AES Encryption Algorithm

By
M.V. Ganesh Raju
Networking and
Security Enthusiast

ABSTRACT

We all know that as per the prevailing trend, Advanced Encryption standard (AES) is the latest and most widely used encryption algorithm.
Almost every security audit recommends using the AES for the encryption purpose.
This is because the aging Data Encryption Standard (DES) that was used earlier became too vulnerable to brute-force attacks.
So the purpose of the seminar is to present the Advanced Encryption standard.

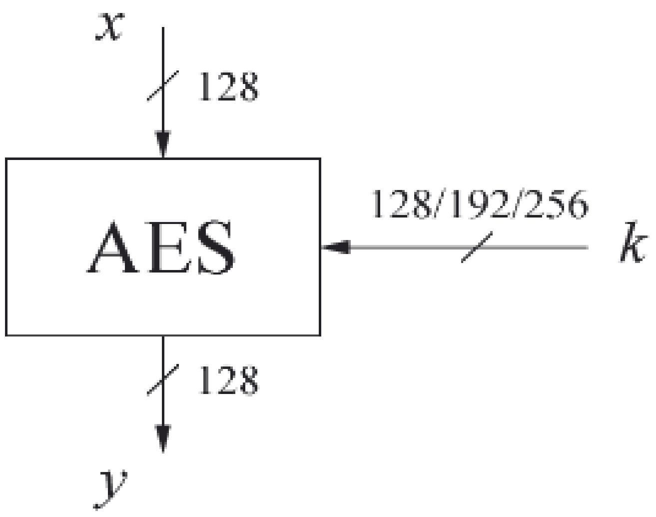
INTRODUCTION

The **Advanced Encryption Standard** (**AES**) is a most important and widely used symmetric-key encryption standard.

It uses the same key for both encryption and decryption process.

Block cipher with 128-bit block size

Three supported key lengths : 128, 192 and 256 bit



Key length (bits)	Number of rounds
128	10
192	12
256	14

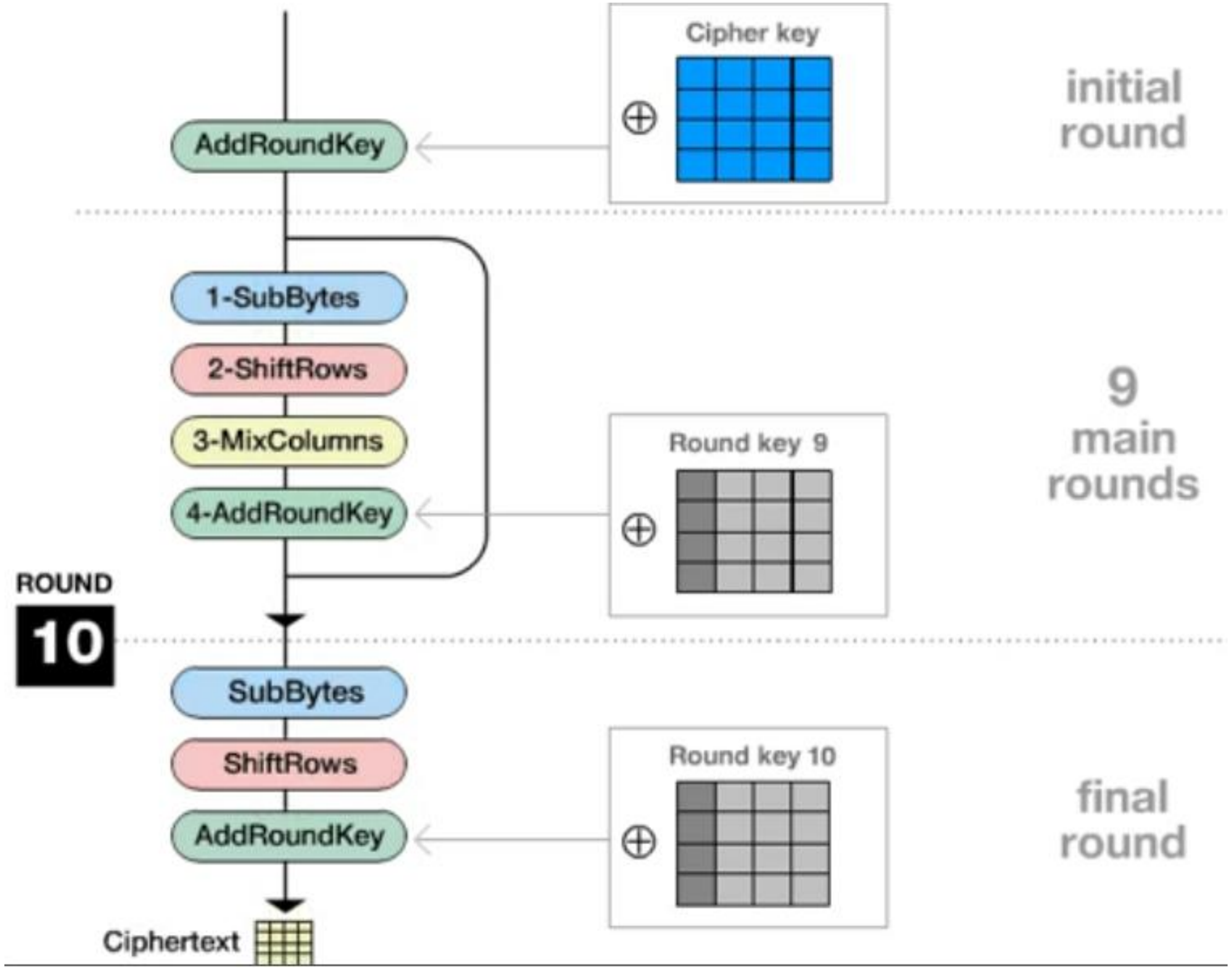
In each round, it performs four transformations, namely SubBytes(), ShiftRows(), MixColumns() and AddRoundKey().

Among the four transformations, SubBytes() and MixColumns() are used to perform simple substitution operations.

The ShiftRows() transformation is used to perform the permutation operation.

The AddRoundKey() transformation is used to perform the XOR operation in the encryption and decryption process.

The encryption process



AddRoundKey()

State

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

This is a block from the plaintext message to be encrypted.

Cipher key

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

SubBytes()

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

Fig:- Before appyling the SubBytes()

hex	y															
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-BOX / Byte Substitution table

SubBytes()

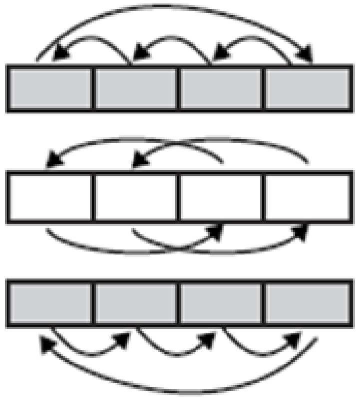
d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig:- After appyling the SubBytes()

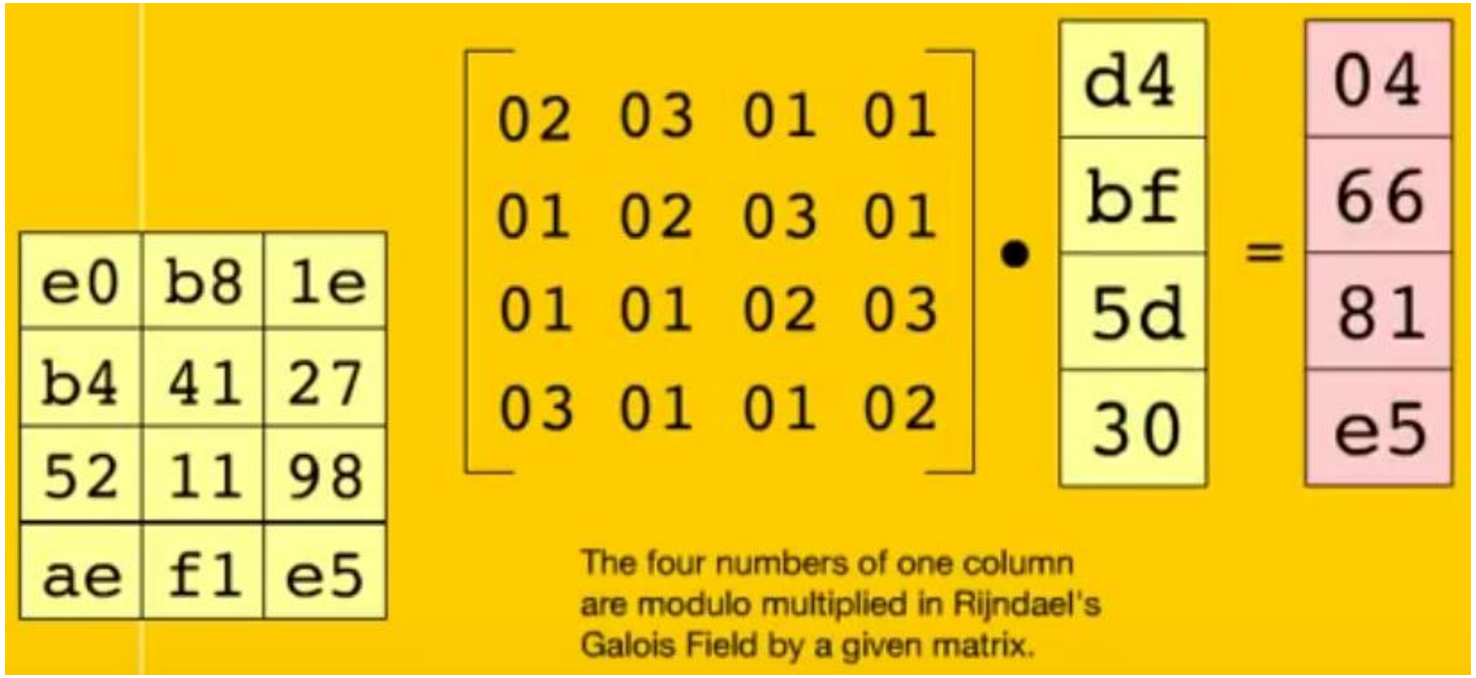
ShiftRows()

d4	e0	b8	1e	
27	bf	b4	41	rotate over 1 byte
11	98	5d	52	rotate over 2 bytes
ae	f1	e5	30	rotate over 3 bytes



d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

MixColumns()



The constant matrix that is used is based on a code which gives good mixing of the bytes within each column

Steps for finding the values:

We know that,

$$r_0 = \{02 \cdot d4\} + \{03 \cdot bf\} + \{01 \cdot 5d\} + \{01 \cdot 30\}$$

The irreducible polynomial that we have selected for the AES algorithm is $p(x) = x^8 + x^4 + x^3 + x + 1$ (Galois field)

Sl. No.	String	Polynomials $\{p(x)\}$
1.	1 0 0 0 1 1 1 0 1	$x^8 + x^4 + x^3 + x^2 + 1$
2.	1 0 1 1 1 0 1 1 1	$x^8 + x^6 + x^5 + x^4 + x^2 + x^1 + 1$
3.	1 1 1 1 1 0 0 1 1	$x^8 + x^7 + x^6 + x^5 + x^4 + x^1 + 1$
4.	1 0 1 1 0 1 0 0 1	$x^8 + x^6 + x^5 + x^3 + 1$
5.	1 1 0 1 1 1 1 0 1	$x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + 1$
6.	1 1 1 1 0 0 1 1 1	$x^8 + x^7 + x^6 + x^5 + x^2 + x^1 + 1$
7.	1 0 0 1 0 1 0 1 1	$x^8 + x^5 + x^3 + x^1 + 1$
8.	1 1 1 0 1 0 1 1 1	$x^8 + x^7 + x^6 + x^4 + x^2 + x^1 + 1$
9.	1 0 1 1 0 0 1 0 1	$x^8 + x^6 + x^5 + x^2 + 1$
10.	1 1 0 0 0 1 0 1 1	$x^8 + x^7 + x^3 + x^1 + 1$
11.	1 0 1 1 0 0 0 1 1	$x^8 + x^6 + x^5 + x^1 + 1$
12.	1 0 0 0 1 1 0 1 1	$x^8 + x^4 + x^3 + x^1 + 1$
13.	1 0 0 1 1 1 1 1 1	$x^8 + x^5 + x^4 + x^3 + x^2 + x^1 + 1$
14.	1 1 0 0 0 1 1 0 1	$x^8 + x^7 + x^3 + x^2 + 1$
15.	1 0 0 1 0 1 1 0 1	$x^8 + x^5 + x^3 + x^2 + 1$
16.	1 0 1 0 1 1 1 1 1	$x^8 + x^6 + x^4 + x^3 + x^2 + x^1 + 1$
17.	1 1 1 1 1 1 0 0 1	$x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1$
18.	1 1 1 0 0 0 0 1 1	$x^8 + x^7 + x^6 + x^1 + 1$
19.	1 0 0 1 1 1 0 0 1	$x^8 + x^5 + x^4 + x^3 + 1$
20.	1 1 0 1 0 1 0 0 1	$x^8 + x^7 + x^5 + x^3 + 1$
21.	1 1 0 0 0 0 1 1 1	$x^8 + x^7 + x^2 + x^1 + 1$
22.	1 1 0 1 1 0 0 0 1	$x^8 + x^7 + x^5 + x^4 + 1$
23.	1 0 1 0 0 1 1 0 1	$x^8 + x^6 + x^3 + x^2 + 1$
24.	1 1 1 0 0 1 1 1 1	$x^8 + x^7 + x^6 + x^3 + x^2 + x^1 + 1$
25.	1 1 1 0 1 1 1 0 1	$x^8 + x^7 + x^6 + x^4 + x^3 + x^2 + 1$
26.	1 1 0 1 0 0 0 1 1	$x^8 + x^7 + x^5 + x^1 + 1$
27.	1 1 1 1 1 0 1 0 1	$x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$
28.	1 1 0 0 1 1 1 1 1	$x^8 + x^7 + x^4 + x^3 + x^2 + x^1 + 1$
29.	1 0 1 1 1 1 0 1 1	$x^8 + x^6 + x^5 + x^4 + x^3 + x^1 + 1$
30.	1 0 1 1 1 0 0 0 1	$x^8 + x^6 + x^5 + x^4 + 1$

Step 1:

$\underline{02 \cdot d4}$

$= 10 \cdot 11010100$

$= (11 \text{ XOR } 01) \cdot 11010100$

$= 110101000 \text{ XOR } 11010100$

$= 10101000$

The output is XOR with 00011011

$$\begin{array}{r} 10101000 \\ \underline{00011011} \\ \underline{10110011} \end{array}$$

$\therefore \quad \underline{02 \cdot d4 = 10110011}$

Step 2:

03.bf

= 11 · 10111111

= (10 XOR 01) · 10111111

= 101111110

XOR 010111111

011000001

The o/p is XOR with 00011011

```
    011000001
XOR  00011011
    ---------
     11011010
```

∴ 03.bf = 11011010

Step 3:

$\underline{01 \cdot 5d}$

$= 01011101$

$\underline{01 \cdot 30}$

$= 00110000$

Step 4:

$r_2 = \{02 \cdot d4\} + \{03 \cdot bf\} + \{01 \cdot 5d\} + \{01 \cdot 30\}$

$= 10110011$ XOR 11011010 XOR

01011101 XOR 00110000

$= 00000100$

$= 04$ (in Hex)

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

So, this is the output of MixColumns() after similarly calculating for all the values

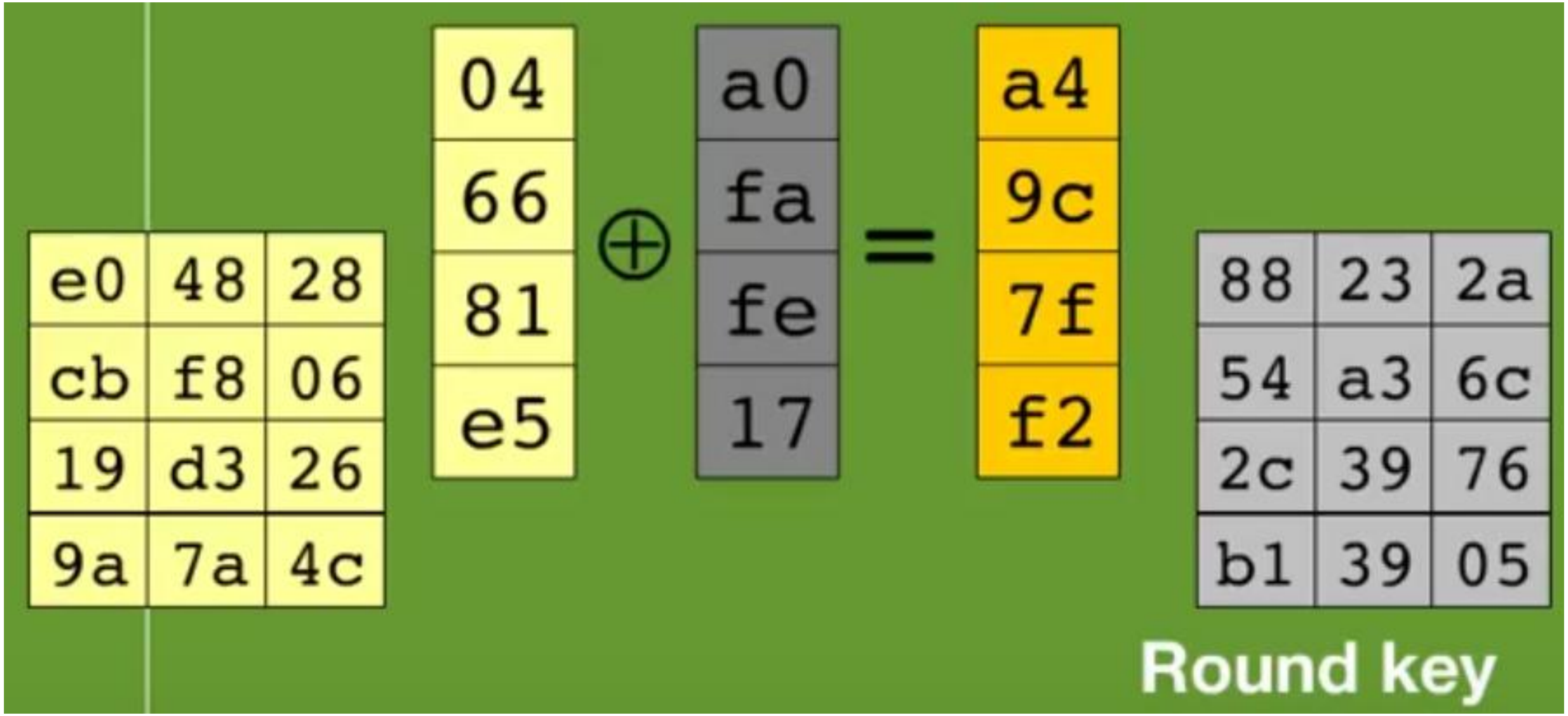
AddRoundKey()

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

Round key

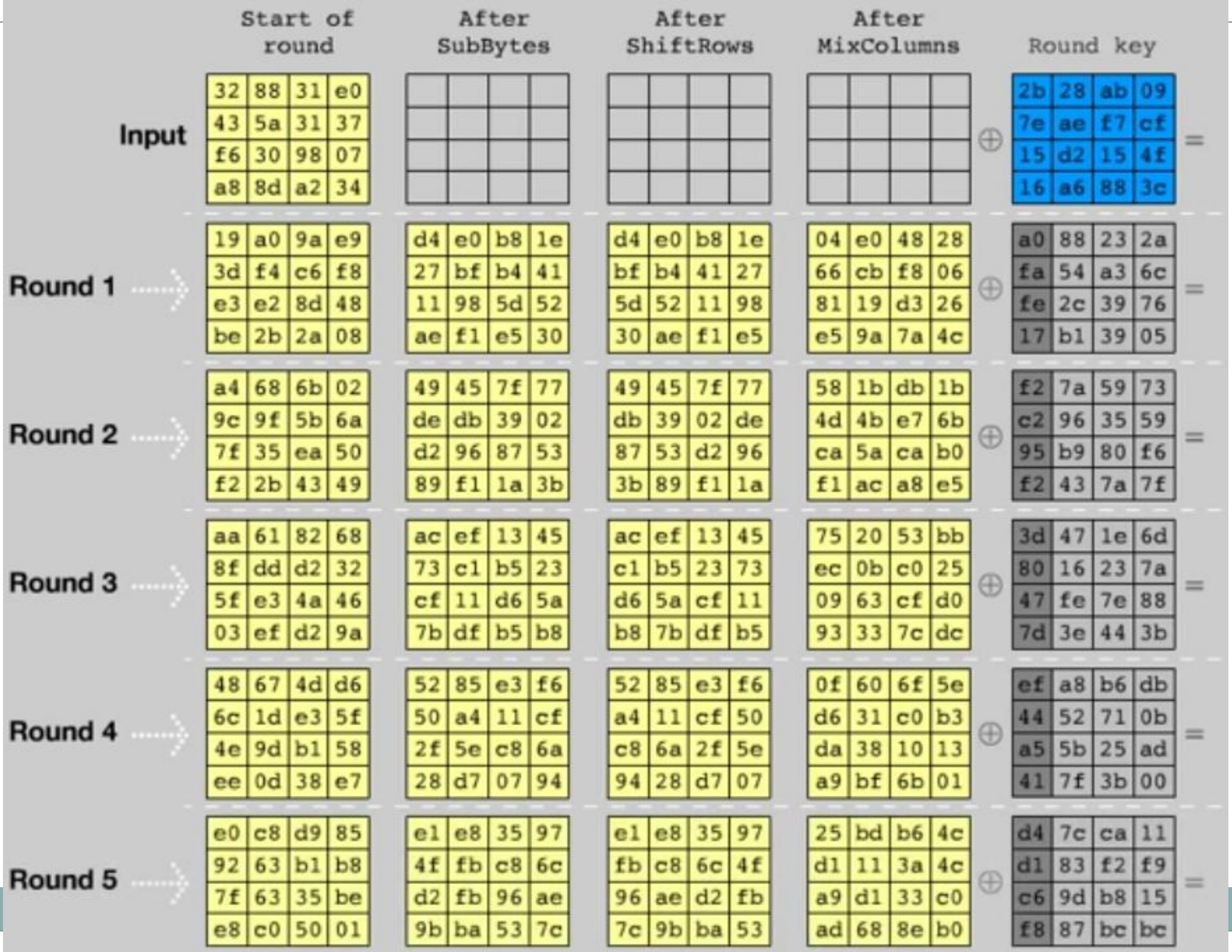
AddRoundKey()



Final output after AddRoundKey()

a4	68	6b	02
9c	9f	5b	6a
7f	35	ea	50
f2	2b	43	49

These transformations are applied to the state for 9 more rounds.
The final round does not include the MixColumns transformation.

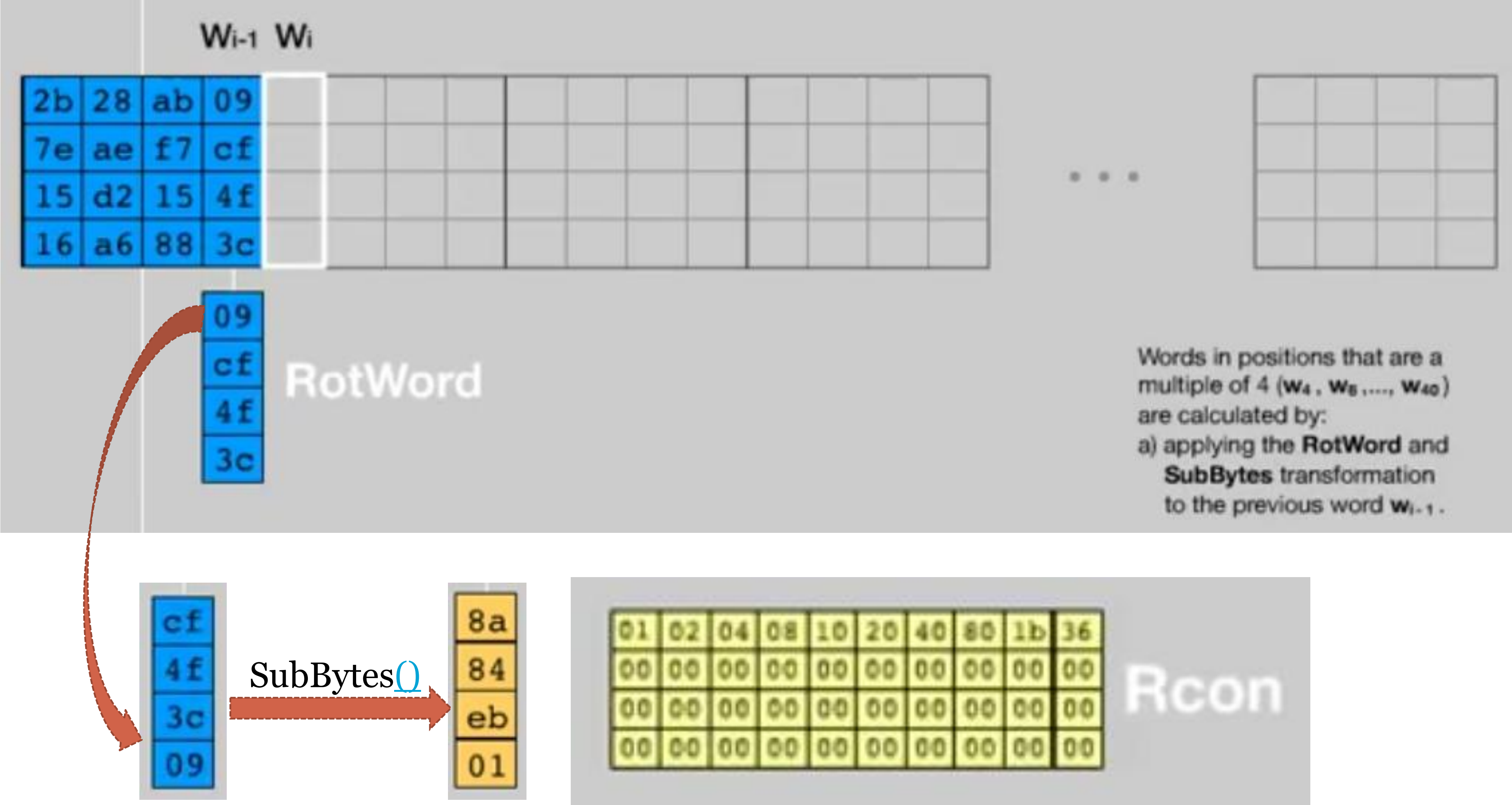


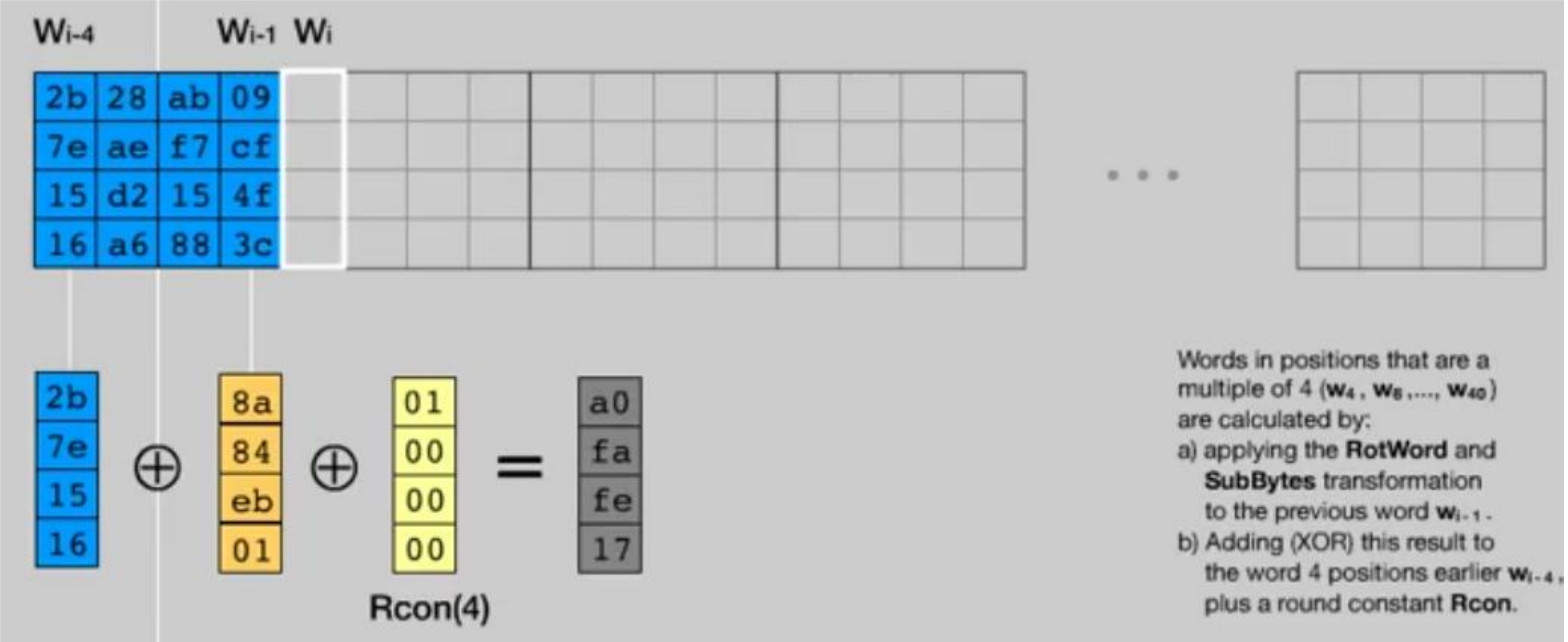
Start of round
After SubBytes
After ShiftRows
After MixColumns
Round key
Input
32 88 31 e0
43 5a 31 37
f6 30 98 07
a8 8d a2 34
2b 28 ab 09
7e ae f7 cf
15 d2 15 4f
16 a6 88 3c
Round 1
19 a0 9a e9
3d f4 c6 f8
e3 e2 8d 48
be 2b 2a 08
d4 e0 b8 1e
27 bf b4 41
11 98 5d 52
ae f1 e5 30
d4 e0 b8 1e
bf b4 41 27
5d 52 11 98
30 ae f1 e5
04 e0 48 28
66 cb f8 06
81 19 d3 26
e5 9a 7a 4c
a0 88 23 2a
fa 54 a3 6c
fe 2c 39 76
17 b1 39 05
Round 2
a4 68 6b 02
9c 9f 5b 6a
7f 35 ea 50
f2 2b 43 49
49 45 7f 77
de db 39 02
d2 96 87 53
89 f1 1a 3b
49 45 7f 77
db 39 02 de
87 53 d2 96
3b 89 f1 1a
58 1b db 1b
4d 4b e7 6b
ca 5a ca b0
f1 ac a8 e5
f2 7a 59 73
c2 96 35 59
95 b9 80 f6
f2 43 7a 7f
Round 3
aa 61 82 68
8f dd d2 32
5f e3 4a 46
03 ef d2 9a
ac ef 13 45
73 c1 b5 23
cf 11 d6 5a
7b df b5 b8
ac ef 13 45
c1 b5 23 73
d6 5a cf 11
b8 7b df b5
75 20 53 bb
ec 0b c0 25
09 63 cf d0
93 33 7c dc
3d 47 1e 6d
80 16 23 7a
47 fe 7e 88
7d 3e 44 3b
Round 4
48 67 4d d6
6c 1d e3 5f
4e 9d b1 58
ee 0d 38 e7
52 85 e3 f6
50 a4 11 cf
2f 5e c8 6a
28 d7 07 94
52 85 e3 f6
a4 11 cf 50
c8 6a 2f 5e
94 28 d7 07
0f 60 6f 5e
d6 31 c0 b3
da 38 10 13
a9 bf 6b 01
ef a8 b6 db
44 52 71 0b
a5 5b 25 ad
41 7f 3b 00
Round 5
e0 c8 d9 85
92 63 b1 b8
7f 63 35 be
e8 c0 50 01
e1 e8 35 97
4f fb c8 6c
d2 fb 96 ae
9b ba 53 7c
e1 e8 35 97
fb c8 6c 4f
96 ae d2 fb
7c 9b ba 53
25 bd b6 4c
d1 11 3a 4c
a9 d1 33 c0
ad 68 8e b0
d4 7c ca 11
d1 83 f2 f9
c6 9d b8 15
f8 87 bc bc

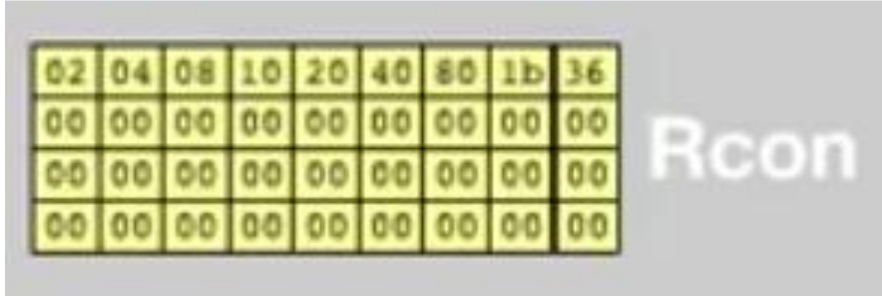
	Start of round	After SubBytes	After ShiftRows	After MixColumns		Round key	
Round 6	f1 c1 7c 5d 00 92 c8 b5 6f 4c 8b d5 55 ef 32 0c	a1 78 10 4c 63 4f e8 d5 a8 29 3d 03 fc df 23 fe	a1 78 10 4c 4f e8 d5 63 3d 03 a8 29 fe fc df 23	4b 2c 33 37 86 4a 9d d2 8d 89 f4 18 6d 80 e8 d8	⊕	6d 11 db ca 88 0b f9 00 a3 3e 86 93 7a fd 41 fd	=
Round 7	26 3d e8 fd 0e 41 64 d2 2e b7 72 8b 17 7d a9 25	f7 27 9b 54 ab 83 43 b5 31 a9 40 3d f0 ff d3 3f	f7 27 9b 54 83 43 b5 ab 40 3d 31 a9 3f f0 ff d3	14 46 27 34 15 16 46 2a b5 15 56 d8 bf ec d7 43	⊕	4e 5f 84 4e 54 5f a6 a6 f7 c9 4f dc 0e f3 b2 4f	=
Round 8	5a 19 a3 7a 41 49 e0 8c 42 dc 19 04 b1 1f 65 0c	be d4 0a da 83 3b e1 64 2c 86 d4 f2 c8 c0 4d fe	be d4 0a da 3b e1 64 83 d4 f2 2c 86 fe c8 c0 4d	00 b1 54 fa 51 c8 76 1b 2f 89 6d 99 d1 ff cd ea	⊕	ea b5 31 7f d2 8d 2b 8d 73 ba f5 29 21 d2 60 2f	=
Round 9	ea 04 65 85 83 45 5d 96 5c 33 98 b0 f0 2d ad c5	87 f2 4d 97 ec 6e 4c 90 4a c3 46 e7 8c d8 95 a6	87 f2 4d 97 6e 4c 90 ec 46 e7 4a c3 a6 8c d8 95	47 40 a3 4c 37 d4 70 9f 94 e4 3a 42 ed a5 a6 bc	⊕	ac 19 28 57 77 fa d1 5c 66 dc 29 00 f3 21 41 6e	=
Round 10	eb 59 8b 1b 40 2e a1 c3 f2 38 13 42 1e 84 e7 d2	e9 cb 3d af 09 31 32 2e 89 07 7d 2c 72 5f 94 b5	e9 cb 3d af 31 32 2e 09 7d 2c 89 07 b5 72 5f 94		⊕	d0 c9 e1 b6 14 ee 3f 63 f9 25 0c 0c a8 89 c8 a6	=
Output	39 02 dc 19 25 dc 11 6a 84 09 85 0b 1d fb 97 32						

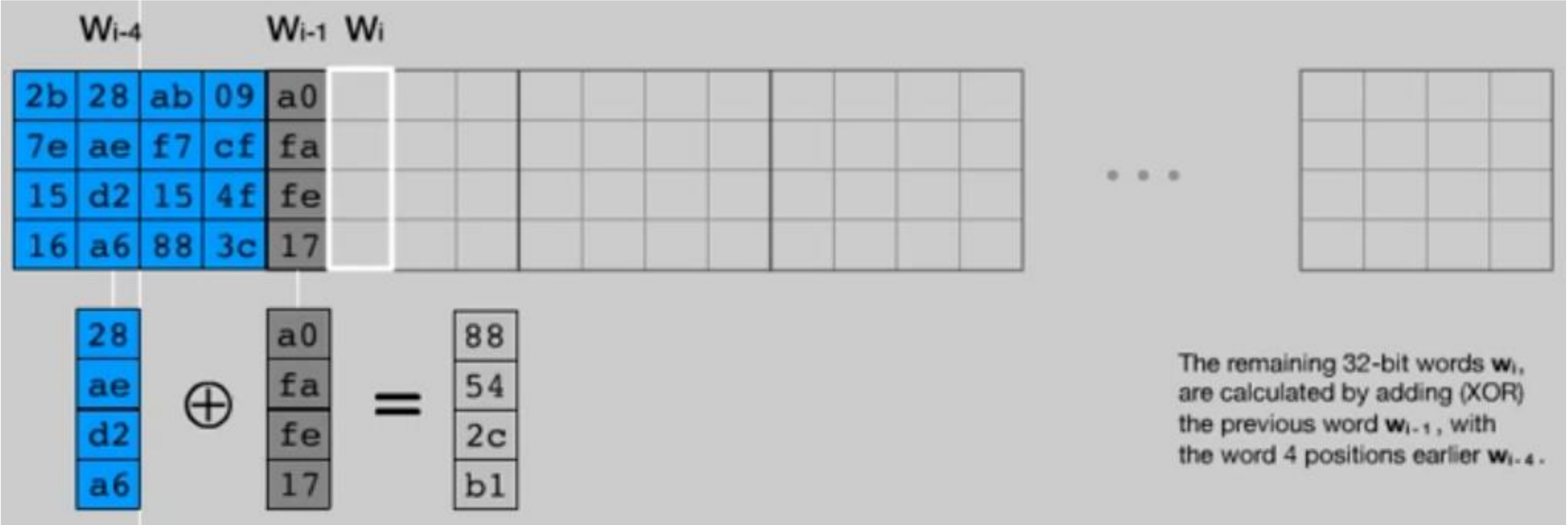
Ciphertext

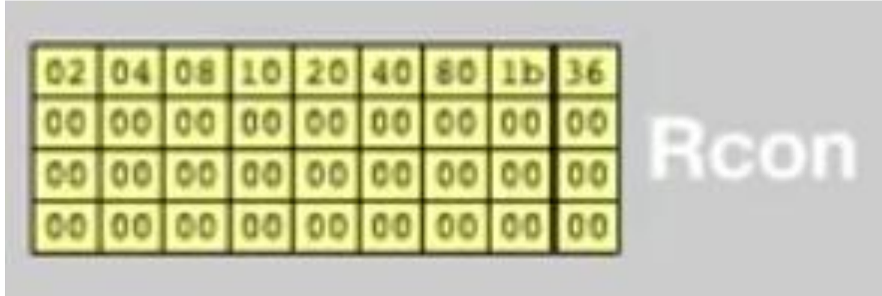
Key scheduling algorithm




Wi-4
Wi-1 Wi
2b 28 ab 09
7e ae f7 cf
15 d2 15 4f
16 a6 88 3c
2b 7e 15 16
8a 84 eb 01
01 00 00 00
a0 fa fe 17
Rcon(4)
Words in positions that are a multiple of 4 (w4 , w8 ,..., w40) are calculated by:
a) applying the RotWord and SubBytes transformation to the previous word wi-1 .
b) Adding (XOR) this result to the word 4 positions earlier wi-4 , plus a round constant Rcon.


02 04 08 10 20 40 80 1b 36
00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00
Rcon


Wi-4
Wi-1 Wi
2b 28 ab 09 a0
7e ae f7 cf fa
15 d2 15 4f fe
16 a6 88 3c 17
28 ae d2 a6
a0 fa fe 17
88 54 2c b1
The remaining 32-bit words wi, are calculated by adding (XOR) the previous word wi-1, with the word 4 positions earlier wi-4.


02 04 08 10 20 40 80 1b 36
00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00
Rcon


2b 28 ab 09 a0 88 23 2a f2 7a 59 73 3d 47 1e 6d
7e ae f7 cf fa 54 a3 6c c2 96 35 59 80 16 23 7a
15 d2 15 4f fe 2c 39 76 95 b9 80 f6 47 fe 7e 88
16 a6 88 3c 17 b1 39 05 f2 43 7a 7f 7d 3e 44 3b
d0 c9 e1 b6
14 ee 3f 63
f9 25 0c 0c
a8 89 c8 a6
Cipher key
Round key 1
Round key 2
Round key 3
Round key 10

SBOX Generation Example

- Suppose we begin with (hexadecimal) {53}, which is 01010011 in binary.
- The corresponding field element is:

$$x^6 + x^4 + x + 1.$$

- The multiplicative inverse (in F_{2^8}) can be shown to be

$$x^7 + x^6 + x^3 + x.$$

Therefore, in binary notation, we have

$$(a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0) = (11001010).$$

- Next, we compute

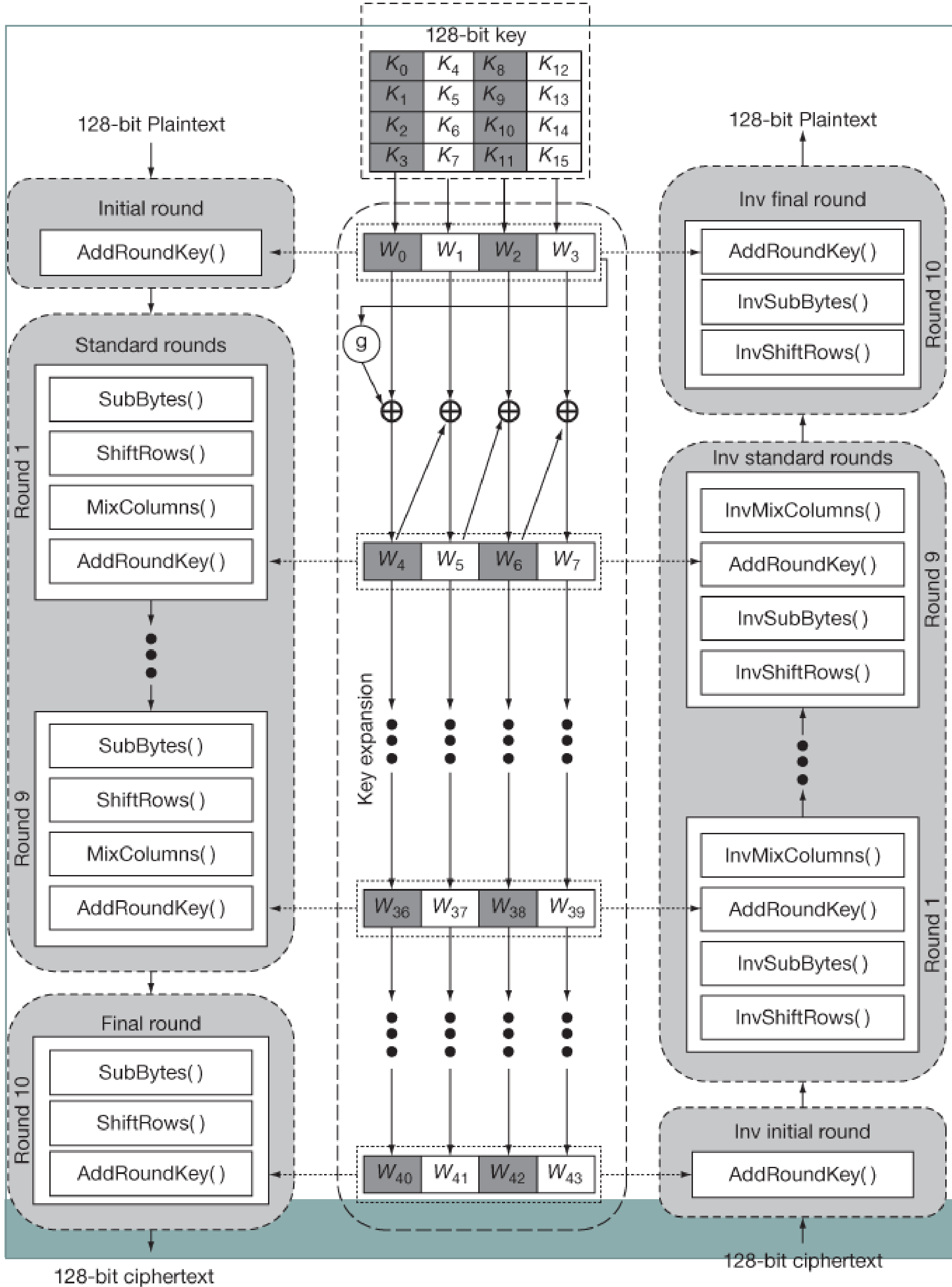
$$\begin{aligned} b_0 &= a_0 + a_4 + a_5 + a_6 + a_7 + c_0 \bmod 2 \\ &= 0 + 0 + 0 + 1 + 1 + 1 \bmod 2 \\ &= 1 \\ b_1 &= a_1 + a_5 + a_6 + a_7 + a_0 + c_1 \bmod 2 \\ &= 1 + 0 + 1 + 1 + 0 + 1 \bmod 2 \\ &= 0 \\ &\vdots \end{aligned}$$

- The result is: $(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) = (11101101)$, which is $\{ED\}$ in hexadecimal notation.

Fig:- SBOX

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

A summary of the encryption and decryption process



A Simple implementation of AES

These are the steps I have followed to implement AES-128

1. Python is by default installed in Linux. To check, type *python -v*
2. In linux, I had installed the PyCrypto – The Python Cryptography Toolkit by typing *pip install pycrypto*
3. Then we type python to go to the python command line.
4. In the prompt we type *from Crypto.Cipher import AES*

```
ganesh@ubuntu:~$ python
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from Crypto.Cipher import AES
```

Encryption

5. Next we give the key and plain text and generate the cipher using *AES.new(key)* method and ciphertext using *cipher.encrypt(plain)* method.
6. We encode the ciphertext to hex using encode function (*ciphertext.encode("hex")*)because in binary it is very long (128 bits).

```
>>> key = "Sixteen byte key"
>>> plain = "Secret: 16 bytes"
>>> cipher = AES.new(key)
>>> ciphertext = cipher.encrypt(plain)
>>> print ciphertext.encode("hex")
433811598181fed6d59e265249f8c6a8
```

Decryption

7. Then we decrypt the cipher using *cipher.decrypt(ciphertext)* method.

```
>>> cipher.decrypt(ciphertext)
'Secret: 16 bytes'
```

8. A one bit change in the key changes half the output bits.

```
>>> key = "Sixteen byte kex"
>>> cipher = AES.new(key)
>>> cipher.encrypt("Secret: 16 bytes").encode("hex")
'91d3ec81c4abf91d68fc026353f26d7f'
>>>
>>> key = "Sixteen byte key"
>>> cipher = AES.new(key)
>>> cipher.encrypt("Secret: 16 bytet").encode("hex")
'90c106728883ece4a2470a352c0865d2'
```

This property is called *'Confusion'*

8. A one bit change in the plaintext changes half the output bits. (Diffusion).

```
>>> from Crypto.Cipher import AES
>>> cipher = AES.new(key)
>>> cipher.encrypt("Secret: 16 bytes").encode("hex")
'433811598181fed6d59e265249f8c6a8'
>>> cipher.encrypt("Secret: 16 bytet").encode("hex")
'90c106728883ece4a2470a352c0865d2'
```

References

1. AES Encryption algorithm https://www.youtube.com/watch?v=evjFwDRTmV0
2. Understanding AES Mix-Columns Transformation Calculation, Kit Choy Xintong, University of Wollongong, Year 3 Student www.angelfire.com/biz7/atleast/mix_columns.pdf
3. Cryptography and Network Security- NPTEL Course https://onlinecourses.nptel.ac.in/noc18_cs07/announcements?force=true
4. Cryptography and Network Security by Ajay Kumar; S. Bose Published by Pearson Education India, 2016
5. Cryptography with Python, Sam Bowne
6. Cryptography and Network Security, 5/e, William Stallings

Thank you

By Malyala Venkata Ganesh Raju, Visakhapatnam, AP, India