Standard block cipher

08 March 2021 12:01

DES, 2DES, 3DES, IDES, Blowfish.

AES : Advanced Encryption Standard:

Block cipher.

Symmetric cipher

fast (6 time fast than 3 DES)

Replacement DES (AES with Large key size)

Plaintext: 128 bit

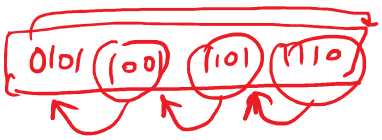
key size 128 / 192 / 256 bit key

↳ Strong

implementation : simple HL programming language like (C, C++, Java)

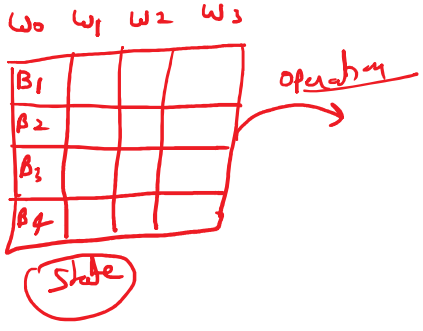
It is based on (Substitution) (Permutation) (Network)

AES perform its all computation on bytes in place of bits.



1001 1101 1110 0000

Plaintext: 128 bit → 16 byte (4 Row, 4 column) → 8 bit



AES use 10 Round (128 bit key)

AES use 12 Round (192 bit key)

AES use 14 Round (256 bit key)

} strength

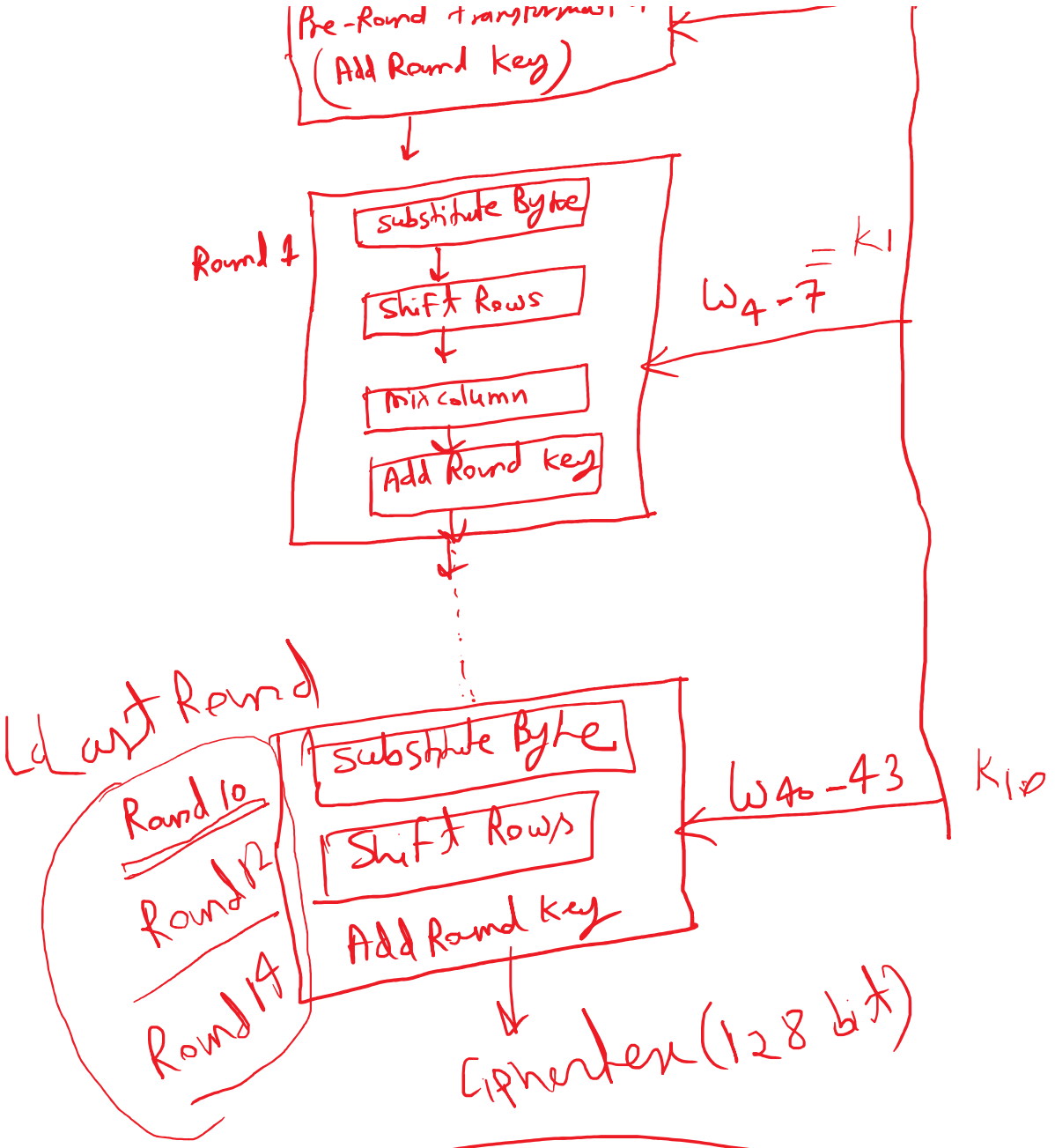
128 bit Plaintext

128 bit key

Key Expand

$W_{0-3} = K_0$

Pre-Round transformation (Add Round key)



128: Input array: (4×4 ie 16 byte) or (4 word)

1 word = 4 byte

Byte 0	S_{00}	S_{01}	S_{02}	S_{03}
Byte 1	S_{10}	S_{11}	S_{12}	S_{13}
Byte 2	S_{20}	S_{21}	S_{22}	S_{23}
Byte 3	S_{30}	S_{31}	S_{32}	S_{33}
	Word 0	w1	w2	w3

→ Byte 2 of word 3

8 bit

128 = 16 bytes

Word #

Key 128 bit : byte (8 bit) = $\frac{128}{8}$ = 16 bytes

K_0	K_4	K_8	K_{12}
K_1	K_5	K_9	K_{13}
K_2	K_6	K_{10}	K_{14}
K_3	K_7	K_{11}	K_{15}
w_0	w_1	w_2	w_3

Stable

Key Exp.

w_0 w_1 w_2 w_3

w_4 w_5 w_6 w_7

w_8 w_9 w_{10} w_{11}

- - - - - -

Data Unit in AES

(1) bit : 0 or 1

(2) byte : sequence of 8 bit

(3) $[b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7]$

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

(3) word (W): sequence of 4 bytes

$$W: [B_0\ B_1\ B_2\ B_3]$$

$$= \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

W

$$W: \begin{bmatrix} b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7 \\ b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7 \\ b_0 - - - - - - - - b_7 \\ b_0 - - - - - - - - b_7 \end{bmatrix}$$

(4) block: sequence of bit (16)

$$[b_0\ b_1\ b_2 - - - - - - b_{15}]$$

(5) State: 4x4 matrix Representation for 128 bit block.
each cell of matrix have one byte.

$S \rightarrow [w_0\ w_1\ w_2\ w_3]$

$$S \rightarrow \begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33} \end{bmatrix}$$

Round functionality:

Plaintext: $[B_0\ B_1\ B_2\ B_3 \dots B_{15}]$

128 bit

16 byte

$\frac{128 \text{ (bit)}}{8 \text{ (Byte}} = 16$ Byte

↓

State

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

State

Key (128)

↓

Key generation

$[w_0 w_1 w_2 w_3]$

↓

K_0

PreRound

R_0

Add Round key

128 bit

R_1

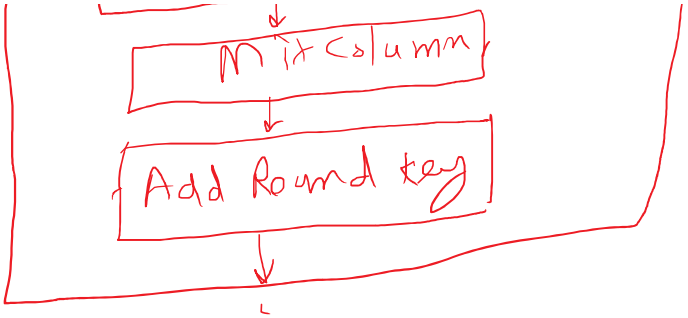
Substitute Byte

↓

Shift Rows

↓

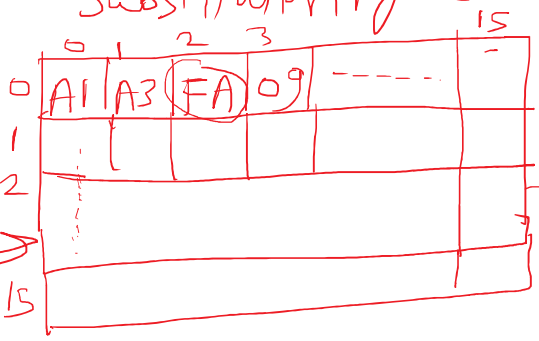
Mix Column



CT.

16 x 16 (S-Box)

Substitution Byte table



① Substitution Byte :

02	AE	1F	2A
03	01	8A	E3
51	62	71	91
52	53	54	A0

Input state

FA

Output state

first cell: 0000 0010

Row → 0

Column → 2

⑪ Shift Row

Row 0	FA	C9	D1	E2
Row 1	F1	E9	E2	E6
			AB	A6

No shifting

1 Byte shifting (left)

2 Byte shifting (left)

Row1	F1	E9	E2	E6
Row2	H0	A3	AB	A6
Row3	01	05	D1	D6

2 Byte shifting (left)

3 Byte Shifting (Left)

↓ after Row Shift

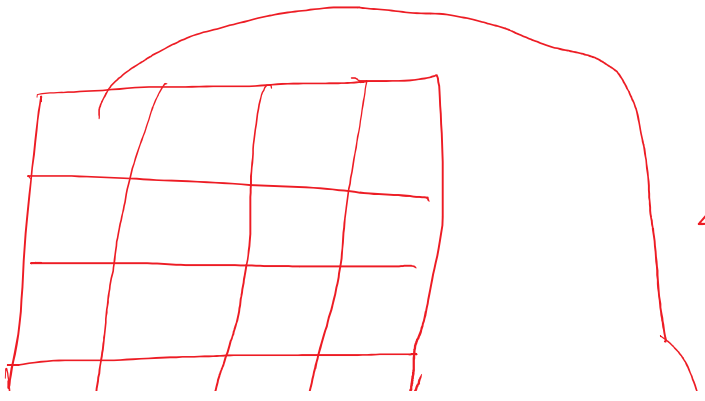
FA	C9	D1	E2
E9	E2	E6	F1
AB	A6	A0	A3
DL	01	05	D1

(11) Mix column:

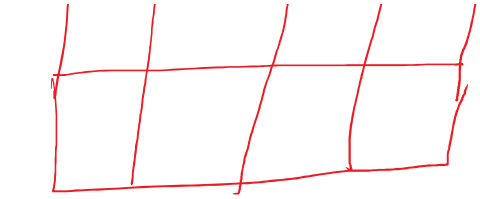
take each word / column (4 byte)

(4 × 1 matrix)

and multiply it with the constant matrix will generate output of 4 × 1 matrix.



4 × 1

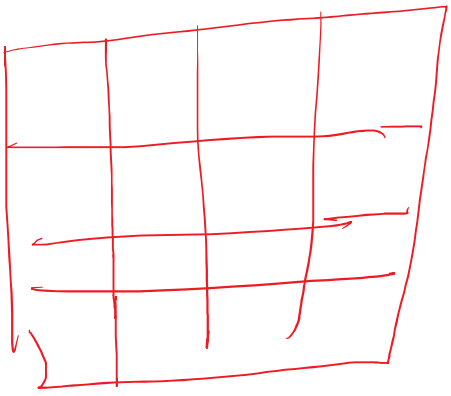


w_0 w_1 w_2 w_3

Constant Matrix

2	3	1	1
1	3	1	15
16	12	1	8
9	10	15	9

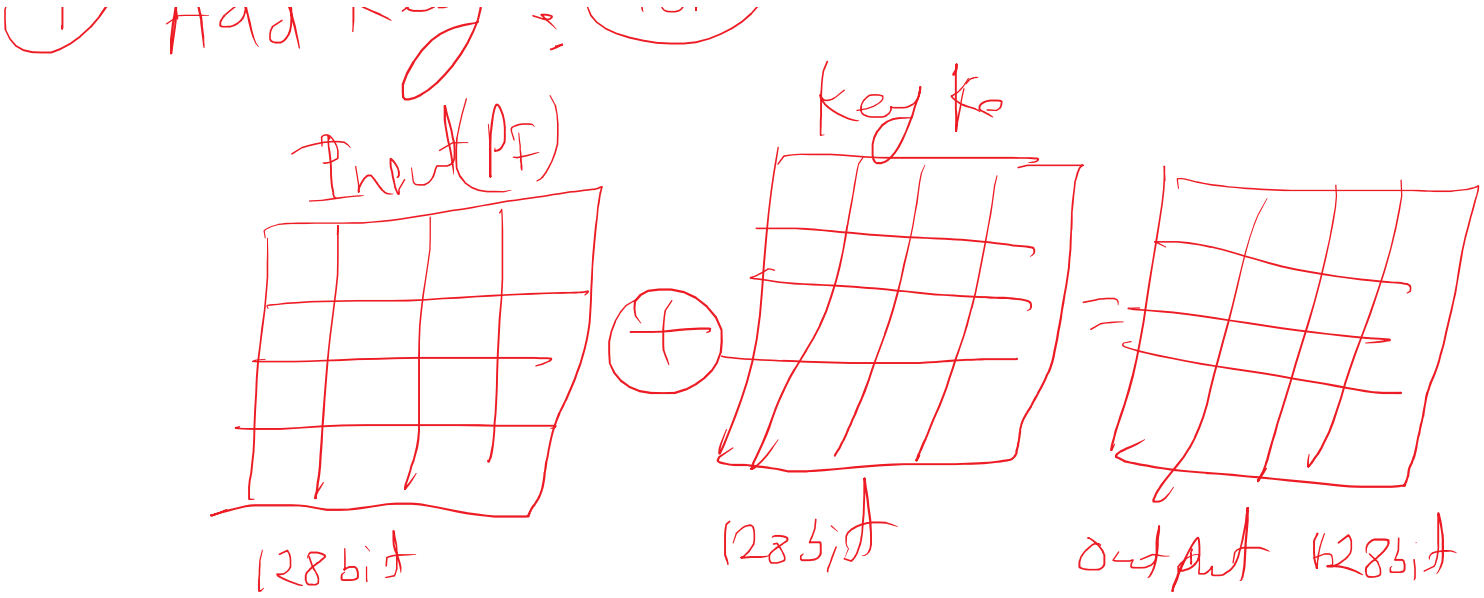
4×4 × W_1 4×1 = 4×1 output (W_1)



output state

w_0 w_1 w_2 w_3

④ Add key : XOR


Input (PT)
Key Ko
128 bit
128 bit
Output 128 bit