IDEA - (International Data Encryption Algorithm)

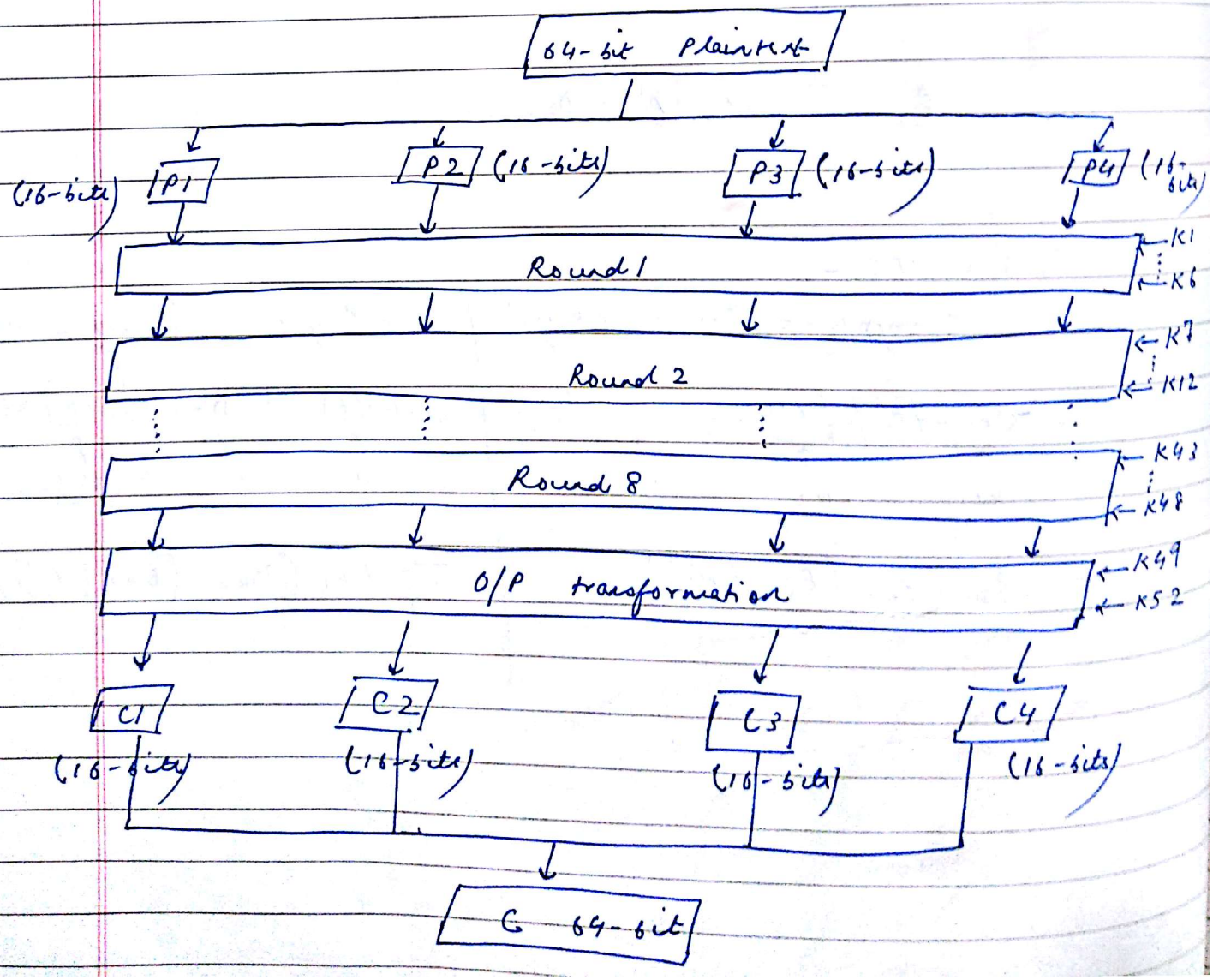
64-bit plain text
128-bit key for encrypting an original message.

→ we have 8 rounds here.

128 bit key ⇒ 6-bit subkeys (each round)
K1 - K6, K7 - K12, ... } 8-rounds

O/P transformation block ⇒ 4-bit subkey (K49 - K52)

Broad level steps in IDEA -



(K1 - K4 each comes 16-bit key)

Round 1

Step 1 :- P1 * K1 (multiply 2 mod)

Step 2 - P2 + K2

Step 3 - P3 + K3

Step 4 - P4 * K4

Step 5 - 1 ⊕ 3 (Step 1 ⊕ Step 3)

Step 6 - Step 2 ⊕ Step 4

Step 7 - Step 5 * K5

Step 8 - Step 6 + Step 7.

Step 9 - Step 8 * K6

Step 10 - 7 + 9

11 - 1 ⊕ 9

12 :- 3 ⊕ 9

13 - 2 ⊕ 10

14 - 4 ⊕ 10

16 bit -

P2 - 1111111111000000

K2 - + 1111111111000001

17 bit ⇒ we want % of 2^{16} ⇒ 16 bit

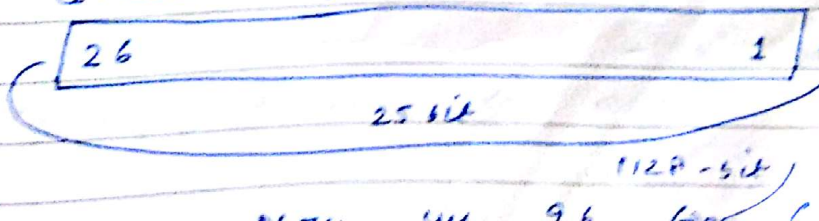
mod with { add ⇒ 2^{16} ; multiply ⇒ $2^{16}+1$

Key shift -

128 bit ⇒ K1 - K6 { (6 - subkey)

K1 - K6 (6 × 16) = 96 bit

For round 1, 128 - 96 = 32 bit (for next round) No. of circular shift 25 bit



(128-bit)

Now, we 96 from

o/p transformation -
4-bit subkey
(*, + & multiply *, add +)

Step 1: R1 * K1
Step 2: R2 + K2
Step 3: R3 + K3
Step 4: R4 * K4

Key strength ⇒ 2^{128} possible keys to break the method.

(After Round 8, the intermediate result is R1 to R4).

(10-12 rounds)

AES (Advanced Encryption Standard) -
Rijdael Algorithm -
PT ⇒ 128 bits to 256 bits
Key size ⇒ 128 bits, 192, 256

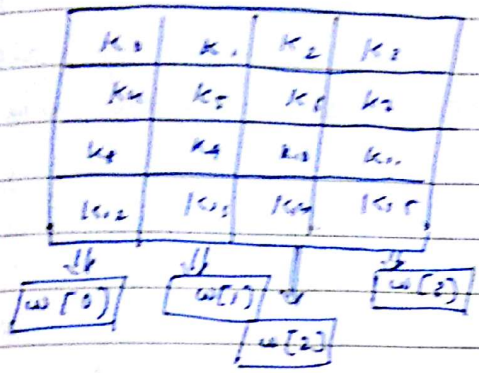
1) 128 bit PT combined with 128 bit key.
2) 128 bit PT combined with 256 bit key.

No. of Rounds → 10, 12, 14 (Depends on size of key)
10 (PT = 128 bit, keysize → 128 bit)
12 (PT = 128 bit, key → 192)
14 (PT = 128, key → 256)

① One-time initialization Process
a) Expand the 16-byte key to get actual key block.
b) Do one time initialization of the 16-byte plain text block. (called as state)
c) XOR the state with the key block.

Details (a) -
16 byte key ⇒ 1 word ⇒ 4 byte ⇒ 4 byte word.
↓
11 array (each array contain 4×4)
(11×4×4 = 176 byte)

There are 16 words in out of 44 array, 1 used for initialization & 16 for words.



Byte Partition Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12
Hex	00	01	02	03	04	05	06	07	08	09	0A	0B	0C

13	14	15
0D	0E	0F

Key Expansion step-1

w[0] = 00 01 02 03
w[1] = 04 05 06 07
w[2] = 08 09 0A 0B
w[3] = 0C 0D 0E 0F

If multiple of 4, w[4], w[8] the use a process

temp = w[i-1]
= w[4-1] = w[3]
= 0C 0D 0E 0F

Since i = 4, i mod 4 = 0.

temp = substitute (Rotate (temp)) XOR constant (i/4)

* Rotate (temp) -

• 0C 0D 0E 0F (Apply circular left shift on the contents of the word by one byte)

• ⇒ 0D 0E 0F 0C (After rotation).

* Substitute (Rotate (temp))

Substitute (0D 0E 0F 0C) (using AES S-box)

└76┘ → FE

x = 0, y = D

= D7.

x = 0, y = E

= AB

* Constant (i/4) - w[4] -

constant (4/4) = constant (1)

= 01 (from table).

* XOR →

D7 AB 76 FE

XOR

01 00 00 00

D6 AB 76 FE

New temp = D6 AB 76 FE

* Final process - w[4]

temp XOR (i - 4)

= temp XOR w(0)

D6 AB 76 FE

XOR

00 01 02 02

w[4] : 0 6 A A 7 4 F 0

w[i] = w[i-1] XOR w[i-4]
[i not a multiple of 4 & i >= 5].

w[5] = w[5-1] XOR w[5-4]
= w[4] XOR w[1]

0 6 A A 7 4 F 0
XOR 0 4 0 5 0 0 0 7

b)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13

B14	B15	B16

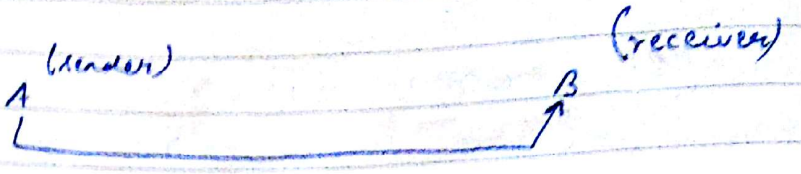
⇓

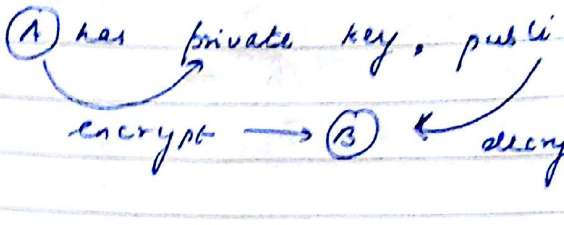
B1	B5	B9	B13
B2	B6	B10	B14
B3	B7	B11	B15
B4	B8	B12	B16

⇒ (state)

16-byte Plaintext box copied into 2-D 4x4 array, called state.

Digital signature -





No. message has no confidentiality, anyone can decrypt it.

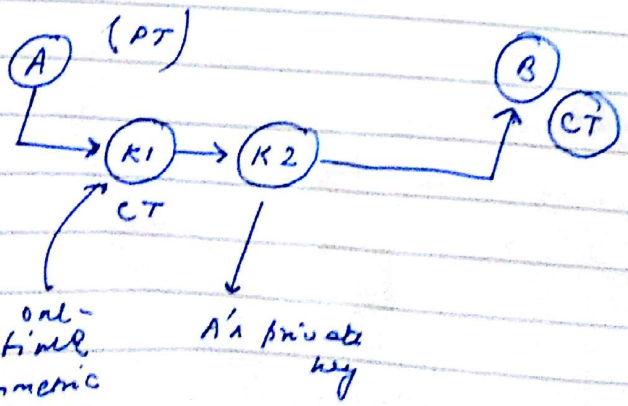
Now C encrypt using C's private key message ↙B not able decrypt using A's public key.

So some authentication must be there that A is sending & integrity also as for C's encryption. B's decryption will become non-readable.

Digital signature depends on the confidentiality or not?

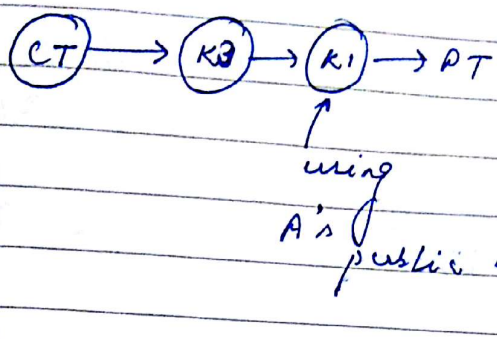
* Message Digest -

Digest or Hash -



⇒ Encryption process

Decryption - process.



This whole process ⇒ math function

It depends on

→ Authentication → Non-repudiation → Integrity

but not on conf.

LRC → is a longitudinal redundancy check

& CRC → cyclic Redundancy check

LRC →

original data

11100100 11011101 00111001 10101001

→ 11100100
→ 11011101
→ 00111001
→ 10101001

(XOR them) ← original data

00101001 → LRC.

Append LRC to original data.

At receiver side discard LRC. XOR rest of data & check with received LRC.

Digest on normal number.

original Num → 7391743

Operation	Result
7 × 3	21
Discard 1st digit from right	1
(Multiply with discarded digit) 1 × 9	9
9 × 1	9
9 × 7	63
Discard 1st digit	3
3 × 4	12
Discard	2
2 × 3	6 ⇒ Message digit.

Now 6 is sent to B

A → B.

If 2 MD are same ⇒ collision.

MD5 → (128 bit data)

Step 1: Padding - The total length of this should be 64 bit less than a multiple of 512.

If initial data bits 1000 ⇒ 1472 ← padded length.

1536 ⇒ (512 × 3) = 1536.

1536 − 64 = 1472.

i.e. (1000 + 472) → padding.

⇒ 1472 length of data after padding process.

Step 1: { Original message + padding (1 − 512 bits)

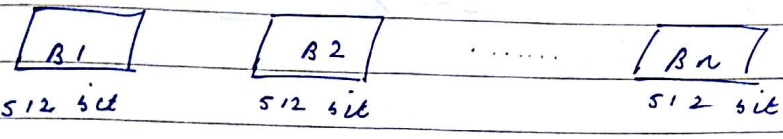
text, not 1000.

1 week

Step 2 - Add (length of the original message) with o/p of step 1.

step 1 + length of original message. + 1000

Step 3 : Divide the original message into 512 blocks.



Step 4 : Chaining process - A, B, C, D.

A Hex 01 23 45 67
B Hex ... ...
C Hex ... ...
D ... ...

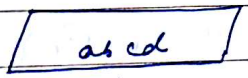
} Refer work

Last ← a = b + ((a + process P(b, c, d) + M[i] + T[k]) <<< s)

Step 5 -

Process block first →

5.1 a = A, b = B, c = C, d = D (copy values)



128-bit register (Abstract views)

intermediate result + final o/p.

Internal views.

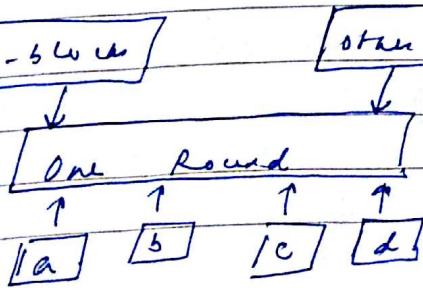
5.2 for all (B1 .... Bn)

512-bit block ⇒ 16 sub-blocks & each block contain 32-bit

5.3 It has 4 rounds -

M[0] .... M[15] ⇒ 16 sub-blocks

T ⇒ array 64 elements & each element contain 32 bits.



Contd...

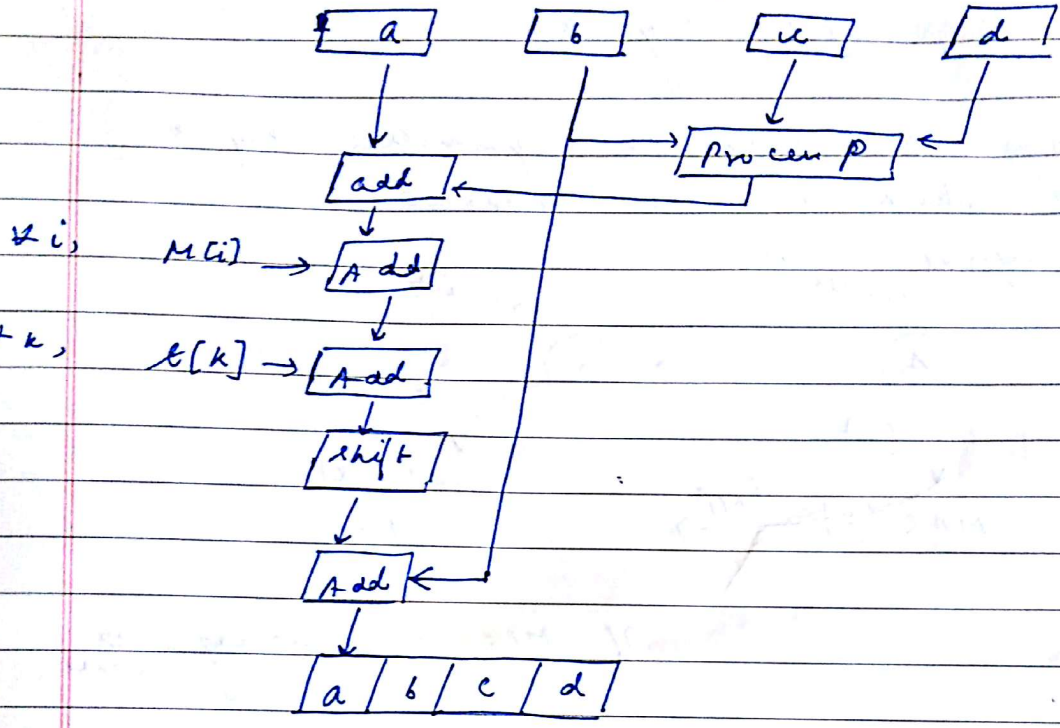
Cryptography & N/w security.

Cont... MD5 →

Process P -

Round 1

Process P
(b AND c) OR ((NOT b) AND (d))



calc for Round 2, 3, 4. (Refer Book).

CT-2

→ IDEA - (shift 1 imp.)

✓ → AES - (1 imp.) (no. of blocks, T & K no. of elements) ✓ P in 3rd round

→ MD5 → Algo & process P. (eg. Device P in 3rd round)

X → RC4

X → SHA → learn formula

→ MAC / HMAC

→ Digital signature - (LRC ₱ → 1 imp.) in Message Digest.

→ Hash function.

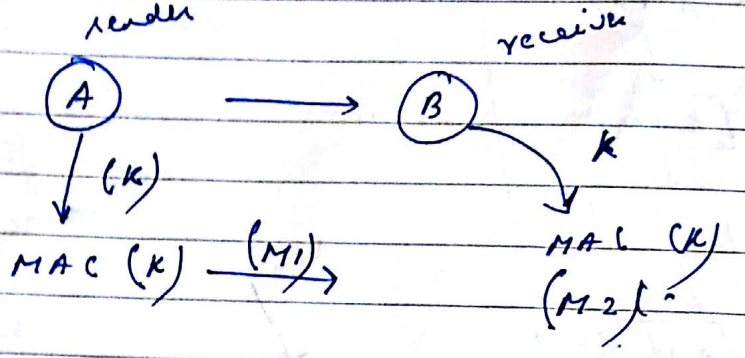
X → RSA, DSA of MD. (Difference b/w them using digital signature).

MAC - It depends on symmetric key cryptography.

(A) sender (B)

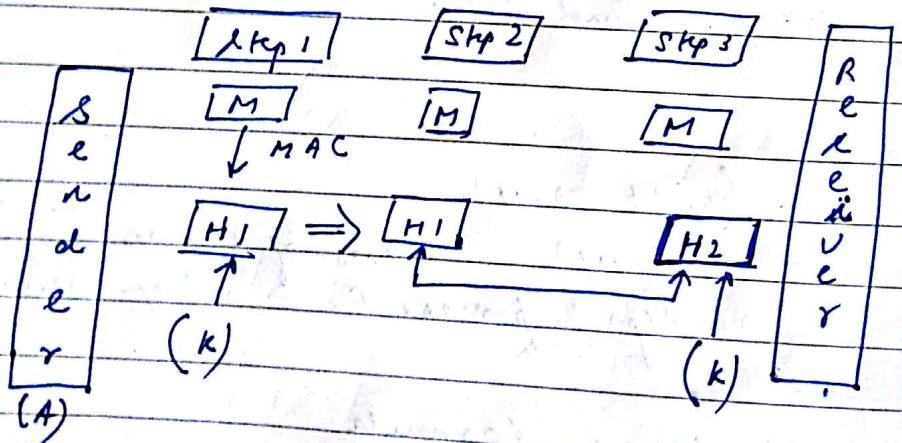
encrypts the message using a secret key K.

A & B share a symmetric key K, which is not known to anyone else.



If M1 = M2, message is same.

If M1 ≠ M2, B rejects this message.



MD5 is not an encryption / Decryption algo, so not a cryptographic algorithm.

MAC is a cryptographic algo which uses MD5 or SHA.

If we use HMAC (Hash based MAC) for encryption / decryption.

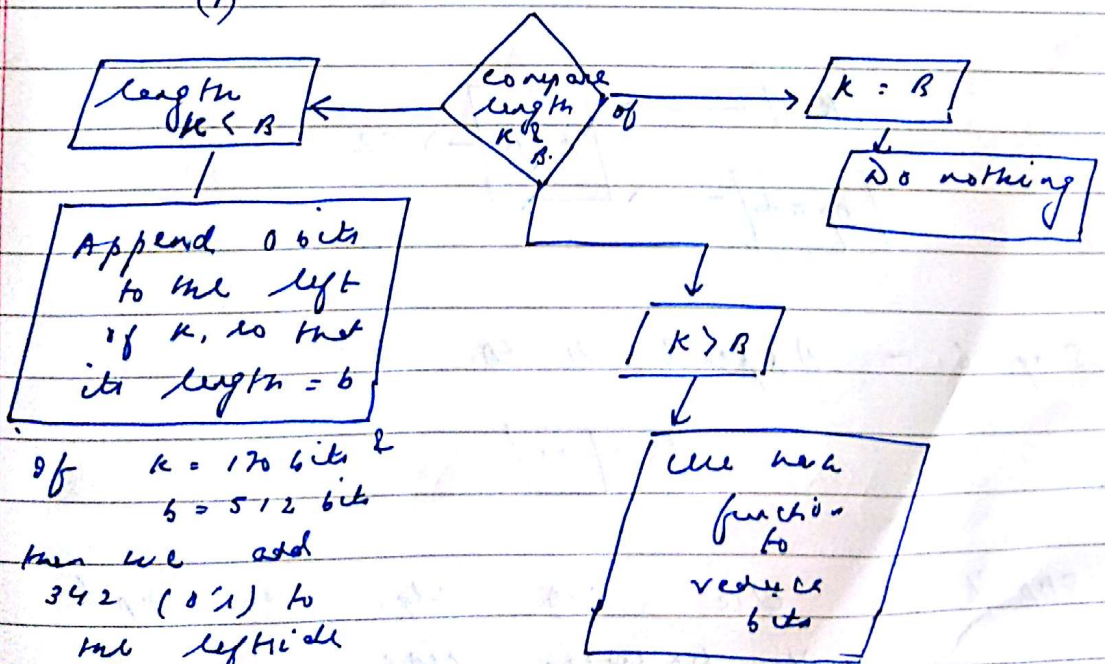
HMAC - The message digest of hash function used are MD5 or SHA-1.
M is the i/p message whose MAC is to be calculated. L is the no. of blocks in the message M. B is the no. of bits in each block. K is the shared symmetric key. ipad is a string 00110110 ..... repeated B/8 times.
opac → the string 01011010 ..... repeated B/8 times.

MD5, SHA-1
↓
original message → (MD)
↓
Encrypt → (MAC)
Final output
↑
(K)

HMAC concept -

Step 1 : Make the length of K = B.

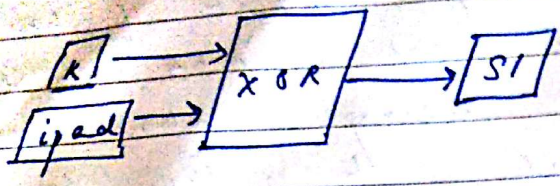
(1)



If K = 170 bits & b = 512 bits
then we add 342 (0's) to the left side of K.

∴ K = B.

Step 2 : XOR K with ipad to produce S1.



, S1 is a variable.

Step 3 : Append (M) to S1 (M → original message)

(S1) + [original message] (M)

↓

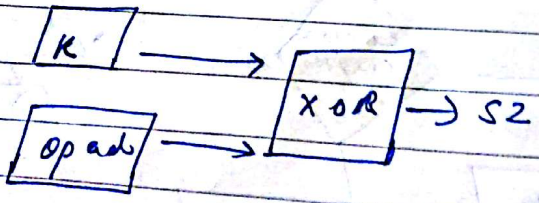
[S1 | (M)]

Step 4 : Use any MD algo on Step 3 (MD5, SHA 1)

Output = H.

~~HMAC Concept~~ -

Step 5 - XOR K with opad to produce S2



Step 6 - Append H to S2

[S2 | H]

Step 7 : Use MD algo on Step 6. It produces MAC.

Q. Difference b/w symmetric & asymmetric key cryptography.

Q. The process of the digital signature & hash function.

Q. LRC.

Q. Create a MD using word given by us/no. (Write complete process in steps)

Q. No. of bits used in each process. (MD5 = 128 bits) 2^{128} combinations to break the code.

Q. What is collision?

Q. Requirements of Message Digest.

Q. Padding process of MD5.