4th Aug, 2025

Information and Network Security

Cryptography and Network security

→ Forouzan → William Stallings

→ Atul Kahate

- plain text —encryption→ cipher text

→ Error control: Data Link Layer, Transport Layer

→ Confidentiality: No data breach, no unauthorized Access

→ Integrity: Data should reach in entirety + No alteration midway

→ Availability: Availability of data at the time it is required

→ Authentication: Digital signature

→ Non-Repudiation: Trusted 3rd party presides over data transfer

② SSL: Secure socket layer: placed b/w Application and Transport layer.

→ protocol used to connect web Browser and web server → connection oriented protocol

→ placed b/w Application & Transport Layer as headers are added at Transport, Network & DLL (port No., IP Address, MAC Address) → so SSL should not encrypt these Addresses.

Cryptography:

- Symmetric (same key is used for encryption & decryption)
- Asymmetric: each user has a pair of keys
  - Public
  - Pvt.

Page No.
Date

→ public key: can be known to other users as well. Both sender & receiver are aware of each other's public key.

→ Sender encrypts plain text into cipher with its private key. Receiver diciphers using sender's public key → used in Digital signature { not secure, as public key may also be known to other users }

Compiler Design

7/8/24

INS

Modular Arithmetic

→ In modular arithmetics, both i/p & o/p belong to same set (0 to n-1) for Z_n

- ~~Additive~~ Additive Inverse : Mod - 0
- multiplicative Inverse : Mod - 1
- Residue : Result obtained after modulo op.
- for any matrix A to become a key → it's multiplicative Inverse must exist ⟹ it's det. A ≠ 0 & det(A) & n must be co-prime, i.e.

$$gcd(det(A), n) = 1$$

where n = No. of symbols

- cipher Text = AB → Key
  (plain text)
- → All the op's involved are modular ⇒ (a+b) % n
- Ciphers → Stream : char by char
  → Block : eg. modular { a Block of chars. }

EJOTY
4 9 14 19

Substitution cipher

→ monoalphabetic → polyalphabetic

* **Substitution**: char replaced by symbol that may or may not be present in plain text

— **monoalphabetic**: ONE to ONE mapping. A char only replaced by 1 particular same char.

— **polyalphabetic**: ONE to MANY.
multiple chars can be used to replace a char.
A key stream is generated.

Monoalphabetic ciphers

① **Additive cipher**: eg. caesar cipher (K=3)
each symbol in plain text is converted to int; a key is agreed upon → Added during encryption, subt. during decryption. It is a stream ciphe

eg: key = 15, data = cryptography

c → (2+15) %.26 = 17 → R
R → (17+15) %.26 = 6 → G
y → (24+15) %.26 = 13 → N
P → (15+15) %.26 = 4 → E
T → (19+15) %.26 = 8 → I
O → (14+15) %.26 = 3 → D
G → (6+15) %.26 = 21 → V
R → (17+15) %.26 = 6 → G
A → (0+15) %.26 = 15 → P
P → (15+15) %.26 = 4 → E
H → (7+15) %.26 = 22 → W
Y → (24+15) %.26 = 13 → N

39
34
32
39

2 RST 19

7x15 = 105 % 26

Page No.
Date

* No. of keys possible in Z_{26} = 0 to n-1
  0 to 25 = 26 keys
* Key Domain = 26 (No. of keys) Z_n

(2) Multiplicative Inv cipher

Let K=7 → as gcd(7,26) = 1 ✓ eligible key

For decryption key (7 x x) % 26 = 1
→ 15

7x15 = 105 % 26 = 1 ✓

→ Eligible keys in Z_{26} (key Domain)

key	Inverse
0	x
1	1
2	x
3	9
4	gcd ≠ 1
5	21
6	x
7	15
8	x
9	3
10	x
11	19
12	x
13	x
14	x
15	7
16	x
17	23
18	x
19	11
20	x
21	5
22	x
23	17
24	x
25	25
26	x

→ Find multiples of n (=26)
→ check if key can be multiplied with a no. in range [0 to n-1] to obtain (multiple (26) + 1)

0, 26, 52, 78, 104

Page No. _____
Date

* as key domain of Additive < Multiplicative → Additive cipher is stronger.

(3) Affine cipher : combination of Additive + multiplicative ciphers.

$T \leftarrow P \times K_1$

$C \leftarrow T + K_2$

$P \leftarrow T \times K_1^{-1}$

$T \leftarrow C - K_2$

Key domain = 26×12 (strongest among 3)

→ Bias cannot be controlled by increasing dataset.

Regularization
↳ penalize large coeff. values

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\|w\|^2$$

(magnitude of vector)

↳ To achieve better fitting

(vector of coeffs $w_0, w_1, w_2 \ldots$)

1/8/25 INS

* Transpositional cipher → permutation / arrangement of chars. change.

(4) monoalphabetic substitution cipher → Each char. is to be replaced by a fixed char.

A B ---- Z

↓

$26 \times 25 \times \cdots \times 1 \Rightarrow (26!)$ → key domain (strongest)

Polyalphabetic substitution ciphers (ONE to MANY)
↳ depends on ① plain text char. ② position of char.

(i) Auto key cipher

- 1st subkey → one agreed upon by senders & receiver (predetermined)
- 2nd subkey → Int corresponding to first char of plain text.
- 3rd subkey → Int corresponding to 2nd char of plain Text

pre-determined by sender & receiver

Page No.
Date

Plain Text → INFORMATION

8 13

Keys → 3 8 13

cipher → 11 21 ---

(L V ----)

(iii) Playfair cipher

↳ key is predecided by sender & receiver in form of 5×5 matrix (I & J share same cell.)

U	T	F	C	A
V	S	E	D	B
W	P	Q	G	H
X	R	M	L	I/J
Y	K	N	O	Z

→ Any random combination

→ If there are any 2 consecutive same chars. → Insert a bogus char. in b/w

→ Finally length must be even → if not then add a bogus char. at end.

→ Process the text in pairs.

→ For each pair of chars in Plain Text:

① They may be in same row in matrix → Then they are replaced by next char. in that row (circular)

② They may be in same column → Replaced by 1 char. down in same column

③ They are in diff row & col → Each char. replaced by char lying on intersection of these chars & in same row as char. to be replaced.

eg: GH → HW

SQ → PR

AD → LS

(iii) Vignere cipher

Sender & Receiver agree upon a subkey word

→ Does not depend upon position of chars.

INFORMATION

subkey ← HELLOHELLOH

(iv) Hill cipher

→ Block cipher

→ Key used → A square matrix (having an inverse) of size mxm

↳ $|A| \neq 0$

size of block to be encrypted

→ Plain Text taken in a matrix of size $l \times m$ (l = No. of Blocks in plain Text)

$$\underset{l\times m}{[\ ]} \times \underset{m\times m}{[\ ]} = \underset{l\times m}{[\ ]}$$

plain Text × key = cipher Text

→ Find determinant of key matrix → If its multiplicative Inverse exists in Z_{26} → it is ~~an~~ a valid key

18/8/25

Transposition ciphers

key ① **Railfence Technique** : Arrange alternate chars. in zigzag manner & then write row by row.

eg: INFORMATION

① → I F R A I N

② → N O M T O

⇒ IFRAINNOMTO

key ② **Columnar Technique** : Fix no. of columns & place text row by row. eg. let no. of columns (pre-agreed) = 4

↓ ↓ ↓ ↓

I N F O

R M A T

I O N

⇒ IRINMOFANOT

1st col

* key → an element getting used at both ends to cipher / decipher.

(3) (1) with key → encryption ↓ ↑ decryption

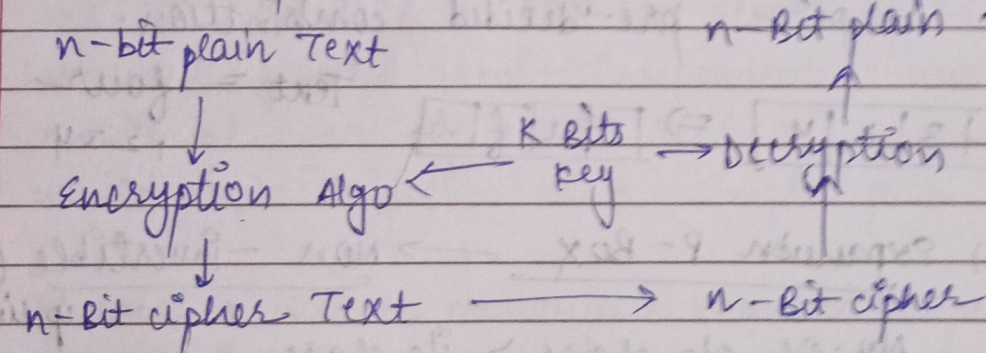
1	2	3	4	5
3	2	1	5	4

A Block size chosen & their internal permutation is pre-agreed. (if no. of chars < block size → Add padding/extra/Bogus char)

(2) No. of columns fixed → Text written row-wise and then during encryption → permutation/order of columns decided by key. (Key size = No. of columns)

Modern Ciphers

→ Bit oriented instead of char oriented
→ Block ciphers



Ques: Total No. of char to be encrypted = 100 & using ASCII coding of size 8 & Block size = 64 bits, calculate No. of padding chars.

→ No. of bits for each char = 8
Total Bits = 8 × 100 = 800
No. of padding = 800 % 64 = 32
Total No. of Bits = 800 + 32 = 832

No. of times iteration need = 832/64 = 13 on one side

→ In Bit oriented transposition ciphers → No. of 0's & 1's remain same → Helps adversary in Brute force attack.

Components of modern ciphers

(1) Permutation - Boxes (P Boxes)

↳ (i) straight P-Box → Invertible

→ Same no. of i/p & o/p lines

→ Interconnections are intermixed in hardware

1	2	3	4
2	3	1	4

→ In software : Take an array of size n and directly write i/p chars. as per pre-decided combination.

2	3	1	4

⇒

o	u	f	r

Text = four
1 2 3 4

(ii) Expansion P-Box → Non-invertible (when i/p lines ≠ o/p lines)

No. of o/p chars > i/p chars.

(iii) Compression P-Box (8x4) (Non invertible)

o/p chars < i/p chars

values in Box → i/p chars

position in Box → o/p chars.

Page No.
Date

INS

→ DES: Data Encryption Standard
→ 64 rounds of Algos are performed on DES
→ Data itself can never be encrypted by any non-invertible component
→ Non-invertible components are used for key Generation
→ symmetric Block cipher

Components of modern cipher

(2) S-BOX Substitution BOX
1) Invertible 2) Non-invertible

→ No. of input bits are made to multiples of 3 bits
eg. 10101100 → padding

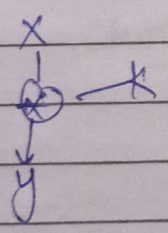
	col 00	01	10	11
0	01	00	10	11
1 (ROW)	11	01	00	10

Now in 3 bit No.

ROW col

eg. 101 → 1 01 ⇒ 01
011 ⇒ 0 11 ⇒ 11
000 ⇒ 0 00 ⇒ 01
Hence, 10101100 ⇒ 011101

(3) X-OR → Invertible



Now,
$y \oplus K \Rightarrow x$

Proof
$y = x \wedge K$
$y \wedge K = x \wedge K \wedge K$
$y \wedge K = x \wedge 0$
$x = y \wedge K$

(4) circular shift
Left shift / Right shift
→ $x << K$
$K \in [0, n-1]$

To make it invertible, just shift k times in decrypting Algo.

(5) Complement (6) Split & combine

(7) Swapper

→ Circular shift where $k = n/2$

→ can be used if No. of Bits are even

→ Divides bits into 2 & swaps

product cipher

Feistal Non-feistal

Diffusion: Extent of complexity shown b/w the relation b/w plain text & cipher Text.

Confusion: Relation b/w cipher text & encryption key

→ product ciphers are Block ciphers wherein multiple Algo / components used for encryption

→ Feistal: Both Types of components (Invertible & Non-invertible) → eg. DES

→ Non-feistal: only invertible components used eg. AES

→ Blocking system call on a thread does not block entire process (other threads can execute parallely)

INS

* Data Encryption Standard (DES)
→ each round composed of swapper & mixer
→ 16 rounds for encryption + 16 decryption
→ uses 56 bit symmetric key ~~initially~~ → each round has different sub key
→ size of sub key = 48 bits (generated by key generator)
→ size of Block = 64 Bits

Ciphering		Deciphering	
Round 1	Mixer swapper	Mixer swapper	Round 16 15
2	Mixer swapper		
3		swapper Mixer	3 2
15	Mixer swapper	swapper	
16	Mixer	Mixer	1

→ Not rounds, but sections of rounds (mixer - mixer, swapper - swapper) are aligned.

→ Plain text is divided into Block sizes of 64 bits

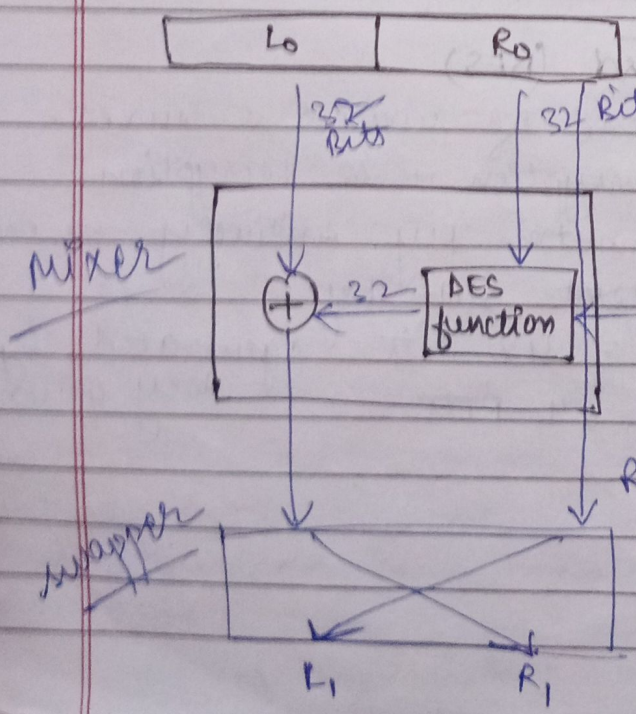
→ straight p-Boxes are used in Initial permutation & Final permutation

↓ Fed to Round 1

→ Size of straight p-Box used = 8 × 8 ~ 64 (1 to 64)

→ swapper: We have 64 bits. Divide into 2 equal halfs & swap.

→ Mixer ← DES function
whitener (X-OR)



→ swapper : Invertible
→ Mixer : self Invertible (effect of Non-Invertible components gets cancelled due to x-OR)

① DES Function

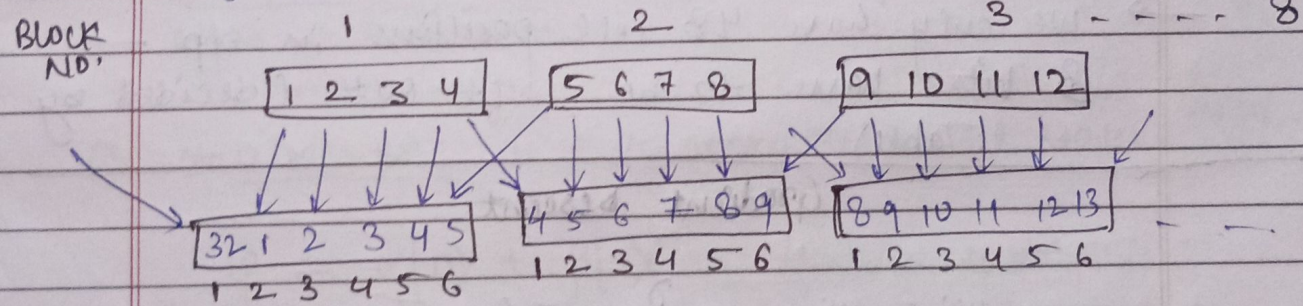
Inputs : ① 32 bits R_0 + ② 48 bit key

4 operations:

① Expansion Key Box
② XOR operation
③ S-Box
④ straight P Box

① Expansion Key Box : converts 32 bits R_0 to 48 bit size = 32 x 48, with a pre-determined logic (repeat certain i/p lines)

→ Divide 32 bits into 8 sub-sections of 4 Bits each. Each 4 bit Block will return 6 Bits



② XOR : XOR of 48 bits of expansion p-Box o/p and 48 bit subkey

③ S-Box : Divide 48 bits into 8 Blocks of 6 Bits each.

Non Invertible

→ DES uses 8 S-Box tables, one for each Block (4 x 16 size each)

→ First & Last Bit will combine to form row no. & b/w 4 Bits = col no.

Assignment : AES Advanced encryption

12 * 16

Page No.
Date

each Block has its own S-Box table (SBox Table 1, 2 ---)

→ only sender & receiver are aware of 56 bit key

* Key Generation → Generate 16 subkeys of 48 bits each

(1) Split key into 28 bits each (28 + 28 = 56)

(2) Apply circular shift (Left) op on Both parts

For Rounds 1, 2, 9, 16 left circular shift by 1 place

Remaining left circular shift by 2 places

(3) combine 2 halves

(4) Apply compression P-Box (56 x 48)

→ we only have 48 cell positions in o/p → 8 bits have to be left out (decided by Table)

· Gradient Descent

$$w_j = w_j - \alpha \frac{\partial}{\partial w} J(w_0, w_1)$$

18

Page No.
Date

To compute follow of symbols

Rule 1: Follow (S) = {$}, if S is start symbol

Rule 2: if there is a production rule of form
$A \rightarrow \alpha B \beta$, where $\beta \neq \epsilon$, it means
then
follow (B) = {first (β)}
= {first (β) − ϵ}
↳ everything included in First (β) except ϵ

Rule 3: for prodⁿ rule of form
$A \rightarrow \alpha B$ or $A \rightarrow \alpha B \beta$, when First($\beta$) contains ϵ, then
follow (B) = {follow (A)}

25

INS

- **RSA**: Rivest Shamir Adleman → Asymmetric cipher
  - → used for all authentication, non-Repudiation, confidentiality
  - → 2 different keys used for encryption/decryption
  - → Receiver always initiates communication → By sending its public key & a number n.
  - → sender then encrypts the text using public key of receiver & number N.
  - → Encrypted text is sent to Receiver, which decrypts it using its own private key
  - → for a system of n nodes → there are total 2n keys
  - → It is difficult to factorize product of 2 large prime no.

Page No.
Date

let P = 7 & Q = 17

N = P x Q = 119

(P-1) x (Q-1) = 6 x 16 = 96

* Eqn to be followed

$$(D \times E) \bmod (P-1)(Q-1) = 1$$

→ choose E such that factor of E & (P-1)(Q-1) do not have any factor in common.

Encryption key

→ factors of (P-1)(Q-1) = 96 = 1 x 2 x 3 x 4 x 6 x 8 x 12 x 24 x 48 x 96

let, E = 5

(D x 5) mod 96 = 1 → multiples of 96 - 96, 192, 288, 384...

as 384 + 1 = 385 (a multiple of 5)

D x 5 = 385 → D = 77

→ Now we need +1 for Remainder = 1

E → public key

D → private key

→ Receiver sends E and N = P x Q to sender

→ cipher text $CT = (PT)^E \bmod N$ → plain Text

→ plain text $PT = (CT)^D \bmod N$

* Digital signature

→ senders initiates the communication

→ used for authentication of sender

→ created using private key of sender & decrypted

by receiver using public key of sender.

(*) when both confidentiality & Authentication needed
→ use digital signature over cipher text.
→ Receiver then deciphers using public key of sender & further decrypts using its own symmetric / asymmetric key.

26/8/25

INS

① Integrity: content of msg. should remain intact, NO modification.

→ In MD5 → NO 2 diff. messages can have same msg Digest.

→ proposed by Rivest; version 5 in use

Message digest: It is a fingerprint of plain Text

→ Block size used in MD5 = 512 bits

→ output generated by MD5 = 128 bits (size of Message Digest)

↳ Total size (Not per Block)

Inputs:

① ABCD Registers: 4 registers of 32 bits each

Total = 128 bits

→ 4 Registers are initialized with some values

→ These contain 128 bit output (Message Digest)

② T-Array : Array of 64 elements having pre-determined values. Each element → 32 Bits

t[1] - - - - t[64]

→ Block size of 512 bits is divided into 16 BLOCKS of 32 bits each. Each Block to be processed in 4 Rounds. Total Rounds = 4X16 = 64 Rounds; Each round requires 1 element of t-array.

Algorithm

→ NO. of i/p bits is 64 bits less than multiple of 512

→ Add padding bits to satisfy the cond^n

→ Add length of original msg in remaining 64 bits

→ M.

⑤ processing of 1 Block

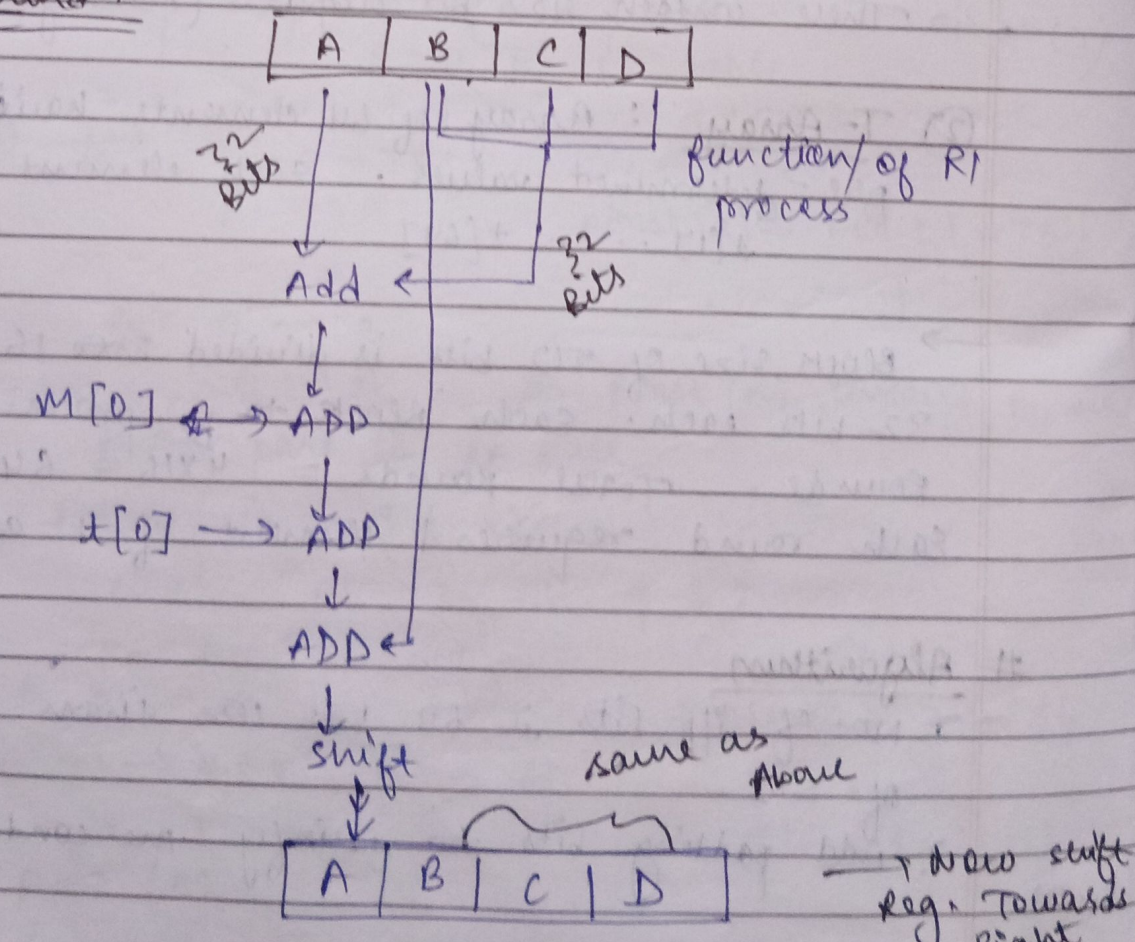
M[0] ---- M[15] — 16 sub-Blocks of 32 bits

→ All 16 sub-Blocks passed via R1, R2, R3, R4 sequentially → 64 iterations Total.

Initialization of ABCD

A	01	23	45	67
B	89	AB	CD	EF
C	FE	DC	BA	98
D	76	54	32	10

Round 1



→ Message digest is one way entity → can't reform / decipher to obtain plain Text Back.

D	A	B	C

→ New ABCD Register for 2nd iteration

→ Ully for each ~~Block~~ sub-Block M[1] --- M[15] perform R1.

→ Then at 17th iteration, → 2nd Round of M[0]
33rd → Round 3
49th → Round 4

27/8/25 ITC

28/8/25

Page No.
Date

INS

→ MAC: Message Authentication Code → shared symmetric key
→ same key is used at both ends → Not
∴ objective is to generate a hash (MAC) → to decrypt → used to check integrity
MAC generated at both ends & then compared.

→ As shared symmetric key is used → MAC
∴ can be also used for ① Authentication
② Non-Repudiation

→ Exchanging shared symmetric key on an insecure channel is an issue.
→ key size (bits) not fixed → decided by sender & Receiver

→ Inputs:
K : Bits of key
L : No. of Blocks
b : Bits in each Block
ipad : 86 bits
opad : 56 bits

inner padding Bits
outer padding
Step 1

→ No. of bits in key = No. of bits in Block
↳ Algo makes bits in key same as No. of Bits in Block, if already not equal.

① If K < b → Add padding bits (0) to left of key.

② If K > b → Trim No. of bits in K using Message Digest

Step 2 — Apply XOR b/w ipad bits and key K.

~~XOR (H1. K) ==~~ XOR (ipad K) = S1

(3) S1 + M ⟶ append original msg

(4) ↳ Apply message Digest, on o/p of step 3 & call it H1

(5) XOR opad bits and key K
XOR (opad K) = S2

(6) S2 + H1 ⟶ append H1 to S2

(7) Apply message Digest to generate H2

⇒ No. of times MD algo is applied: 2 (in Algo) + 1 (for trimming K)

→ Message is sent along with H2 to receiver (plain text). Receiver finds MAC from same Algo (same values of ipad, opad, key K) ⟶ & then compare. If MAC's are same → data is intact.

problems:

(1) sharing secret key

(2) multiple receivers ⟶ Key has to be shared with all

(3) Non-Repudiation in case of multiple receivers → How to identify msg came from sender & not other receivers ⟶ as they also have same key.