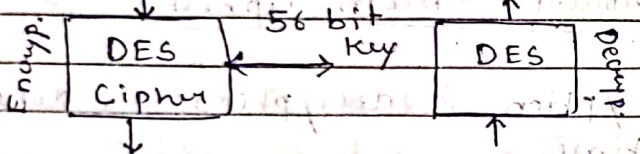
CH-4

DATA ENCRYPTION STANDARD (DES)

→ 64 bit plaintext → 64 bit P.T.



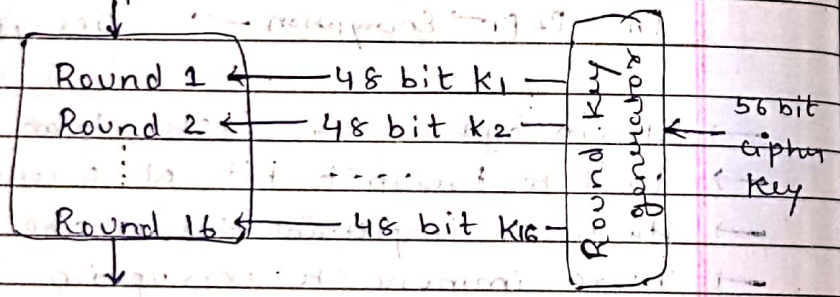
64 bit C.T. → 64 bit C.T.

→ 64 bit plaintext

↓

Initial permutation

↓



↓

Final permutation

↓

64 bit C.T.

- Initial and final permutation:

→ P-boxes

→ 64 bit input & 64-bit outputs

→ Permutations are keyless straight permutation.

→ Don't have cryptography significance in DES.

Eg: Find the output of this initial permutation box when the input is given in hexadecimal.

0x0002 0000 : 0000 0001

0002 = 0000 0000 0000 0010

0000 = 0000 0000 0000 0000

0000 = 0000 0000 0000 0000

0001 = 0000 0000 0000 0001

Output:

Two places will contain 1's (i.e. 63rd bit and 25th bit) everything else will be 0.

Eg: Prove that the initial and final permutations are the inverse of Each other by finding the outputs of the final permutation if the input is:

0x 0000 0080 0000 0002

0000 = 0000 0000 0000 0000

0080 = 0000 0000 1000 0000

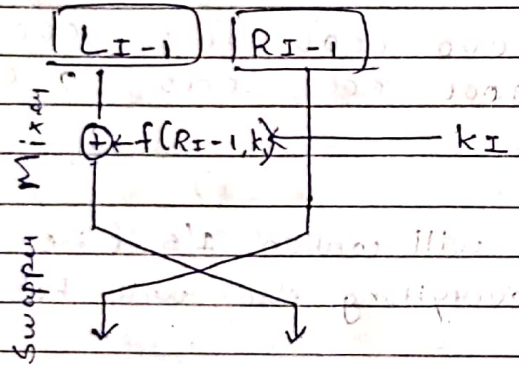
0000 =

0002 = 0000 0000 0000 0010

→ First 1's has occured on 25th bit and in final permutation table we have 64th bit as output.

→ Similarly, 2nd 1 will occured on 63rd bit and in final permutation table we have 15 bit as output.

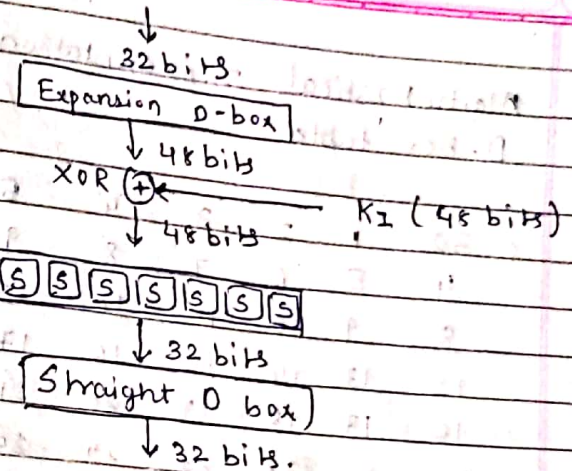
- **Rounds:**



→ Last round there will be no swap.

- **DES function:**

→ Applies 48 bit key to the rightmost 32 bits to produce 32-bit output.



- **Expansion D-box:**

→ R_{I-1} is divided into 8 4-bit sections.

→ Each 4 bit section then expanded to 6 bit.

→ I/p bits 1, 2, 3, 4 are copied to O/p bits 2, 3, 4, 5 resply.

→ O/p bit 1 comes from I/p bit 4 from previous section.

→ O/p bit 6 comes from I/p bit 1 of next section.

- Mathematical representation – Expansion D-box table.

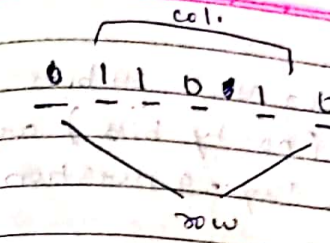
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- Whitener (XOR):

→ XOR operation on expanded right section & the round key.
→ Both are having length of 48 bits.
→ Round key is only used.

- S-boxes:

→ S-boxes do the real mixing (confusion)
→ DES uses 8 S-boxes, each with 6 bit input and 4 bit



row = 00 = 0
col. = 1101 = 13

→ 1001 (9) is the output of (011010) using S-box table.

→ 100011 (input)

row = 11 = 3
col. = 0001 = 1

~~00~~ (1111) (15) is the output of 100011 (input)

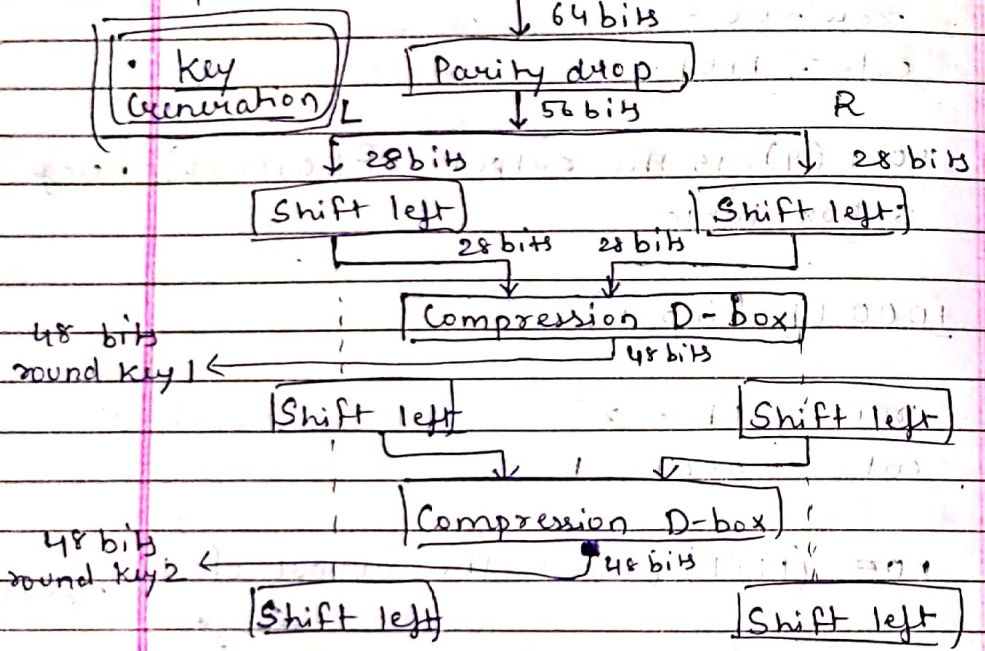
- Final permutation (Straight D box):

→ 32 bit i/p gives output of 32 bits.

- Key generation.

→ Round key generate 16 48 bit keys out of a 56 bit cipher key.

→ Normally key is given as a 64 bits in which 8 extra bits (parity bits) are dropped before actual key-generation process.



- **Step: 1 Parity drop:**

→ Processes before key expansion is compression transposition step that is known as parity drop.

- **Step: 2 : Shift left**

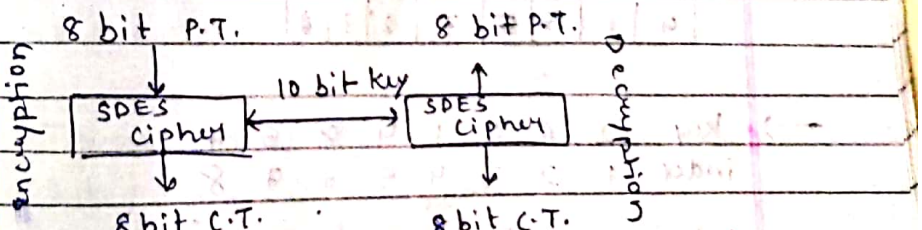
→ Key is divided into two 28 bits part
→ Each part is shifted left (Circular shift) one or two bits
→ Rounds - 1, 2, 9, and 16 : shift one bit
→ Other rounds : shift two bits
→ Combine both the parts to get 56 bit.

- **Step: 3 : Compression D-box :**

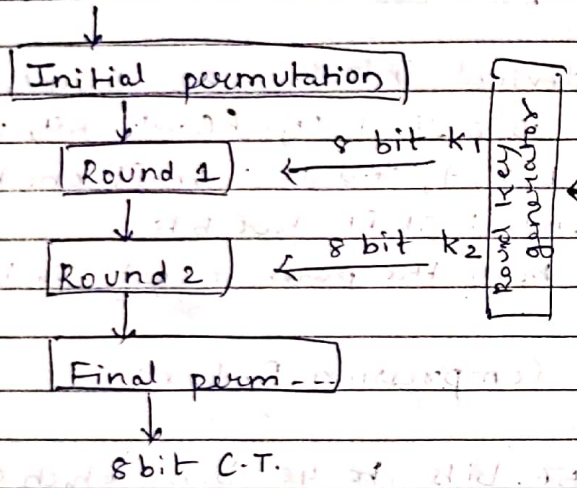
→ Changes 56 bits to 48 bits which are used as a key for a round.
→ Compression D-box table.

* **Simplified DES (S-DES):**

→ Smaller version of DES.
→ It is a block cipher.



- 8 bit P.T.



- Initial permutation. (IP)

2	6	3	1	4	8	5	7

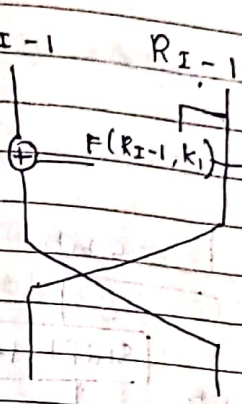
- Final permutation: (IP^{-1})

4	1	3	5	7	2	8	6

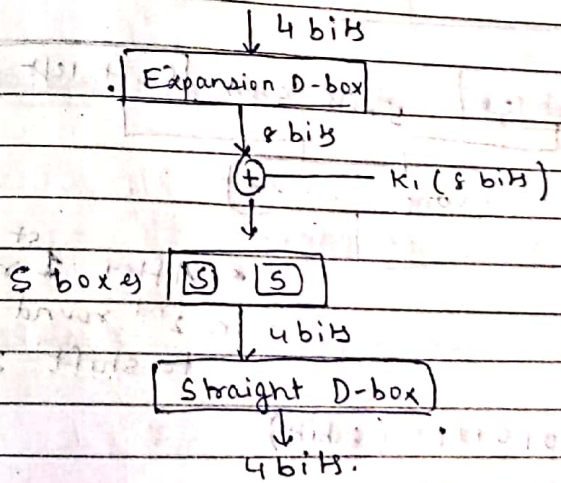
→ key 2 6 3 1 4 8 5 7
index 1 2 3 4 5 6 7 8

index 1 2 3 4 5 6 7 8
key 2 6 3 1 4 8 5 7

⇒ 4 1 3 5 7 2 8 6



- S-DES function:

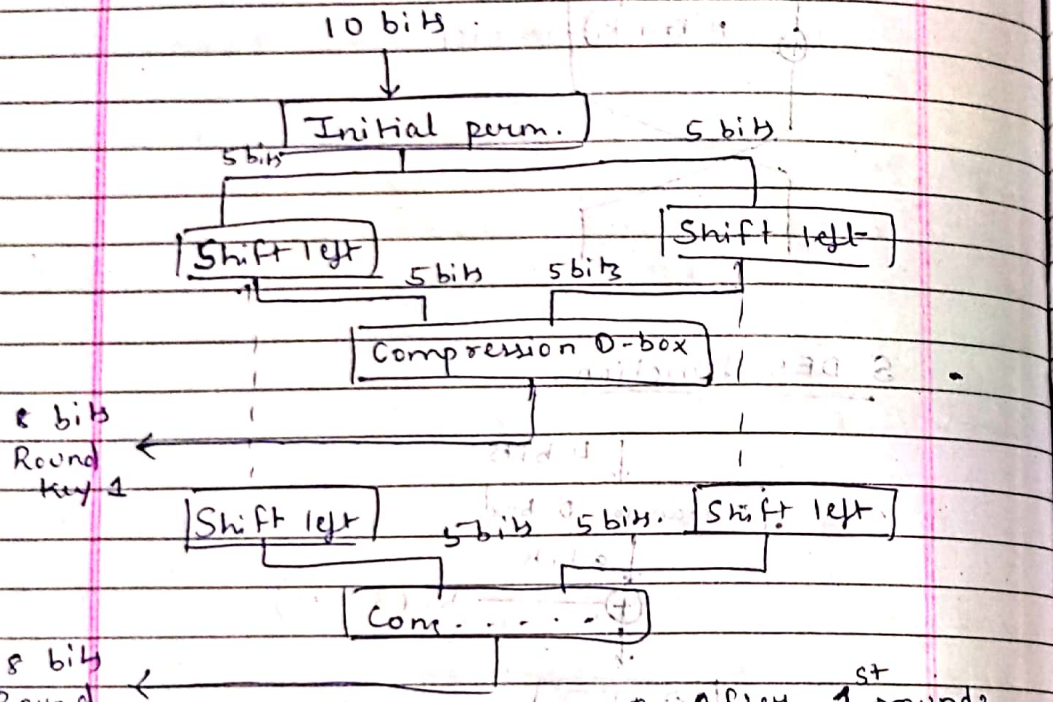


- Expansion D-box.
- Whitner (XOR)
- S-boxes (1st and 4th bit → row, 2nd & 3rd → col.)

→ Divide the 8 bits into 2 halves (4 bit Left and 4 bit Right)

∴ → Cell is o/p.

- Key generation:



* After 1st rounds, in 2nd round we have to shift 2 bits.

Q. P.T. = 10010111 (8 bit)

Key = 1010000010

C.T. = ?

Algo. = S-DES.

P-10 box : Initial permutation

3	5	2	7	4	10	1	9	8	6

Step: 1 → Key generation,

Step: 1 Initial permutation.

I/P = 1010000010

O/P = 1 0 0 0 0 0 1 1 0 0

Step: 2 Divide in two → Left and Right

L = 10000

R = 01100

Apply circular shift left (1 bit)

∴ L = 00001

R = 11000

Step: 3 Compression D-box

I/P = 0000111000

O/P = 1 0 1 0 0 1 0 0

Compression D-box.

6	3	7	4	8	5	10	9

Now, 2nd round key generation:

~~I/P = 10100100~~

~~Now, L = 10000~~
~~R = 01000~~
~~L = 10000 L = 00001~~
~~R = 01100 R = 11000~~

Shift 2 bit,

∴ ~~L = 00010~~ L = 00100
~~R = 01000~~ R = 00011

Now, Compression D box

I/P = ~~00010100011~~ I/P = 00100000111
O/P = ~~10010010~~ O/P = 01000011

Now, K_1 = 10100100
K_2 = 01000011
P.T. = 10010111

Step: 2

→ Encryption process:

Step: 1 I/P = 10010111
O/P = 01110101

Step: 2 Apply S-DES function on Right.

R = 1101 = I/P

→ Apply, expansion D-box.

∴ O/P = 11101011

Expansion D box

4	1	2	3	2	3	4	1

→ Now, Apply XOR operation.

~~I/P~~ 11101011
⊕ 10100100
01011111

Step: 3 Apply S-boxes,

Divide I/P in 2 parts.

L = 0101 R = 1111
row = 00 00 row = 11
col = 10 col = 11

Thus I/P is 0100 Thus I/P is 1111
O/P is 11 = O/P is 11
from S-box. from S box.

S_0

1	0	3	2
3	2	1	0
0	2	1	3
3	1	3	2

S_1

0	1	2	3
2	0	1	3
3	0	1	0
2	1	0	3

Final o/p is 1111.

Step: 4 Apply straight D-box.

I/P = 1111

O/P = 1111

from 4-bit permutation (table will be given)

Step: 3 Now,

→ XOR b/w Function & L side.

```
  0101
⊕ 1111
------
  1010
```

Step: 4 Swap.

I/P

L = 1010, R = 1101

O/P

L = 1101, R = 1010

∴ Round 1 C.T. = 11011010

This will become I/p to second round.

Step: 5 Again apply SDES function on right.

R = 1010 = I/P

Apply expansion D-box,

∴ O/P = 01010101

Apply, XOR operation,

```
  01010101
⊕ 01000011
----------
  00010110
```

Now, apply S-box,

L = 0001 R = 0110

Row = 01 Row = 00 } 11

Col. = 00 Col. = 11 }

Thus, I/P = 00010110

O/P = 1111

Now, apply straight D-box,

I/P = 1111

O/P = 1111

from 4-bit permutation (table will be given)

Step: 6 XOR b/w Function & L side

```
  1101
⊕ 1111   ⇒ 0010
```

Step:7 Combine L and R;
00101010

Step:8 Apply final permutation;
I/P = 00101010
O/P = 00111000 (Final C.T.)

Q. P.T. = 123456ABCD132536
Key = AABB09182736CCDD
C.T. After round 1 → ?
Algo. = DES

Step:1 Key generation;
Step:1 Parity drop.
Key: Remove parity bits.

1 0 1 0 | 1 0 1 0
1 0 1 1 | 1 0 1 1
0 0 0 0 | 1 0 0 1
0 0 0 1 | 1 0 0 0
0 0 1 0 | 0 1 1 1
0 0 1 1 | 0 1 1 0
1 1 0 0 | 1 1 0 0
1 1 0 1 | 1 1 0 1

Step:2

1 1 0 0 0 0 1 1
1 1 0 0 0 0 0 0
0 0 1 1 0 0 1 1
1 0 1 0 0 0 0 1
0 0 1 1 1 1 1 1
0 0 0 0 1 1 0 1
1 1 1 1 1 0 1 0

28 bits L
28 bits R

(Table will be given)

Step:3 Apply circular shift left: on both side.

1 0 0 0 | 0 1 1 1
1 0 0 0 | 0 0 0 0
0 1 1 0 | 0 1 1 1
0 1 0 1 | 0 1 1 0
0 1 1 1 | 1 1 1 0
0 0 0 1 | 0 1 1 1
1 1 1 1 | 0 1 0 0

Step:4 Apply compression D-box:

0 0 0 1 1 0 0 1 1
0 1 0 0 1 1 0 0 1
1 1 0 1 0 0 0 0 0
0 1 1 1 0 0 1 0 1
1 1 0 1 1 1 1 0 1
1 0 0 0 1 1 0 0 0

∴ Key = 0001100101001100110100000111
00101101111010001100

Step-2 Encryption process.

Step 1

I/P = 0001 0010 0011 0100 0101 0110 0111 1000

L = 0 0010 100 / 1 0100 111 / 1 1010 110 / 0 1111 001

R = 1 2 3 4 5 6 7 8 : 0 0 0 1 1 0 0 0
9 10 11 12 13 14 15 16 : 1 1 0 0 1 0 1 0
17 18 19 20 21 22 23 24 : 0 0 0 1 1 0 0 0
25 26 27 28 29 30 31 32 : 1 0 1 0 1 1 0 1

Step 2

Apply DES function.

→ Apply expansion D-box on R.

R = I/P

~~1 0 0 0 1 1~~ O/P = 1 0 0 0 1 1 1 1
1 1 0 0 0 1 0 0 0 1 0 1 1 0
0 1 0 1 0 1 0 0
0 0 0 0 1 1 1 1
0 0 0 1 0 1 0 1
0 1 0 1 1 0 1 0

Apply XOR with Key 1.

1 0 0 1 0 1 1 0
0 1 0 1 1 0 1 0
1 0 0 0 0 1 0 0
0 1 1 1 1 1 0 1
1 1 0 0 1 0 1 1
1 1 0 1 0 1 1 0

Apply S-boxes.

	I/P	O/P
(i)	1 0 0 1 0 1 ~~1 0~~	
(ii)	~~0 1 0 1 1 0~~ 1 0 0 1	
(iii)	1 0 1 0 1 0	
(iv)	0 0 0 1 0 0	
(v)	0 1 1 1 1 1	
(vi)	0 1 1 1 0 0	
(vii)	1 0 1 1 1 1	
(viii)	0 1 0 1 1 0	

Apply S-boxes:

	SBox	Row	Col	O/P
(i)	S_0	$11 \Rightarrow 3$	2	8
(ii)	S_1	$11 \Rightarrow 3$	2	10
(iii)	S_2	$10 \Rightarrow 2$	5	15
(iv)		$00 \Rightarrow 0$	2	14
(v)		$01 \Rightarrow 1$	15	6
(vi)		$00 \Rightarrow 0$	14	5
(vii)		$11 \Rightarrow 3$	7	7
(viii)		$00 \Rightarrow 0$	11	14

→ Applying final permutation:

(From Straight D box table which will be given)

0 1 0 0 1 1 1 0
1 1 0 1 1 1 1 1
0 0 1 1 0 1 0 1
1 1 1 0 1 1 0 0

→ Now, XOR will intial left permutation which gives,

0 1 0 1 1 0 1 0
0 1 1 1 1 0 0 0
1 1 1 0 0 0 1 1
1 0 0 1 0 1 0 0

Now, final output..

0 1 0 1 1 0 1 0
0 1 1 1 1 0 0 0
1 1 1 0 0 0 1 1
1 0 0 1 0 1 0 0
0 0 0 1 1 0 0 0
1 1 0 0 1 0 1 0
0 0 0 1 1 0 0 0
1 0 1 0 1 1 0 1

∴ C.T. after Round 1 = 5A78D

Page
Date

Q The plain text which is given to us is 10010110 and the key which is given 1010000010

→ We have already done this,

Step: 1

$K_1 = 10100100$

$K_2 = 01000011$

→ Encryption process:

Step: 2

Step: 1 I/P = 10010110

O/P = 01011001

Step: 2 Apply SDES function on Right.

R = 1001

→ Apply expansion D-box,

∴ O/P = 11000011

→ Now, apply XOR operation.

$$\begin{array}{r} 11000011 \\ \oplus\ 10100100 \\ \hline 01100111 \end{array}$$

Step: 3 Apply S-boxes,

Divide I/P in 2 parts

L = 0110 , R = 0111

row = 00 row = 01

col. = 11 col. = 11

o/p = 10 from s-box o/p = 11 from s-box

Final o/p = 1011

Step: 4 Apply straight D-box,

o/p = 0111

Step: 3 Now,

XOR b/w Function & L side

$$\begin{array}{r} 0110 \\ \oplus\ 0101 \\ \hline 0011 \end{array} \qquad \begin{array}{r} 0101 \\ \oplus\ 0111 \\ \hline 0010 \end{array}$$

Step: 4 Swapping,

I/p

L = 0010 , R = 1001

o/p

L = 1001 , R = 0010

Round 1 C.T. = 10010010 This will become I/p to second round

Step: 5 Again apply SDES function on right

R = 0010

Apply expansion D-box,

∴ o/p = 00010100

Apply XOR operation,

$$\begin{array}{r} 00010100 \\ \oplus\ 01000011 \\ \hline 01010111 \end{array}$$

Now, apply S-box,

L = 0101 R = 0111

Row = 01 } 01 Row = 01 } 11

Col. = 10 Col. = 11

I/p = 01010111

o/p = 0111

Apply Straight D-box,

I/p = 0111

o/p = 1110

Step: 6 XOR b/w Function & L side

$$\begin{array}{r} 1001 \\ \oplus\ 1110 \\ \hline 0111 \end{array}$$

Step:7 Combine L & R.

01110010

Step:8 Final permutation,

I/P = 01110010

O/P = 10101100 (Final C.T.)

- DES Analysis:

→ Properties:

→ Avalanche effect - small change in PT or key should create significant change in CT.

→ Completeness effect - each bit of CT needs to depend on many bits on PT.

- DES weakness:

S-box:

→ In S-box 4, last 3 bits can be derived in same way as first output bit by complementing some of the input bits.

→ 2 specifically chosen inputs to S-box can create same output.

CH-5
Advanced Encryption Standard (AES)

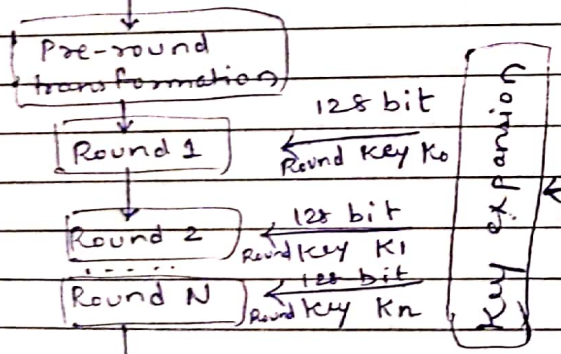
- Criteria:

→ Security:
  - 128 bit key

→ Cost:
  - computational efficiency and storage requirement for diff. implementation such as hardware, software or smart cards.

→ Implementation:
  - Flexibility & simplicity.

→ 128 bit P.T.



Round	key
10	128
12	192
14	256

- Data units:

→ Bit:
  → Binary digit with value 0 & 1
  → Use lowercase letter b refer to a bit.

→ Byte:
  → Group of 8 bits that can be treated as single entity.
  → Use lowercase bold letter b

→ Word:
  → Group of 32 bits that can be treated as single entity.
  → Use lowercase letter w.

→ Block:
  → Group of 128 bits.

→ State:
  → AES uses several rounds in which each round is made of several stages.
  → At beginning and end of the cipher, AES uses the term data block; before and after each stage, data block is referred to as a state.
  → Use uppercase letter S. to refer to a state.
  bold

Q. Given text block is : "AES USES A MATRIX" Represent it in the form of state.

→ State is made up of 128 bits i.e. 16 bytes we have only 14 bytes therefore we have to add bogus characters.

AESUSESAMATRIXZZ

Decimal →

00 04 18 20 18 04 18 00 12 00 19 17 08 23 25 25

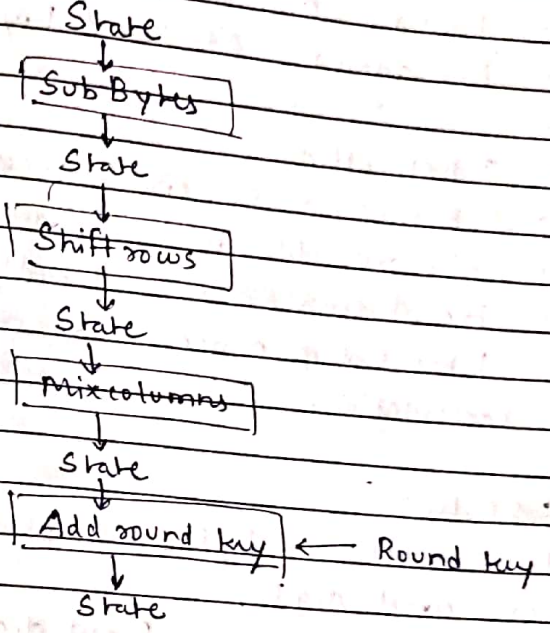
Hexa decimal →

00 04 12 14 12 04 12 00 0C 00 13 11 08 17 19 19

Now, Arrange all the values in 4×4 matrix.

00	12	0C	08
04	04	00	17
12	12	13	19
14	00	11	19

- Structure of each round:



Note: One Add round key is applied before first round.

Note: 3rd transformation (Mix columns) is missing in last round.

- Transformation:

→ To provide security, AES uses four types of transformation.

(1) Substitution : SubBytes.

(2). Permutation: Shift Rows
(3) Mixing : Mix Columns
(4) Key adding : Add Round Key.

(1) Substitution:
→ Substitution is done for each byte.
→ Only one table is used for substitution.
→ Transformation is defined by either table lookup process or mathematical calculation.

SubBytes:
- To substitute a byte, interpret byte as two hexadecimal digits.
- Left digit - row , Right digit - col.

Eg: Apply subByte transformations on below state.

0th row and 0th col. ←	00	12	0C	08	SubByte ⟹
	04	04	00	23	
	12	12	13	19	
1st row and 4th col. ←	14	00	11	13	4×4

63	C9	FE	30
F2	F2	63	26
C9	C9	7D	D4
FA	63	82	7D

4×4

(2) Permutation:
→ Permutes bytes
→ Order of bits in byte is not changed.

ShiftRows:
→ Shift left (circular)
→ No. of byte shifting depends on row no.
→ Row 0 - no shifting, row 1 - shift 1 byte, and so on.

Eg: On the previous example apply shift Rows:

63	C9	FE	30	(no shifted)
F2	63	26	F2	(1 byte shifted)
7D	D4	C9	C9	(2 byte shifted)
7D	FA	63	82	4×4 (3 byte shifted)

(3) Mixing:
→ Changes the content of each byte by taking 4 bytes at a time and combining them to recreate 4 new bytes.

→ matrix multiplication - square matrix X column matrix.

Mix Columns:
→ Operates at column level.
→ Multiplication of constant square matrix and state column.

$$\begin{bmatrix}\text{const.}\\ \text{matrix.}\end{bmatrix}_{4\times4} \times \begin{bmatrix}1^{st}\\ \text{column}\end{bmatrix}_{4\times1} = \begin{bmatrix}\ \\ \ \end{bmatrix}_{4\times1}$$

Repeating this 4 times, we will get all columns and hence will get 4x4 mixcolumn matrix.

Eg: Suppose const. matrix,

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & 63 & 26 & F2 \\ 7D & D4 & C9 & C9 \\ 7D & FA & 63 & 82 \end{bmatrix}$$

$$= \begin{bmatrix} (02\cdot63)\oplus(03\cdot F2)\oplus(01\cdot7D)\oplus(01\cdot7D) \\ (01\cdot63)\oplus(02\cdot F2)\oplus(03\cdot7D)\oplus(01\cdot7D) \\ (01\cdot63)\oplus(01\cdot F2)\oplus(02\cdot7D)\oplus(03\cdot7D) \\ (03\cdot63)\oplus(01\cdot F2)\oplus(01\cdot7D)\oplus(02\cdot7D) \end{bmatrix} \quad \text{(1)}$$

→ Lets see some calculations,

1 x 63 = 01100011
1 x F2 = 11110010
1 x 7D = 01111101
1 x C9 = 11001001
1 x D4 = 11010100
1 x FA = 11111010
1 x FE = 11111110
1 x 26 = 00100110
1 x 30 = 00110000
1 x 82 = 10000010

Note: In $gf2^8$
00000001 = 1
00000010 = x
00000011 = x+1
$x^8 = x^4 + x^3 + x + 1$

∴ $2 \times 63 = x(x^6 + x^5 + x + 1) = x^7 + x^6 + x^2 + x$
~~$3 \times F2 = (x+1)(x^7 + x^6 + x^5 + x^4 + x)$~~
$= 11000110$

$2 \times F2 = x(x^7 + x^6 + x^5 + x^4 + x)$
$= x^8 + x^7 + x^6 + x^5 + x^2$
$= x^4 + x^3 + x + 1 + x^7 + x^6 + x^5 + x^2$
$= 11111111$

$2 \times 7D = x(x^6 + x^5 + x^4 + x^3 + x^2 + 1)$
$= x^7 + x^6 + x^5 + x^4 + x^3 + x$
$= 11111010$

$2 \times C9 = x(x^7 + x^6 + x^3 + 1)$

$= x^8 + x^7 + x^4 + x$

$= x^4 + x^3 + x + 1 + x^7 + x^4 + x$

$= 10000\,1001$

$2 \times D4 = x(x^7 + x^6 + x^4 + x^2)$

$= x^8 + x^7 + x^5 + x^3$

$= x^4 + x^3 + x + 1 + x^7 + x^5 + x^3$

$= 10110011$

$2 \times FA = x(x^7 + x^6 + x^5 + x^4 + x^3 + x)$

$= x^8 + x^7 + x^6 + x^5 + x^4 + x^2$

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

$= 11101111$

$2 \times FE = x(x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x)$

$= x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2$

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

$= 11100111$

$2 \times 26 = x(x^5 + x^2 + x)$

$= x^6 + x^3 + x^2$

$= 01001100$

$2 \times 30 = x(x^5 + x^4)$

$= x^6 + x^5$

$= 01100000$

$2 \times 82 = x(x^7 + x)$

$= x^8 + x^2$

$= x^4 + x^3 + x + 1 + x^2$

$= 00011111$

$3 \times 63 = (x+1)(x^6 + x^5 + x + 1)$

$= x^7 + x^6 + x^2 + x + x^6 + x^5 + x + 1$

$= 10100101$

$3 \times F2 = (x+1)(x^7 + x^6 + x^5 + x^4 + x)$

$= x^8 + x^7 + x^6 + x^5 + x^2 + x^7 + x^6 + x^5 + x^4 + x$

$= x^4 + x^3 + x + 1 + x^2 + x^4 + x$

$= 00001101$

$3 \times 7D = (x+1)(x^6 + x^5 + x^4 + x^3 + x^2 + 1)$

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

$= 10000111$

$3 \times C9 = (x+1)(x^7 + x^6 + x^3 + 1)$

$= x^8 + x^7 + x^4 + x + x^7 + x^6 + x^3 + 1$

$= x^4 + x^3 + x + 1 + x^7 + x^4 + x + x^7 + x^6 + x^3 + 1$

$= 01000000$

Similarly,

$3 \times D4 =$

$3 \times FA =$

$2 \times FE =$

$2 \times 26 =$

$2 \times 30 =$

$2 \times 82 =$

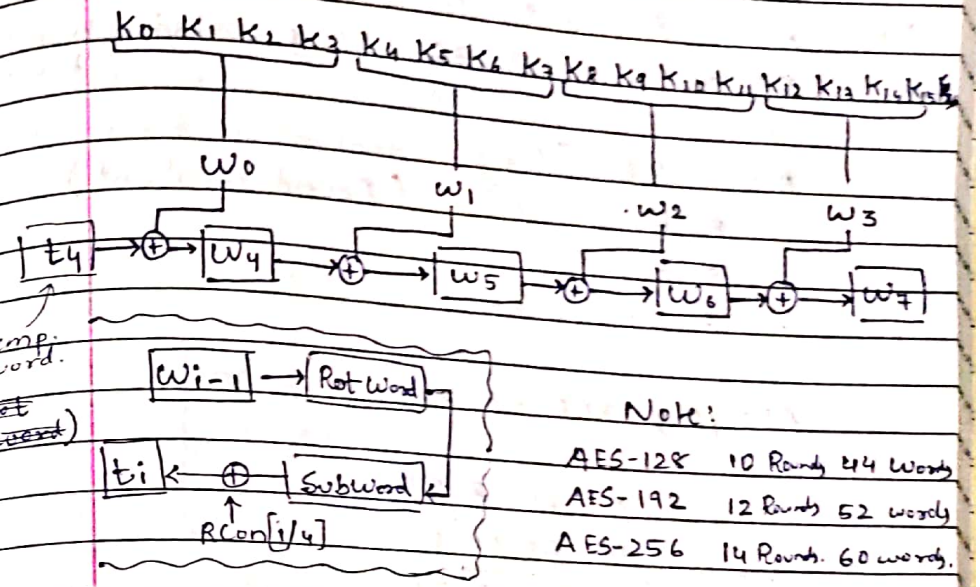
From ①,

CB

```
  1 1 0 0 0 1 1 0
⊕ 0 0 0 0 1 1 0 1
-----------------
  1 1 0 0 1 0 1 1
⊕ 0 1 1 1 1 1 0 1
-----------------
  1 0 1 1 0 1 1 0
⊕ 0 1 1 1 1 1 0 1
-----------------
  1 1 0 0 1 0 1 1
```

= CB

- Key expansion:



Note:

AES-128 10 Rounds 44 Words
AES-192 12 Rounds 52 words
AES-256 14 Rounds. 60 words.

∴ [P.T.] (4×4) ⊕ $\begin{bmatrix} K_0 & K_4 & K_8 & K_{12} \\ K_1 & K_5 & K_9 & K_{13} \\ K_2 & K_6 & K_{10} & K_{14} \\ K_3 & K_7 & K_{11} & K_{15} \end{bmatrix}$ 4×4.

Round Key 0

Eg: Cipher key : 24 75 A2 B3 34 75 56 88 8B
31 E2 12 00 13 AA 54 87

Here, i = 4

So, Wi-1 = W3

We take W3 as i/p

W_3 = 13 AA 54 87

Rotword ← R.W. = AA 54 87 13 (Left shift kind of)

S.B. = AC 20 17 70 (Using subbytes transformation table)

Now, RCon[i/4]

= RCon[4/4] (Round constant)

= RCon[1]

= 01000000

Applying XOR operation,

S.B. = AC 20 17 7D

$\oplus$ 01 00 00 00

t_4 = AD 20 17 7D

→ AES has no weak key ~~and~~ as compared to DES which has 4 weak key.

→ Here, if in cipher key all are zeroes in C.T. then also we will get some text, hence brute force is not possible.

& ~~Round constants~~

0 - 01000000	5 - 20000000
1 - 02000000	6 - 40000000
2 - 04000000	7 - 80000000
3 - 08000000	8 - 1B000000
4 - 10000000	9 - 36000000

Q → P.T. which is given to us is AES USES A MATRIX.

→ Cipher key given to us, is:

24 75 A2 B3 34 75 56 88 31 E2 12

→ ~~Need to calculate~~ 00 13 AA 54 87

Need to calculate cipher text after round 1.

Step:1 Convert P.T. to state.

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 17 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}_{4\times4}$$

Step:2 Apply pre-round transformation.

→ Here the round 0 key is:

$$\begin{bmatrix} 24 & 34 & 31 & 13 \\ 75 & 75 & E2 & AA \\ A2 & 56 & 12 & 54 \\ B3 & 88 & 00 & 87 \end{bmatrix}$$

→ Apply ~~two~~ matrix addition with I/P state.

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 17 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \oplus \begin{bmatrix} 24 & 34 & 31 & 13 \\ 75 & 75 & E2 & AA \\ A2 & 56 & 12 & 54 \\ B3 & 88 & 00 & C7 \end{bmatrix}$$

$$= \begin{bmatrix} 24 & 26 & 3D & 1B \\ 71 & 71 & E2 & BD \\ B0 & 44 & 01 & 4D \\ A7 & 88 & 11 & 9E \end{bmatrix}$$

Step:3 Apply Round transformation / operation.

Step:1 Apply subBytes. (using subBytes transformation table.)

$$= \begin{bmatrix} 36 & F7 & 27 & AF \\ A3 & A3 & 98 & 7A \\ E7 & 1B & 7C & E3 \\ 5C & C4 & 82 & 0B \end{bmatrix}$$

Step:2 Shift Rows.

$$\leftarrow \begin{bmatrix} 36 & F7 & 27 & AF \\ A3 & 98 & 7A & A3 \\ 7C & E3 & E7 & 1B \\ 0B & 5C & C4 & 82 \end{bmatrix}$$

Step:3 Mix Columns:

→ Constant matrix:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} 36 & F7 & 27 & AF \\ A3 & 98 & 7A & A3 \\ 7C & E3 & E7 & 1B \\ 0B & 57 & C4 & 82 \end{bmatrix}$$

$$= \begin{bmatrix} (02 \cdot 36) \oplus (03 \cdot A3) \oplus (01 \cdot 7C) \oplus (01 \cdot 0B) \\ (01 \cdot 36) \oplus (02 \cdot A3) \oplus (03 \cdot 7C) \oplus (01 \cdot 0B) \\ (01 \cdot 36) \oplus (01 \cdot A3) \oplus (02 \cdot 7C) \oplus (03 \cdot 0B) \\ (03 \cdot 36) \oplus (01 \cdot A3) \oplus (01 \cdot 7C) \oplus (02 \cdot 0B) \end{bmatrix}$$

→ 1 x 36 = 00110110
1 x A3 = 10100011
1 x 7C = 01111100
1 x 0B = 00001011
1 x F7 = 11110111
1 x 98 = 10011000
1 x E3 = 11100011
1 x 57 = 01010111
1 x 27 = 00100111
1 x 7A = 01111010
1 x E7 = 11100111
1 x C4 = 11000100

1 x AF = 10101111
1 x 1B = 00011011
1 x 82 = 10000010

→ $2 \times 36 = x(x^5 + x^4 + x^2 + x)$

$= x^6 + x^5 + x^3 + x^2$

$= 01101100$

→ $2 \times A3 = x(x^7 + x^5 + x + 1)$

$= x^8 + x^6 + x^2 + x$

$= x^4 + x^3 + x + 1 + x^6 + x^2 + x$

$= 01011101$

→ $2 \times 7C = x(x^6 + x^5 + x^4 + x^3 + x^2)$

$= x^7 + x^6 + x^5 + x^4 + x^3$

$= 11111000$

→ $2 \times 0B = x(x^3 + x + 1)$

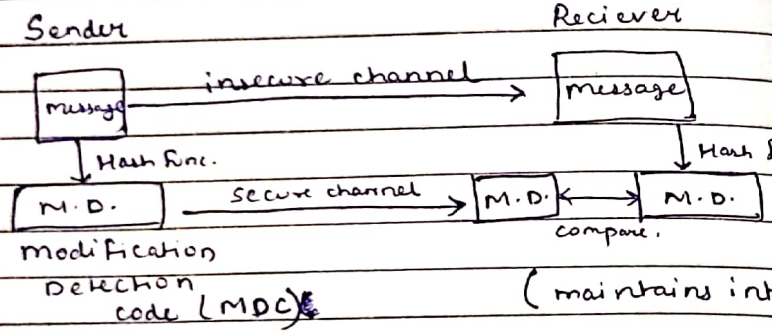
$= x^4 + x^2 + x$

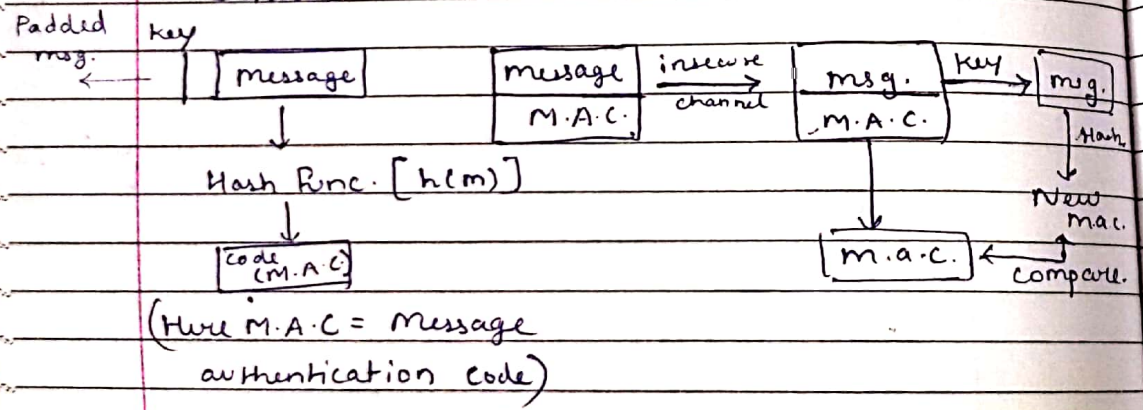
$= 00010110$

UNIT

Cryptographic hash Functions

- **Hash function** (Message Digest)
  - ↳ A code having fixed size. (compression)
  - integrity
  - Authentication





→ h(K|m)

h = hash, K = key, | = append, m = message.

- **Criteria for hash function:**

→ There are 3 properties:

(1) Pre-image resistance

$y = h(x)$

↳ message digest

→ we have to take x such that if any attacker takes another x (say x') then he/she should not get same message digest i.e. $h(x') \neq y$, is known as pre-image resistance.

(2) Second pre-image resistance

$h(x) = h(x')$

→ We have to take x such that if any attacker takes another x, it will take almost infinite time to find another x' which will give same hash function.

$x \neq x'$

(3) Collision resistance

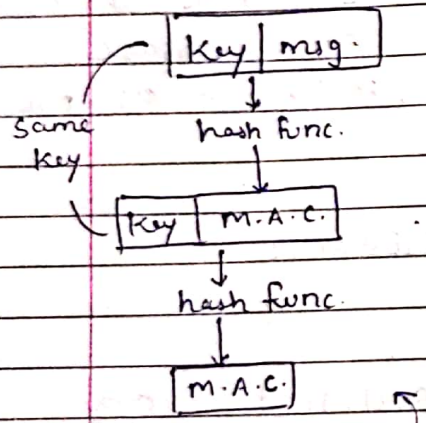
$h(x) = h(x')$, $x \neq x'$. Same as second pre-image resistance but the difficulty level increases in collision resistance.

- **Security of M.A.C:**

→ key revealed.
→ pre image attacks
→ ~~same~~ for some pair same M.A.C. is ~~even~~ possible

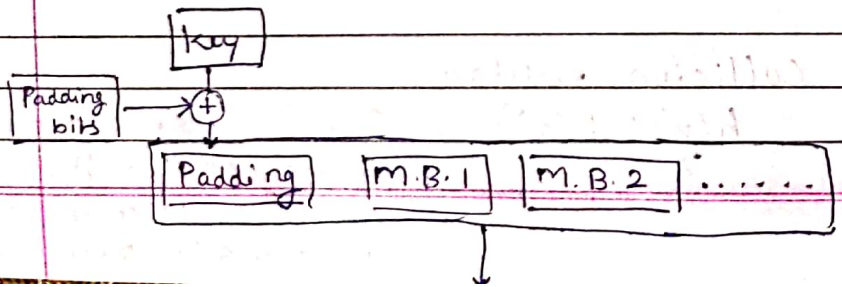
- **Nested M.A.C.:**

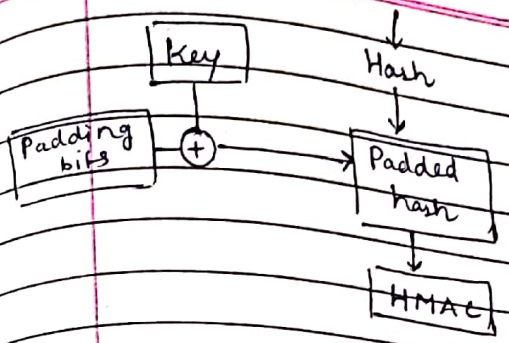
→ In this we will repeat 2 times.



→ Simple M.A.C.

→ HMAC (Hashed MAC)





→ CMAC / CBCMAC:

Chaining MAC — Code Block chaining MAC