to verify that received messages come from the legal source & have not been altered.

Diffie-Hellman key Exchange:

→ The purpose of the algorithm is to enable 2 users to exchange a key securely that can be used for subsequent encryption of messages.

→ First we define a primitive root of a primenumber p as the whose powers generate all the integers from 1 to p-1.

→ i.e; if 'a' is a primitive root of the prime numbe p, then the numbers $a \bmod p, a^2 \bmod p, \ldots a^{p-1} \bmod p$ are distinct & consist of the integers from 1 through p-1 in same permutation.

→ with this background we define Diffe-Hellman key exchange.

→ for this scheme, there are 2 publicly known numbers requiti namely a prime no. 'q' & an integer α that is primitive root of q. ($\alpha < q$)

→ user A Selects a random integer which is privati sequential by x_A ($x_A < q$)

→ It calculates its public key $y_B = \alpha^{x_B} \bmod q$.

→ The generation of the Secret key by user A in the following manner (i.e) $K = (y_A)^{x_B} \bmod q$

→ user B also generates the Secret key in the samemanner as A i.e; $K = (y_A)^{x_B} \bmod q$.

→ These 2 results produce identical reculti.

$$K = (y_B)^{x_A} \bmod q$$

$$= (\alpha^{x_B} \bmod q)^{x_A} \bmod q \text{ [by the rule of modulu arithmetic]}$$

$$= (\alpha^{x_B})^{x_A} \bmod q$$

$$= (\alpha^{x_A})^{x_B} \bmod q$$

$$= (\alpha^{x_A} \bmod q)^{x_B} \bmod q$$

$$= (y_A)^{x_B} \bmod q$$

Simple Secret key Distribution:

→ If 'A' wishes to communicate with B, the following procedure is employed.

1) 'A' generates a key pair {kua, kra} and transmits a message to B consisting of kua & an identifier of A (ID_A)

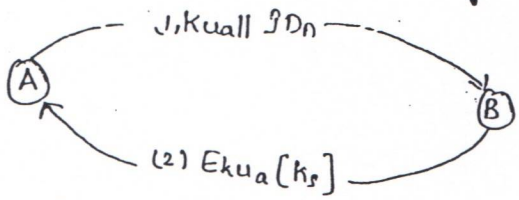
2) 'B' generates a secret key k_s & transmits it to A, encrypted with A's public key.

3) A computes decryption to recover the secret key like.
$D_{KRA}[E_{Kua}[K_1]]$.

4) Now the secret key is known to both the sender & receiver.

5) A discards kua and kra and B discards kua

→ Now A & B can now securely communicate using conventional encryption with the session key k_s.



Secret key distribution with confidentiality & authentication:

→ After the distribution of the publickeys b/w A and B the following steps will occur to distribute the secret key.

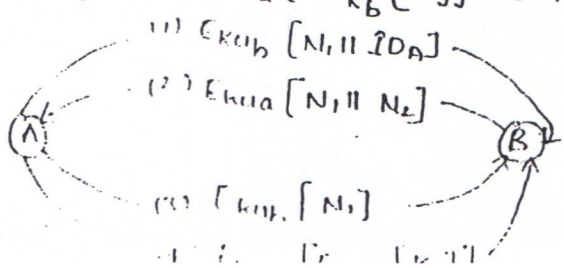
(1) A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) & a nonce (N_1)

(2) B sends a message to A with kua and containing A's nonce (N_1) as well as a new nonce generated by B (N_2).

(3) A returns (N_2), encrypted using B's public key to answer B that its correspondent is A.

(4) A selects a secret key k_s & sends $M = E_{KU_b}[E_{KR_a}[K_1]]$. to B.

(5) Now B computes $D_{Kua}[D_{KR_b}[H]]$ to recover secret key.



Public-key Certificates:

→ This approach is used to exchange the keys without contacting a Public-key authority

→ A participant conveys its key information to another transmitting its certificate.

→ Any participant can read a certificate to determine the A and Public key of the certificate's owner.

→ only the Certificate authority can create and update certification

→ Each participant applies to the Certificate authority, supplying a public key and requesting a certificate.

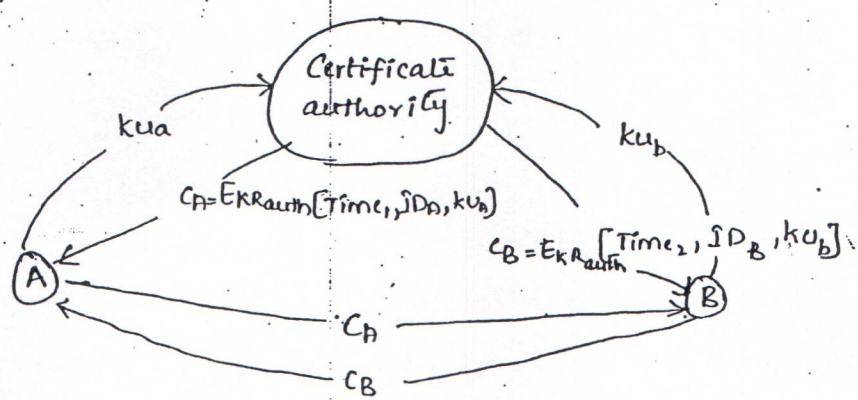
→ for participant A, the authority provides a certificate of the form.

$C_A = E_{KR_{auth}}[T, ID_A, ku_a]$ KR_{auth} → private key.

→ A may then pass its certificate on to any other participant who reads & verifies the certificate as follows.

$D_{KU_{auth}}[C_A] = D_{KU_{auth}}\left[E_{KR_{auth}}[T, ID_A, ku_a]\right] = [T, ID_A, ku_a]$

→ The responder uses the authority's public key ku_{auth} to decrypt the certificate.

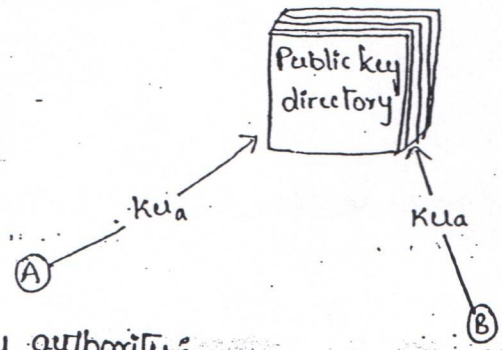


Public key distribution of Secret keys:

→ once the public keys have been distributed/have become accessible the intruders can trap the keys.

→ so, to avoid this, the secret keys should be generated

→ using public key encryption the distribution of secret keys can be done which are further used for conventional.



Public key authority:

→ The stranger security for public key distribution can be achieved by Providing tighter control over the distribution of public keys from the directory.

→ Here, each participant knows the public key of the authority but only the authority knows the corresponding private key.

→ The following steps are involved in keeping the public keys of other systems for future.

(1) 'A' sends a time stamped message to the public-key authority containing a request for the current public key of 'B'.

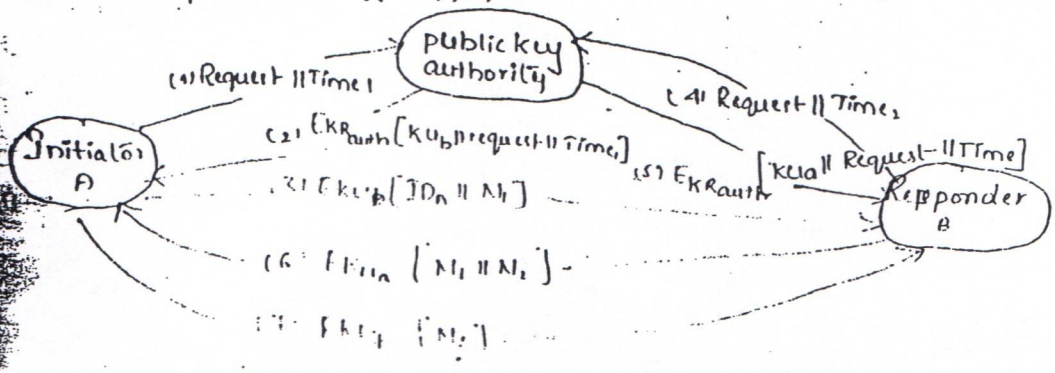
(2) The authority responds to that request and it sends the B's Public key, the time stamp all are encrypted with the authentication private key

(3) 'A' stores the B's public key & uses it to encrypt message to 'B' Containing an identified of A (ID_A) and nonce (N_1)

4 & 5 'B' retrieves A's public key from the authority in the same manner as 'A' retrived B's public key.

6) 'B' sends a message to A encrypted with KU_a and containing A's nonce and newly generated B's nonce.

7) A returns N_1 encrypted using B's public key to assure B that it correspondent is A.



Key Management in public key cryptography:

→ There are actually two distinct aspects to the use of public-key encryption
→ The distribution of public keys
→ The use of public-key encryption to distribute secret key.

Distribution of public keys:

→ This can be done in any one of the following ways.
* public announcement
* publicly availble directory
* public authority
* public-key certificates

Public announcement of public keys:

→ In this approach any participant can send his/her public key to any other participant / broadcast.
→ Although it is convinient, it has major weakness i.e any one can forge such a public announcement.

Publicly available directory:

→ In this approach a trusted third party / organisation will maintain a directory for public keys.
→ It takes the information from the system as {name, public key}
→ Each participant can register a public key by representation of the name / secure authenticated communication
→ Participants can change their public keys.
→ periodically, the authority publishes the directory / updates to the directory

6) Send c as the cipher text to the receiver.

7) Calculate plaintext $p = c^d \bmod n$.

Example:

1) The primenumbers be $x = 47$ and $y = 71$

2) Calculate $n = x.y = 3337$

3) i) find $(x-1)(y-1) = 46 \times 70 = 3220$.

ii) The factors of 3220 are 2, 2, 5, 7 & 23

[$\because 3220 = 2 \times 2 \times 5 \times 7 \times 23$]

iii) we have to choose 'e' value as none of the above factor values

iv) Let us choose $e = 79$

4) Let us substitute the values of e, x and y in the equation

$ed \bmod (x-1)(y-1) = 1$ | Select $d = 1019$

$(79 \times d) \bmod 3220 = 1$ | $(1019 \times 79) \bmod 3220 = 1$.

5) we are sending 688 value from A to the reason B.

∴ Encrypt this using the following formula,

$$C = p^e \bmod n$$

$$= 688^{79} \bmod 3337$$

$$= 1570$$

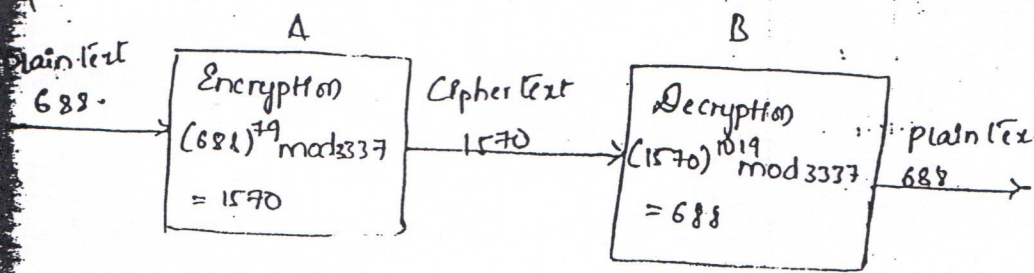
6) Send 1570 to the receiver B.

7) for decryption we use the following formula

$$P = (1570)^{1019} \bmod 3337$$

$$= 688.$$

$$P = c^e \bmod n;$$



RSA algorithm:

→ Rivest - Shamir - Adleman are together proposed this algorithm for the public-key encryption

→ so, this RSA name was given to this algorithm.

→ The RSA scheme is a block cipher in which the plaintext and ciphertext are integers b/w 0 & n-1 for same 'n'

→ A typical size for n is 1024 bits.

Description of the algorithm:

→ This scheme makes use of an expression with exponentials

→ plaintext is encrypted in blocks.

→ The block size must be less than / equal to $\log_2 n$.

Eg: If n = 1024 then block size must be less than or equal to $\log_2 2^{10} = 10$

→ Encryption & Decryption are of the following form for some plaintext p & ciphertext block c.

$$C = P^e \bmod n$$
$$P = c^d \bmod n.$$

→ Both sender & receiver must know the value of 'n'.

→ The sender knows the value of e and only the receiver knows the value of d.

→ so the public key is $kg = \{e, n\}$ and

private key is $kk = \{d, n\}$.

Algorithm:

1) choose 2 large prime numbers 'x' & 'y'

2) calculate $n = x \times y$

3) select the public key 'e' such that it is not a factor of $(x-1)(y-1)$

4) Select the private key of such that the following operation is true

(i.e) $(ed) \bmod (x-1)(y-1) = 1$

5) Calculate ciphertext $C = P^e \bmod n$

→ The above structure will provide both Secrecy & authentication to the message.

→ In this approach A prepares a message to B and encrypts it using A's private key

→ B can decrypt the message using A's public key

→ Here the encrypted message is known as 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 & in terms of data integrity.

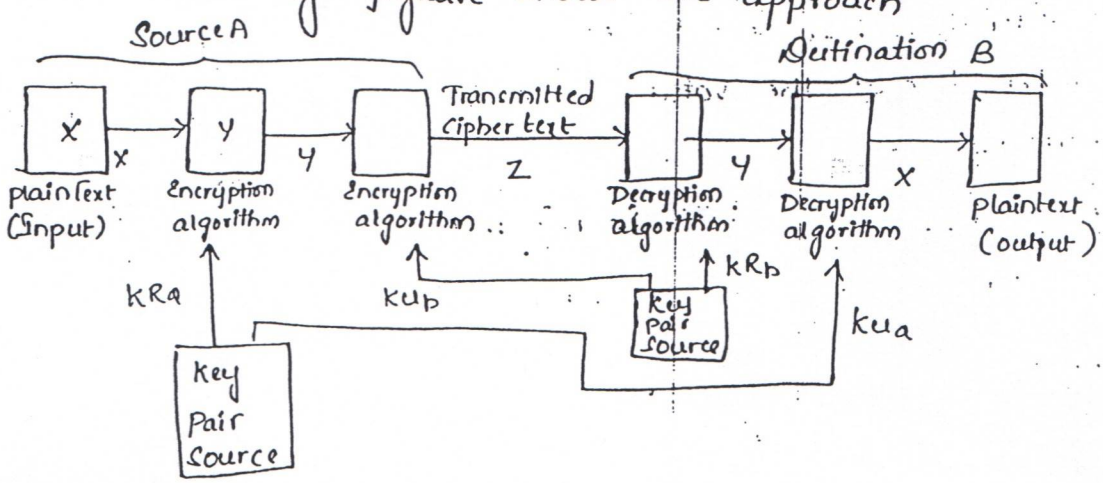
→ we can provide more security for our information by applying the above 2 methods.

→ Suppose 'A' is the sender & 'B' is the receiver then the public & private keys are represented by the following notations.

→ A's public key is KU_a
Private key is KR_a

→ B's public key is KU_b
Private key is KR_b

→ The following figure shows this approach



→ This Encryption & Decryption can be mathematically represented as

$$Z = E_{KU_b}\left[E_{KR_a}[X]\right]$$

$$Y = D_{KU_a}\left(D_{KR_b}[Z]\right)$$

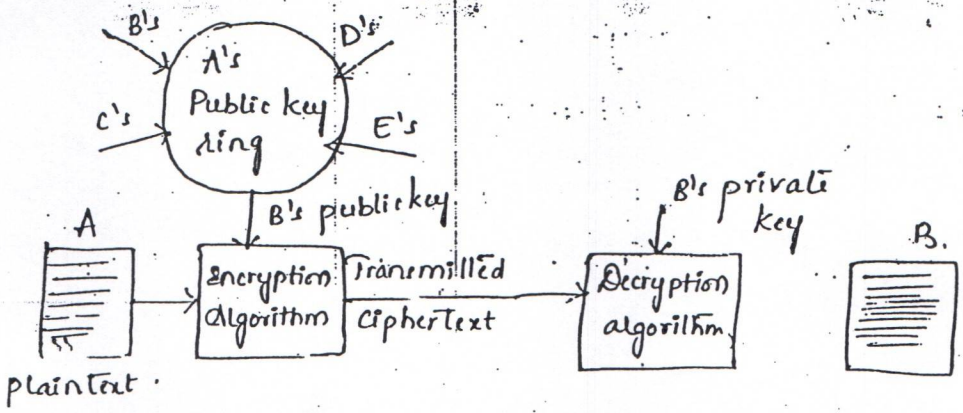
Public key and Assymmetric Cryptography :

→ In public key cryptography 2 different keys are used.

→ one key is used for encryption & only the other corresponding key must be used for decryption.

→ It is impossible to determine the decryption key by knowing the encryption key.

→ The public key encryption process can be defined in the following figure



→ In this process 'A' wants to communicate with the system

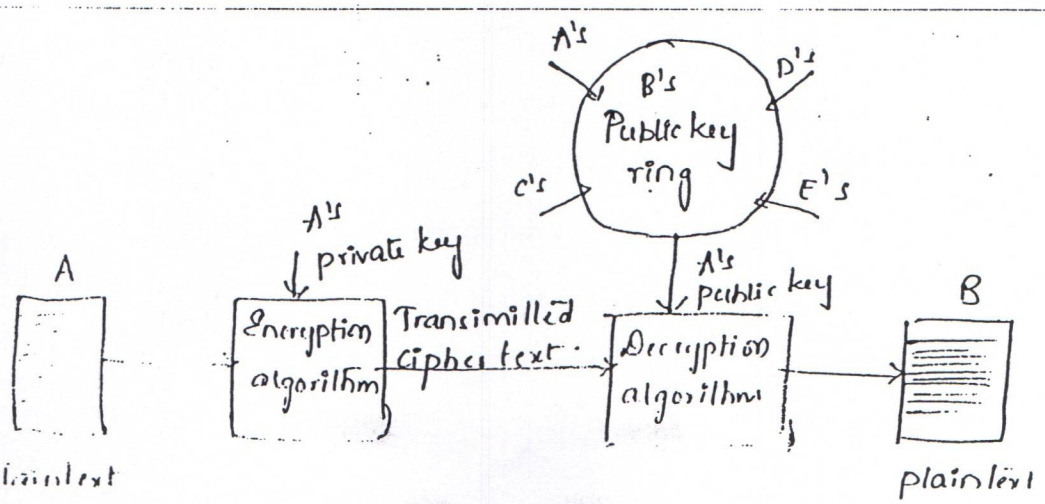
→ for this each end system in a network generates a pair of keys one is used for encryption & another is for decryption.

→ Each system publishes its encryption key by placing it in a public register. This is the public key

→ The companion key is kept private.

→ If A wishes to send a message to B it encrypts the message using B's public key.

→ when 'B' receives the message, B decrypts it using B's private key.



Session key life time:

→ If the session keys are frequently exchanged then we get more security.

→ for connection Oriented protocols use the same session key for the length of the time that the connection is open & use a new session key for each new session.

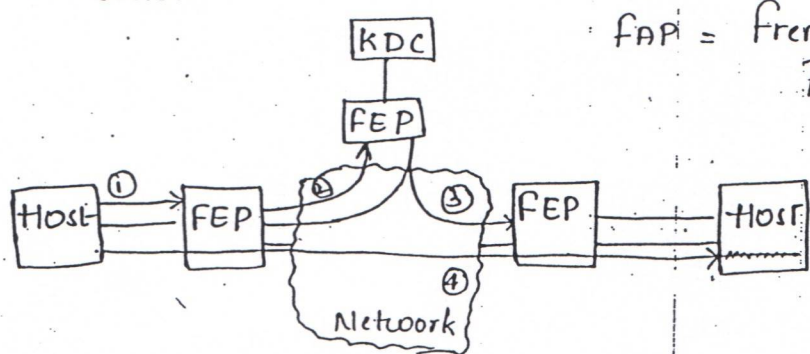
→ If a logical connection has a very long life time, then change the session key periodically.

A transparent key control scheme:

→ This scheme is useful for providing end-to-end encryption at a network / transport level in a way that is transparent to the end user.

→ The following figuare shows the steps involved in establishing a connection.

FAP = Frenlend Processor.



Step-1: when one host wishes to setup a connection to another host then it transmits a connection request packet.

Step-2: The FEP saves that packet & applies to the KDC for Permission to establish the connection.

Step-3: If the KDC approves the connection request, it generates the session key & delivers it to the 2 appropriate FEP using a permenent key for each front end.

Step-4: The requesting FEP can now release the connection request Packet and a connection is set up b/w the 2 end systems.

→ The advantage of this approach is that it minimizes the impact on the end systems.

3) A stores the session key and forwards to B the information that originated at the KDC for B namely $E_{Kb}[K_s \| ID_A]$

Now B knows the session key and who is its sender.

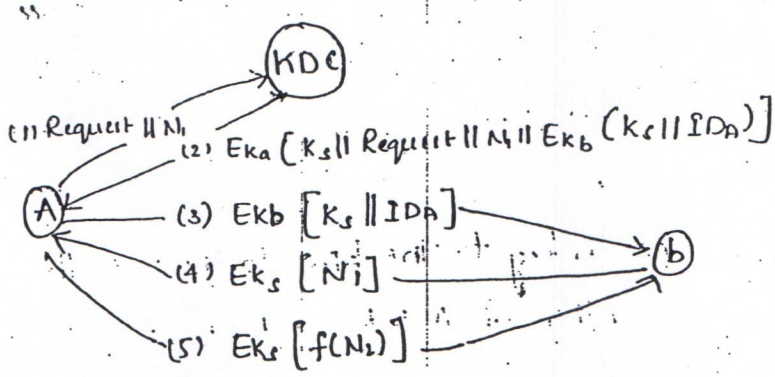
4) using the newly available session key for encryption, B sends a nonce N_2 to A.

5) Also using K_s A responds with $f(N_2)$ where f is a function that performs some transformation on N_2.

→ Actual key distribution involves only at step 1 through step 3

→ But the steps 3, 4 & 5 perform an authentication function.

The following figuare



Hierarchical Key Control:

→ If a single KDC is present there is difficult to distribute the keys among a large network.

→ As an alternative, a hierarchy of KDC's can be established

→ for eg; there can be local KDC's, each responsible for a small domain of the Overall internetwork such as a single LAN.

→ for local communication local KDC is responsible for key distribution.

→ If entities in different domains desire to share a key then the corresponding KDC's can communicate through a global KDC

→ In this case any one of the KDC's can select the Key

Key Distribution Scenario:

→ The key distribution concept can be done in many ways.

i) A key can be selected by A and physically delivered to B.

ii) A 3rd party can select the key & physically deliver it to A & B.

iii) If A & B have previously & recently used a key, one party can transmit the new key to the other, encrypted using the old key.

iv) If A & B each has an encrypted connection to a 3rd party C. 'C' can deliver a key on the encrypted links to A & B.

→ The scenario assumes that each user shares a unique master key with the 'KDC'.

→ let us assume, that user A wishes to establishes a logical connection with B and requires a session key to protect the data transmission.

→ 'A' has a master key K_a which is known to itself & KDC only.

→ 'B' wants to share the master key K_b with the KDC.

→ The following steps occur:

i) A issues a request for session key to the KDC to protect the data. The message includes the identity of A & B and a unique identifier N_1 which is known as a nonce. This nonce may be a time stamp a counter / a random number.

ii) The KDC responds with a message encrypted using k_a. Thus A is the only one who can successfully read the message and 'A' is originated at the KDC

The message includes 2 items for A:

iii) The session key K_s

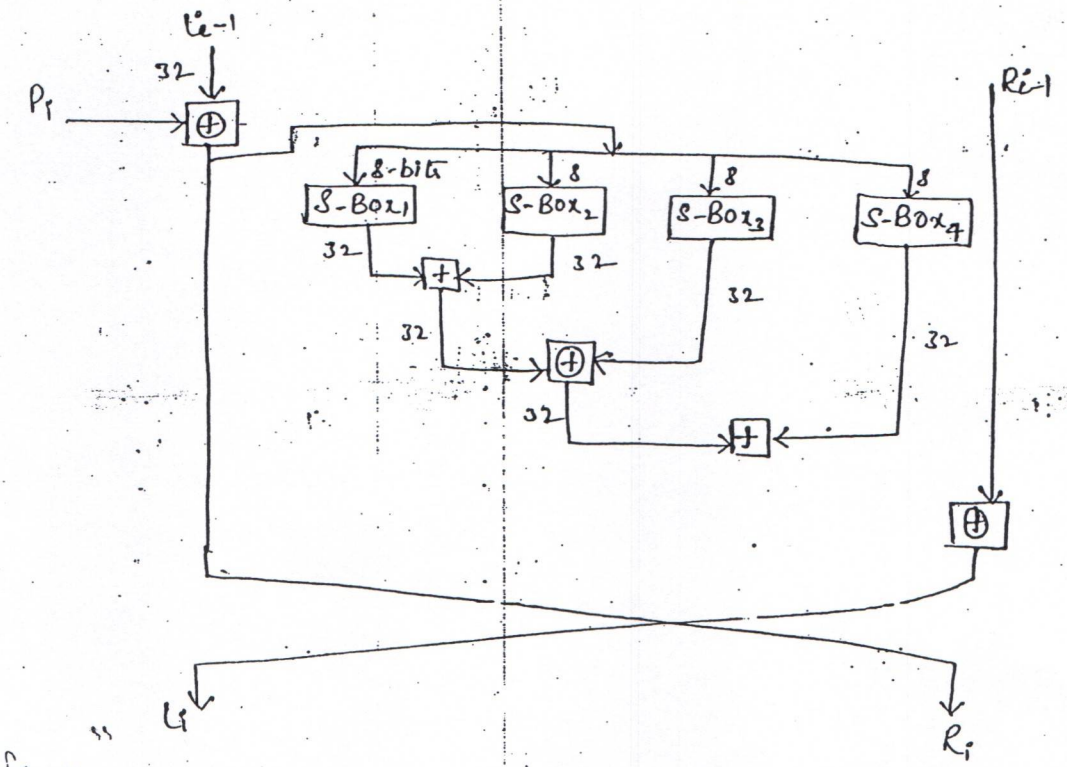
iv) The Original request message, including the nonce to enable A to match this response with the appropriate request.

A to match this response with the appropriate request, addition the message includes 2 items for B.

i) the session key K_s

ii) An identifier of A i.e. (ID_A)

These 2 items are encrypted with the master key k_b



→ If the 4 bytes are named as a, b, c, d then the function of F can be written as,

$$F(a,b,c,d) = \left[(S_{1,a} + S_{2,b}) \oplus S_{3,c}\right] + S_{4,d}.$$

where 4 bytes are named as a, b, c, d.

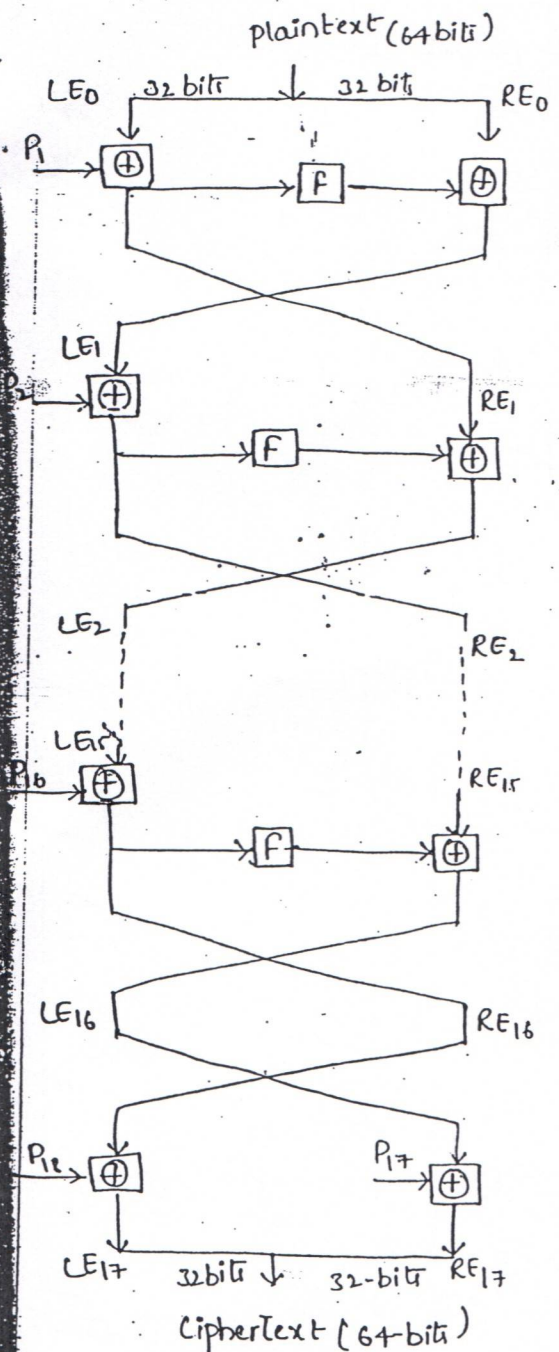
Advantages:

Fast: Blowfish encrypts data on 32-bit microprocessors at a rate of clock cycles per byte

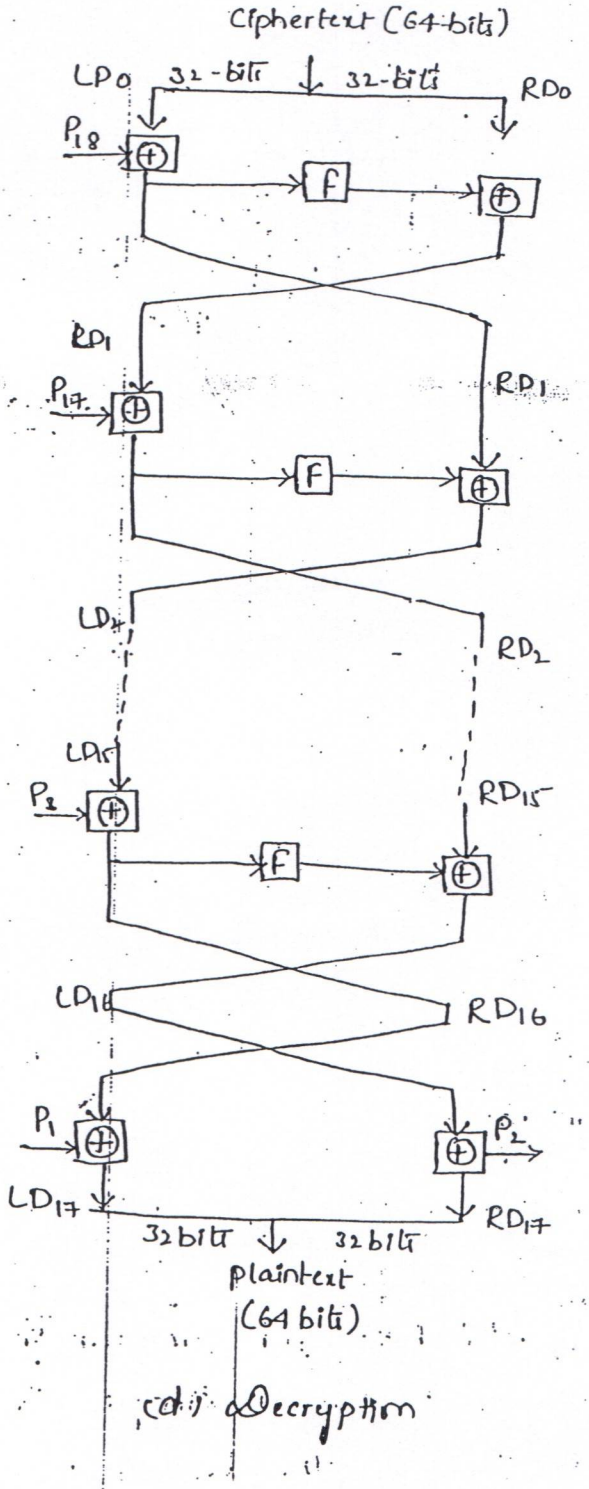
Compact: Blowfish can run in less than 5KB of memory.

Simple: It has a simple structure and is easy to implement.

Variably: Since the key length is high it allows a higher Secure Speed and higher Security.



(a) Encryption.



(b) Decryption

F- Function:

→ The 32-bit input to f is divided into 4 bytes.

→ These 4 bytes are send to the 4- S-boxes and the function is explained in the following figuare.

UNIT - II

BLOW FISH:

→ Blow-fish is a Symmetric block cipher encryption technique developed by Bruce schneier.

→ Blowfish encrypts 64-bit blocks of plaintext into 64-bit block of Cipher text.

Subkey & S-Box generations:

→ Blowfish makes use of a key that ranges from 32 bits to 448 bits [1-14 words i.e; 1 word = 32 bits, 1 byte = 8 bits]

→ This key is used to generate 18 sub-keys each consisting of 32-bits and 4-s-boxes each contains 256 entries.

→ The keys are stored in a K-array i.e; $K_1, K_2, \dots K_{14}$.

→ The sub-keys are stored in a p-array namely $P_1, P_2, \dots P_{18}$ each contains 32-bits.

→ The 4 S-boxes are namely.

$S_{1,0}, S_{1,1} \dots S_{1,255}$

$S_{2,0}, S_{2,1} \dots S_{2,255}$

$S_{3,0}, S_{3,1} \dots S_{3,255}$.

$S_{4,0}, S_{4,1} \dots S_{4,255}$

→ Here each s-box entry will have 32-bits of information.

Encryption:

→ First the plaintext is divided into two halves each contains 32-bits and named as LE_0 & RE_0

→ Left plain text is XORed with the 1st subkey P_1 and is send to the 'f' function

→ The result of this 'f' function is XORed with the right plain text which taken as the new LPT.

→ And the input to the 'f'-function is now taken as the new right plaintext for the next round.

→ We have 16 rounds in this algorithm

→ In the final step RE_{16} is XORed with P_{18} to Produce LE_{17} & LE_{16} is XORed with P_{17} to produce RE_{17} which are then combined to form the final Ciphertext

Output Feed Back (OFB):

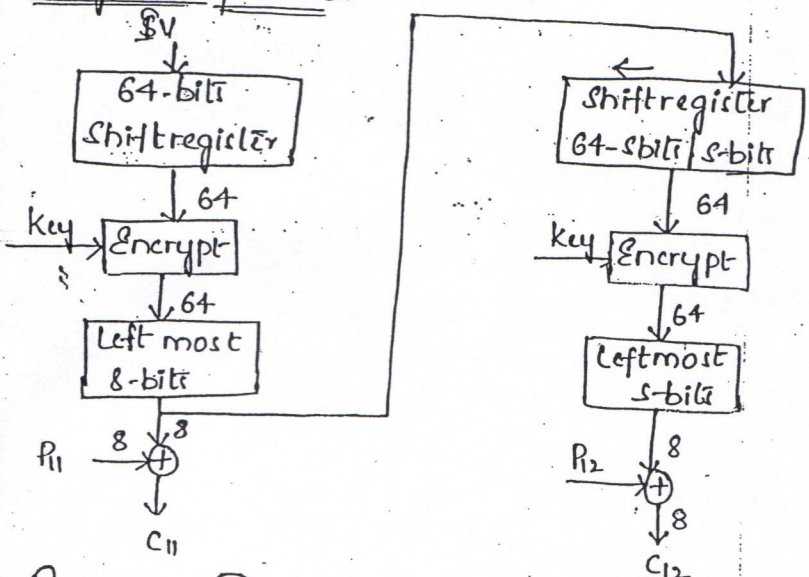
→ This is similar to that of CFB mode

→ Only the difference is we will take the o/p of encryption function to the next register as the input.

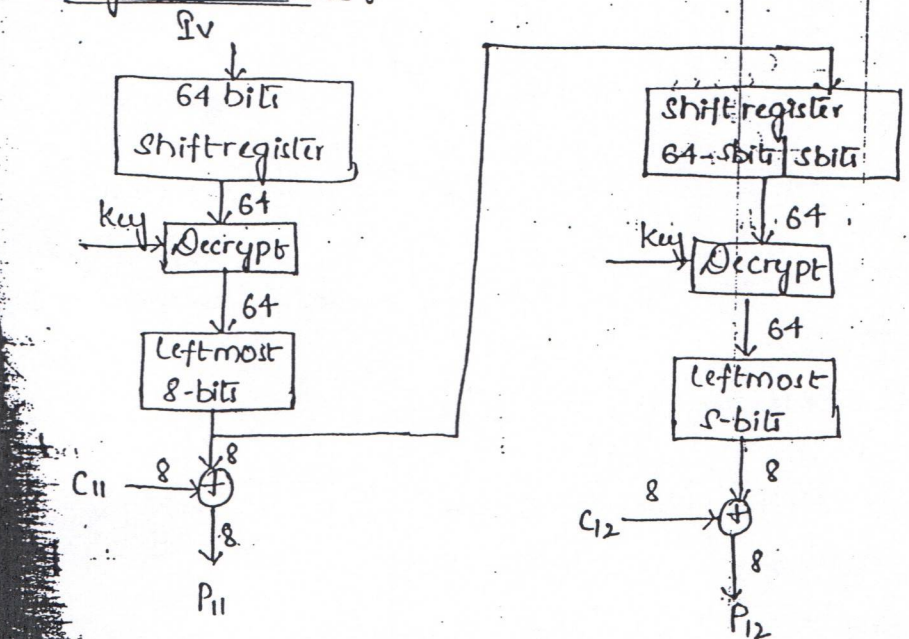
→ One advantage of the OFB method is that bit errors do not occur.

→ This means if a bit error occurs in C_i, only the recovered value of P_i is affected, Subsequent plaintext units are not corrupted

Encryption process:



Decryption Process:



Cipher Feed Back : (CFB)

→ In this case the 64-bits of plaintext is divided into segments of 8 bits.

→ Normally 1 segment has 8-bits.

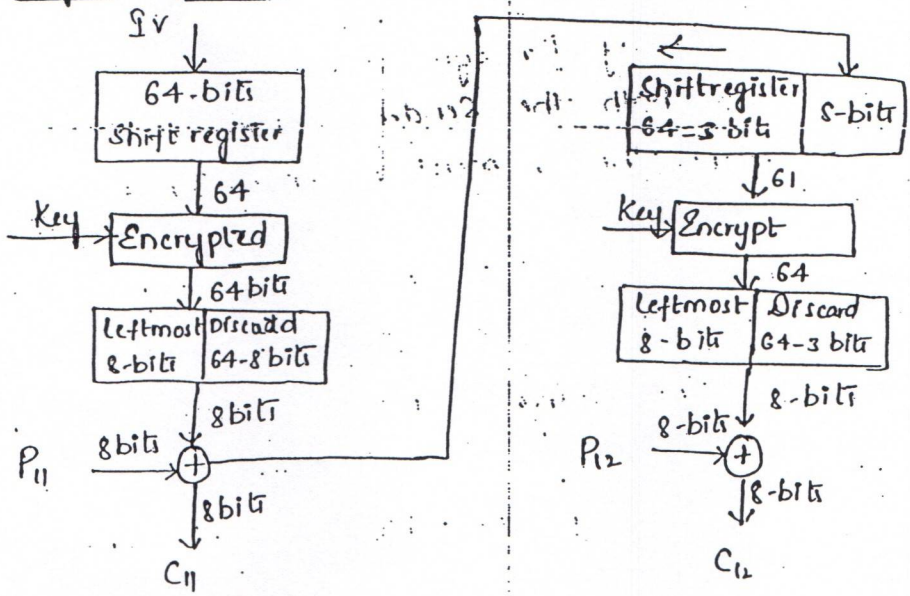
→ In the encryption, we will pass 64-bits shift register that is initially set to some initialization vector (I.V.)

→ The leftmost 8-bits of the output of the encryption function are XORed with the 1st segment of plaintext P_1 to produce the 1st segment of cipher segment text C_1 which is then transmitted to the next segment's encryption.

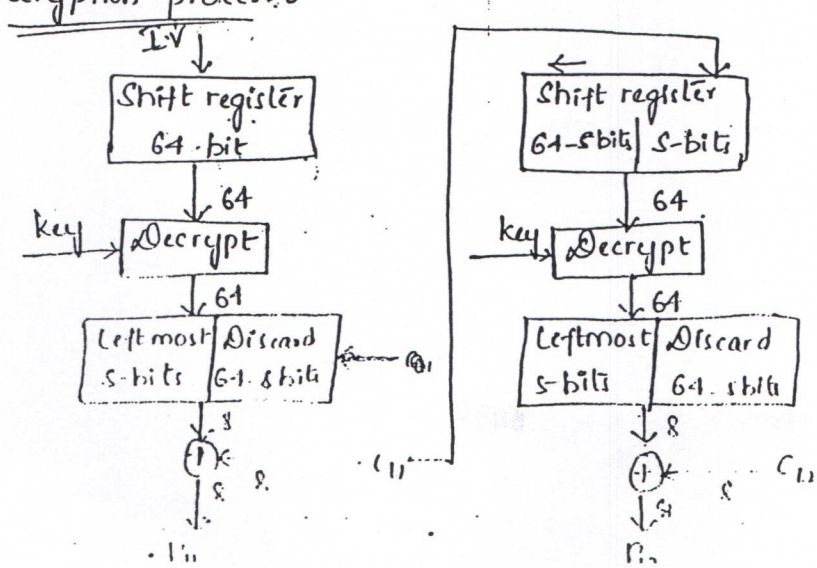
→ This process continues until all the segments of plaintext P_1 are Completed.

→ This is shown in the following

Encryption process:



Decryption process:

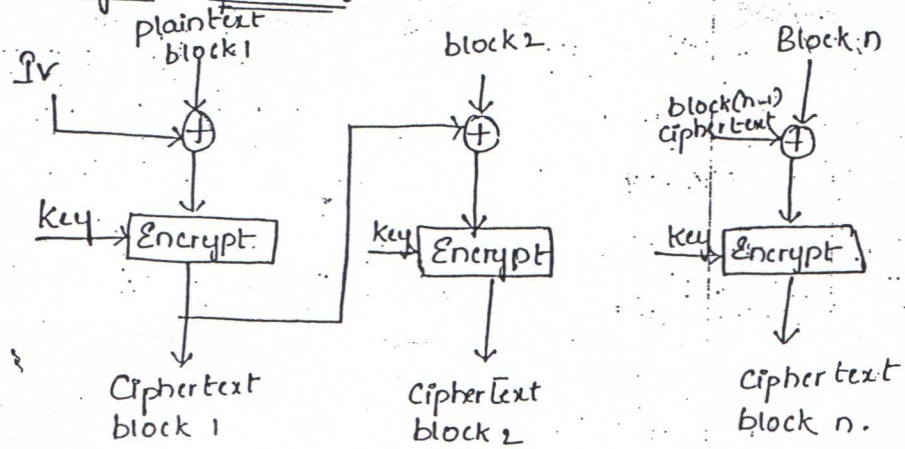


Cipher Block chaining:

→ In this CBC the result of the previous block is given to the current block encryption as the input.

→ Therefore even in the plaintext message contains repeated words it will produce different outputs.

Encryption process:



→ Here IV is a Initialization vector

→ The IV may be a text message / a symbol message.

→ The IV must be known to both the sender & receiver.

→ To make this one secure, the IV must be encrypted using BCB mode.

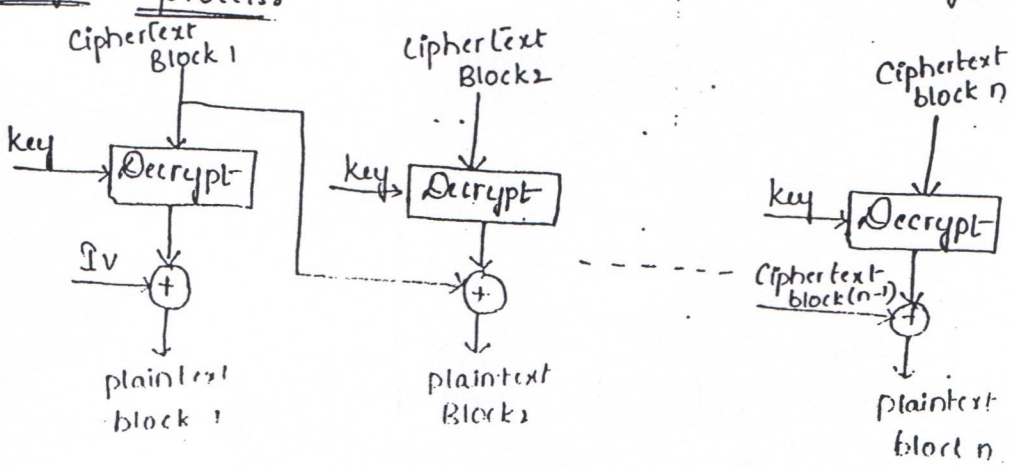
→ In the 1st step the XOR operation on the IV & the plaintext Block

→ Then encrypt the result with a secret key.

→ The result produced at this stage is taken as output of that plaintext block

→ Now this result is send to the next block encryption process.

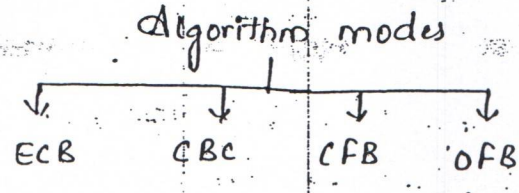
Decryption process:



(*) Block Cipher Design Algorithm modes :

→ There are 4 important Algorithm modes namely :

(i) Electronic code Book (ECB)
(ii) Cipher Block chaining (CBC)
(iii) Cipher Feed Back (CFB)
(iv) Output Feed Back (OFB)

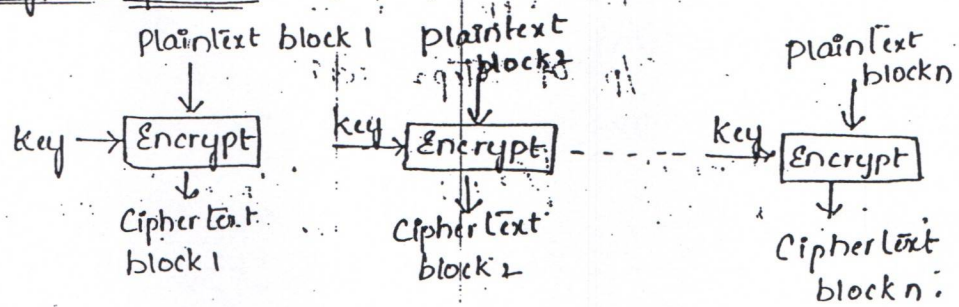


(1) Electronic code Book:

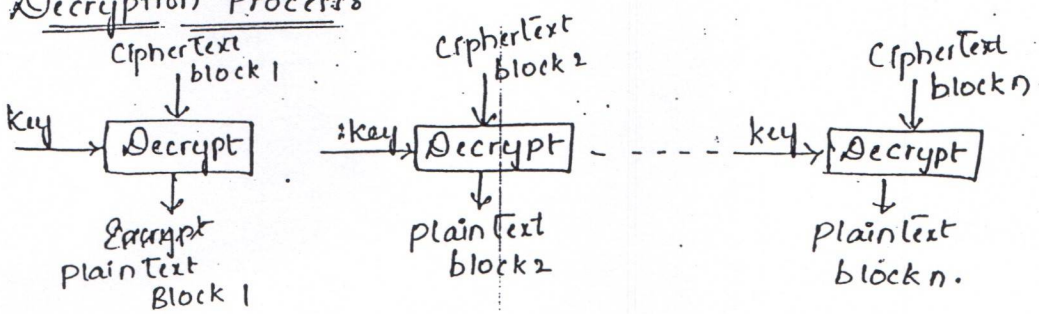
→ Here, the incoming plaintext message is divided into blocks of 64 bits each.

→ Each block is encrypted with the same key independently.

Encryption process:



Decryption Process:

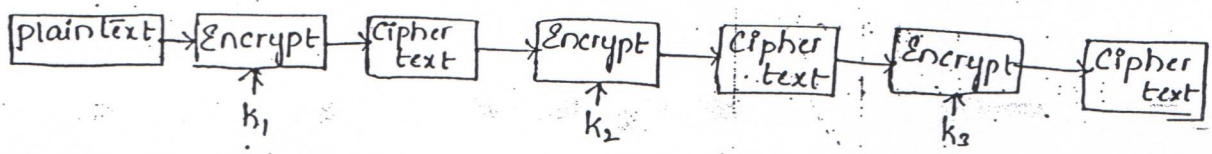


→ Since a single key is used for encrypting all the Blocks of a message. If a plain text block repeat in the Original message, the corresponding cipher text block will also repeat in the encrypted message

→ Therefore ECB is suitable for encrypting small messages, where the scope for repeating the same plain text blocks is quite less.

Triple DES with three keys:

→ In this the plaintext is 1st encrypted with a key k_1, then encrypted with key k_2 and finally with a third key, k_3 where k_1, k_2, k_3 are all different from each other.



Triple DES with 2 keys:

→ In triple DES with three keys we need 3 different keys that means we require 168 bits for the key.

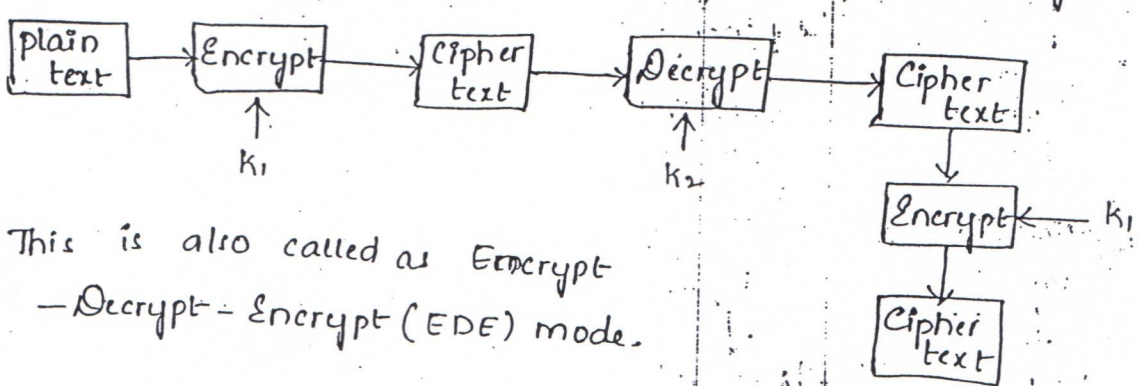
→ To reduce the size of the key, Tuchman suggested the Triple DES with two keys.

→ The algorithm works as follows:

(i) First encrypt the Original plain text with the key k_1

(ii) Decrypt the O/p of first step with the key k_2.

(iii) Finally, encrypt the O/p of step 2 with the key k_1.



→ This is also called as Encrypt – Decrypt – Encrypt (EDE) mode.

DES Decryption:

→ The same algorithm which is used in the encryption is also used for decryption

→ The only difference b/w the encryption & decryption process is the reversal of key portions.

→ If the Original key K is divided into $k_1, k_2, \ldots k_{16}$ for the 16 encryption rounds then for decryption the key should be used as $k_{16}, k_{15}, \ldots k_1$.

Variations of DES:

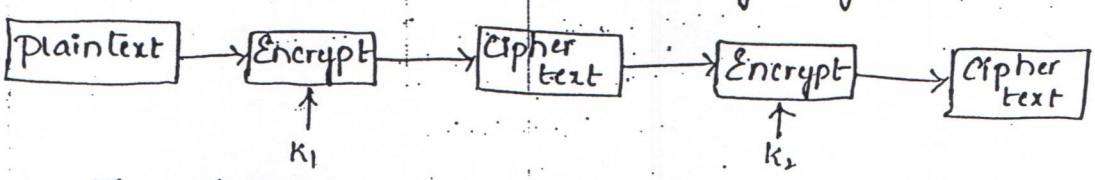
→ It is also easy to understand.

→ It uses 2 keys say k_1 & k_2

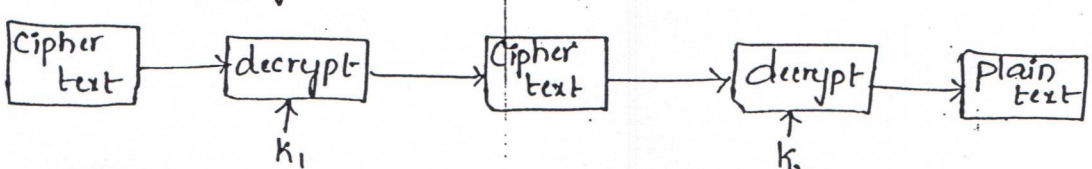
→ It 1st performs DES on the Original plain text using k_1 to get encrypted text.

→ It again performs DES on the encrypt text using another key k_2 to get the final Output.

→ This is shown in the following figuare:



→ The decryption can also be done in the same way but it uses the keys in reverse order.



→ The encryption process mathematically represented in the following manner.

P = plaintext, T = Temporary Result

C = Cipher text, K_1 = key 1 ; K_2 = key 2.

$$T = E_{K_1}(P)$$

$$C = E_{K_2}(T) = E_{K_2}(E_{K_1}(P))$$

→ The decryption process can be represented as

$$T = D_{K_2}(C)$$

$$P = D_{K_1}(T) = D_{K_1}(D_{K_2}(C))$$

Step-4: P-Box permutation:

→ Now the Output of S-box consists of 32-bits

→ These 32-bits are permuted using straight forward permutation mechanism i.e; replacement of each bit with another bit.

→ This is called as P-Box permutation.

Step-5: P-b XOR and Swap:

→ we have been performing all there operations only on the RPT of the 64-bits Original plaintext.

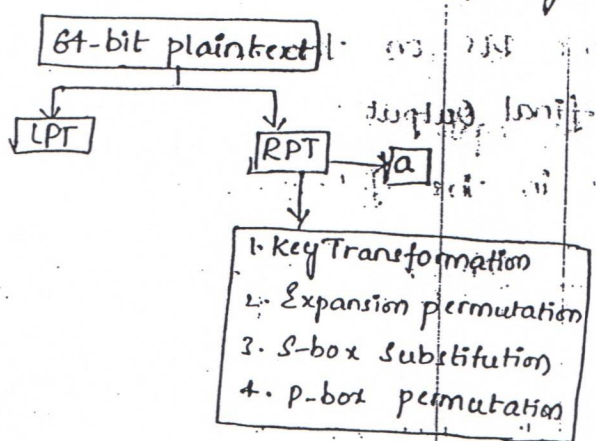
→ The LPT was untouched so far.

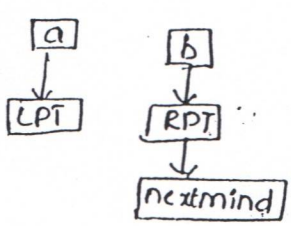
→ At this step this LPT is XORed with the Output produced by the P-box permutation.

→ The result of this XOR Operation becomes the new RPT

→ The old RPT now becomes the new LPT

→ This process is known as Swapping.





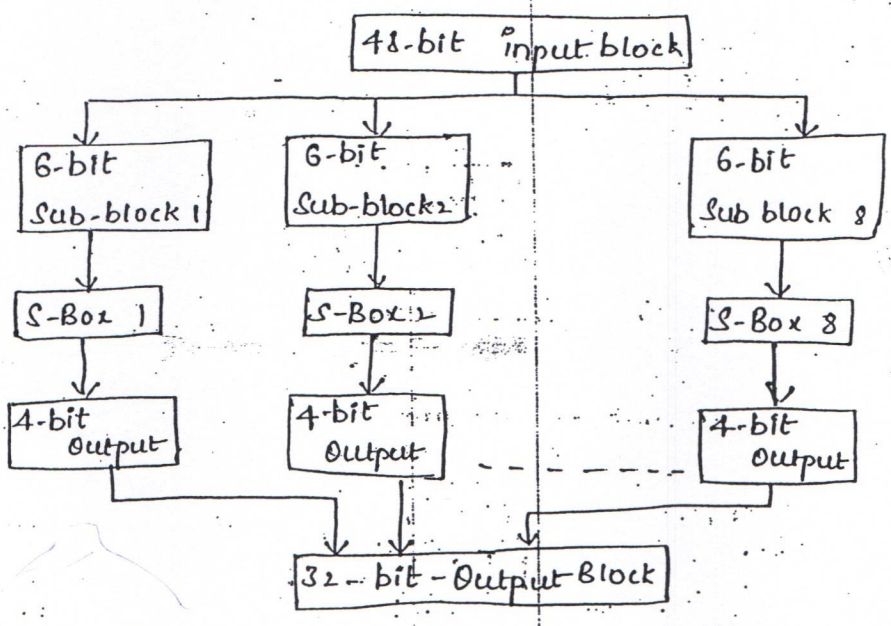
Final permutation:

→ At the end of 16 rounds, the final permutation is performed only one.

→ This is a simple transposition process.

→ At this stage we will get the 64-bit ciphertext as the o/p

→ Now the S-Box transforms the 6-bit input into a 4-bit Output



→ The logic involved in s-box is, it consists of 4 rows namely 0, 1, 2, 3 & 16 columns namely 0, 1, ----- 15.

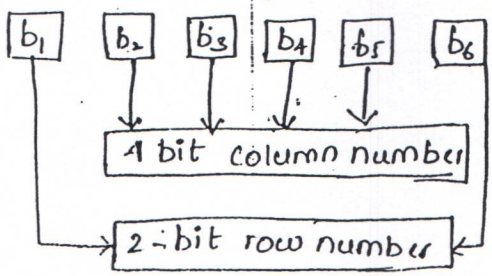
→ The intersection every row & column consists a 4-bit number which is the Output for each s-box.

→ The 6-bit input indicates which row and which column and therefore which intersection element is to be selected which is the 4-bit Output.

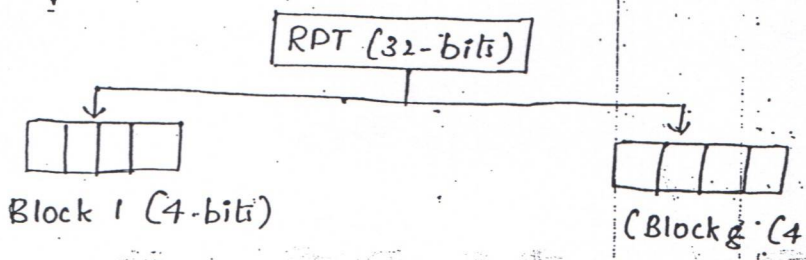
→ let us assume that the 6-bits of a s-box are b_1, b_2, b_3, b_4, b_5 and b_6.

→ Now the bits b_1 and b_6 are Combined to form a two-bit row number lines which lies in b/w 00 → 0 to 11 → 3.

→ The remaining 4 bits b_2, b_3, b_4, b_5 are combined to form a 4-bit column which lies in b/w 0000 → 0 to 1111 → 16

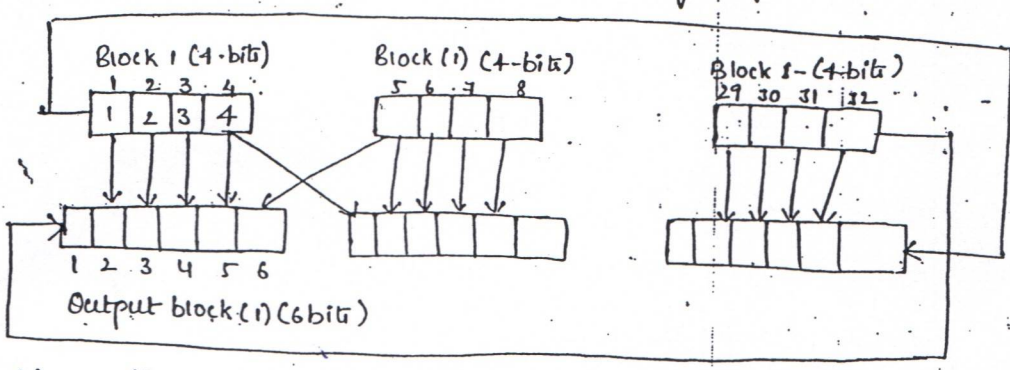


→ First the 32 bit RPT is divided into 8 blocks with each consisting of 4 bits.

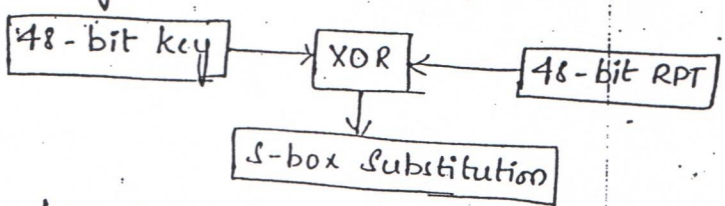


→ Next, each 4-bit block of the above step is expanded to corresponding 6-bit block

→ This is shown in the following figuare.



→ Now the 48-bit key is XORed with the 48-bit RPT, and the resulting Output is given to the next step which is the S-Box Substitution.



→ The following values show the XOR logic

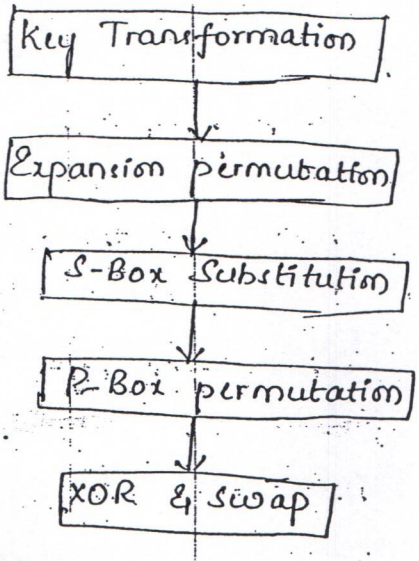
Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Step-3: S-Box Substitution:

→ This substitution is performed by 8 substitution boxes also called as S-Boxes.

→ Each of the 8 s-boxes has a 6-bit Output and produces 4-bit Output

→ The 48-bit input block is divided into 8 Subblocks each consisting of 6 bits; each sub block is given to a S-box.



Step-1 : key Transformation :

→ For each round a 56-bit is require box.

→ from this 56-bit key a different 48-bit subkey is generated during each round using a procus called as a keytransformation.

→ forthis, the 56-bit key is divided into two ~~left~~halves each of 28-bits.

→ These halves are circularly shifted left by 1 or) 2 but positions depending upon the round.

→ for eg; if the round number is 1, 2, 9 or) 16, the shift is done by only one position.

→ for other rounds, the circular shift is done by 2 positions

→ after an appropriate shift out of 56-bits only 48-bits are selected randomly.

→ Since the key transformation procers involves permutation as well as seleetion, it is called as Compression permutation.

Step-2 : Expansion permutation.

→ After IP, we had two 32-bit plaintext areas called LPT and RPT

→ During expansion permutation, the RPT is expanded from 32 bits to 48 bits.

- Along with the increase of bit size from 32 to 48, a permutation also be done on this 48 bits

- Hence this process is known as Expansion permutation.

→ In the first slip, the 64-bit plaintext block is send to an Initial permutation (IP) function.

→ In the second step the initial permutation is performed on the plain text

→ Now the IP produces two halves of the permuted block; say left plain text (LPT) and right plain Text (RPT)

→ In the end Now each of the LPT & RPT are rejoined, and a final Permutation (FP) is performed each with its own key.

→ In the end, Now each of LPT and RPT goes through 16 rounds of encryption process each with its own key.

→ The result of this process produces 64-bit Cipher text.

Initial permutation:

→ Here it uses the transposition technique i.e; rearrangement of the plaintext letters.

Eg:

Bit position in the plaintext	After IP function
1	58
2	50
⋮	⋮
64	7

→ which means after the IP function the first bit of the plaintext is moved to the 58th location, and so on

→ Division of Blocks:

→ Now the resulting 64-bit permuted text block is divided into two half blocks.

→ Each half block consists of 32 bits.

→ we call the left block as LPT and right block as RPT

Rounds:

→ Each of 16 rounds consists of five steps which shown in the following figure.

Data Encryption Standard : (DES)

→ This is also known as Data Encryption Algorithm (DEA)

→ This is a Block cipher Encryption.

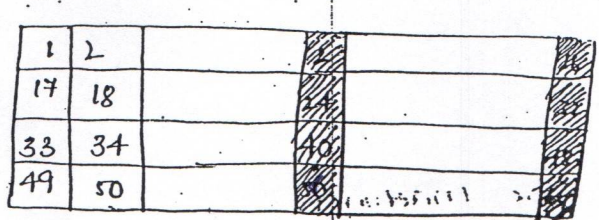
→ It takes 64 bits of plaintext as input which produces the 64 bits of ciphertext as Output.

→ It takes another 64 bits for key which used both for encryption & Decryption.

→ Before the DES process every 8th bit of the key bits are discoded to produce 56 bits.

→ i.e; the bit positions 8, 16, 24, 32, 40, 48, 56 and 64 are discarded.

Eg:

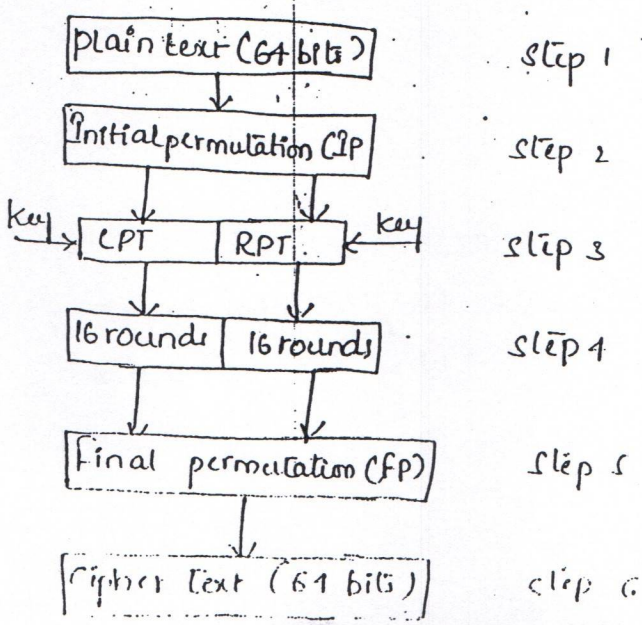


→ The shaded bits are discarded here to produce a 56-bit key which is substituted in DES algorithm.

→ DES Constructs of 16 steps each of which is called as a round.

→ Each round performs the steps of Substitution and Transportion

→ The following figure shows the broad level steps in DES.



→ This is also easy to Cryptanalyst to decade the cipher text by putting it in a matrix order

→ so to avoid this we again rearrange cipher text in a matrix and we give the same order value.

→ This is called a double Transposition

Eg: Key: 4 3 1 2 5

Input: e f h a y
t t e p a
t q t m e
r o r m e
t a

Output: HETRAPMHFTQOAETTRTYAEE

Rotor Machines:

→ The basic principle of the rotor machine is to pass the electrical pulses among the machines.

→ The machine consists of a set of independently rotating cylinders through which electrical pulses can flow

→ Each cylinder has 26 input pins & 26 Output pins, with internal wiring that connects each input pin to a unique Output pin.

→ Consider a machine with a single cylinder

→ After each input key is depressed, the cylinder rotates one position, so that the internal connections are shifted accordingly

→ The power of the rotor machine is in the use of multiple cylinders, in which the Output pins of one cylinder are connected to the input pins of the next.

→ To encrypt a message, a key is needed that is as long as the message

→ usually the key is a repeating keyword.

→ In vigenere table the italic lowercase letters represent the keyword letters, the lower case letters represent the plaintext letters & the upper case letters represent the ciphertext letters.

Transportation Techniques:

→ There are 2 techniques under this concept

1) Rail fence technique &
2) Rotor Machines.

Rail Fence Machines:

→ In this technique the plain text is written down as a sequence of diagonals & then read off as a sequence of rows.

Eg: plain text: meet me after the toga party

m e m a t r h t g p r y
e t e f e t e o a a t

Cipher text: MEMATRHTGPRYETEFETEOAAT

→ Another method is arrange the plaintext letters in a rectangle, row by row and read the message off column by column.

→ And give the numbers to the columns in any order.

→ The order of the columns then becomes the key to the algorithm.

Eg: plaintext: meet me after the toga party.

key:	4	3	1	2	5
	m	e	e	t	m
	e	a	f	t	e
	r	h	h	e	t
	o	g	a	p	a
	r	t	y		

Cipher text: EFHYTEPEAHGTHEKOKHCTA

1. plain text letters are fall in the same row of the matrix we each separated by the letter to the right.

Eg: - is encrypted as RM

2. plain text letters are fall in the same column of the matrix are each replaced by the beneath letter.

Eg: CU is encrypted as EH.

3. Other wise, each plain text letter is replaced by the letter that lies in its own row & the column occupied by the other plain text letter.

Eg: hs becomes BP

ea becomes IH / JH.

4. Repeating plaintext letters that would fall in the same pair are seperated with a filler letter namely x.

Eg: Ballon becomes ba|x|loon

This can be encrypted as IBSUIHNA / JBSUPHNA

Monoalphabetic Ciphers:

→ This can be achieved by increasing the key space and an arbitrary substitution for the plain text.

→ To encode the plainText letters individually we have 26! Possible keys.

→ so, that it would be very difficult for the cryptanalyst

⇒ still there is a scope for cryptanalysis if the cipher text is known.

→ If the cryptanalyst knows the nature of the plaintext, then the analyst can easily extract the regularities of language.

→ As a 1st step, the relative frequency of letters can be determined & compared to a standard frequency distribution for English.

Poly alphabetic Ciphers:

→ we encrypt the plain text in this technique with a k
of vigenere table

Plain Text: 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

Cipher Text: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Example:

plain text: meet me after the toga party

Ciphertext: PHHW PH DIWHU WKH WRJD SDUWB.

→ In this technique, it is very easy to construct the cipher text so that the cryptanalyst can easily trace out the plain text.

→ The process of Break down the cipher text to find the relative plaintext is known as Brute-force Cryptanalysis.

→ Here the Cryptanalyst replace each letter of the cipher text with other letter to find the plaintext.

→ within 25 attempts they can easily trace out the plain Text.

Play fair cipher:

→ This algorithm is based on the use of a 5x5 matrix of letters constructed using keyword.

→ Eg: Here the keyword is Monarchy.

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

→ First we have to arrange the keyword in the 5x5 matrix from left to right and top to bottom excluding the repealed letters.

→ Now arrange the remaining letters of the alphabet treating I and J as one element in the 5x5 matrix.

plain text is encrypted 2 letters at a time according to the following rules:

→ A simple form of steganography is one in which the sequence of first letters of each word of the overall message spells out the hidden message.

→ Various other techniques have been used.

1. Character Marking: Selected letters are printed / typewritten text are overwritten in pencil. The marks are ordinary not visible unless the paper is held at an angle to bright light.
2. Invisible ink: Using some invisible ink the original message can be hidden. When heat / some chemical is applied on that paper then only it will be visible.
3. Pin punctures: Small pin punctures on selected message can be done & that letters are visible only under a strong light.

→ One drawback of steganography is time consuming to construct.

Classical Encryption Techniques:

we have 2 methods to encrypt the plain text

1) Substitution Technique
2) Transposition Technique

Substitution Techniques:

→ In this method, the letters of plaintext are replaced by other letters / by numbers / symbols.

→ There are 4 types of substitution techniques.

1) Ceaser cipher
2) Mono alphabetic cipher
3) Play fair cipher
4) Polyalphabetic cipher.

Ceaser Cipher:

→ This was discovered by Julius Ceaser

→ It is very easy to understand & construct.

→ In this technique replace the each letter of the plaintext with the letter standing three places further down the letter.

* → Cryptographic Systems are generally classified along three independent dimensions.

1) The type of Operations used for transforming plaintext to Cipher text.
→ All the encryption algorithms are based on 2 general Principles
   i) Substitution
   ii) Transposition.

→ In substitution each element in the plaintext (bit, letter, group of bits/letters) is mapped into another element.
→ In Transposition, elements in the plaintext are rearranged without loss of any information.

2) The no. of keys used
→ If both the Sender & receiver use the same key the System is reffered to as asymmetric, two-key, (or) public-key encryption.
→ If both the Sender & receiver each uses a different key, the system is referred to as symmetric (or) single key, secret key (or) conventional encryption.

3) The way in which the plaintext is processed.
→ There are 2 ways to process the plaintext.
→ A Block cipher processes the input one block of elements at a time & produces an output block for each input block.
→ A stream cipher process the input elements continously Producing output one element at a time, as it goes along.

→ A plaintext may be hidden in 2 ways.
   1) Cryptography
   2) Steganography

Steganography:

This method conceal the existance of the original message.

UNIT-1

Network Security model: Cryptography

Terminology:

Plain text: The Original message which is to be transimitted between source and the destination.

Encryption: The process of converting the Original message into a coded form.

Cipher text: The message which is in the encoded format is known a cipher text.

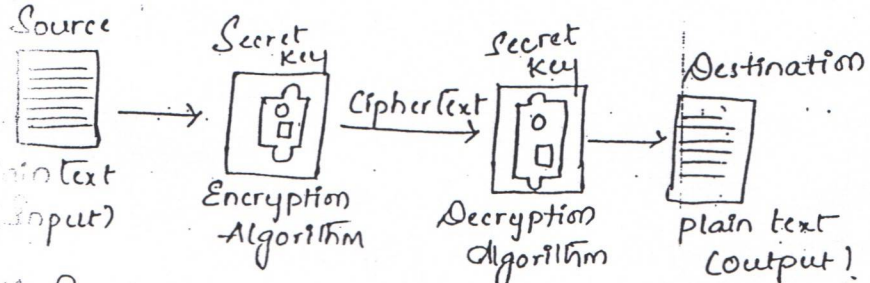
Decryption: The process of converting the encoded message into Original format.

Cryptography: The study of encryption and decryption is known as cryptography.

Cryptanalysis: The process of attempting to discover the plaintext and the keys un authorizidly is known as cryptanalysis.

Conventional Encryption Model: (Pascal Encryption model)

→ This is also known as symmetric encryption (or) Single-key encryption.



Step-1: First we have to select the plain text which is to be transmitted from source to destination.

Step-2: Now the plaintext is converted into cipher text using some encryption algorithm along with a secret key shared by the both sender & the receiver.

Step-3: The ciphertext is converted into plaintext using same encryption algorithm which is reverse of encryption algorithm with the same key which is used in the

Network

Security