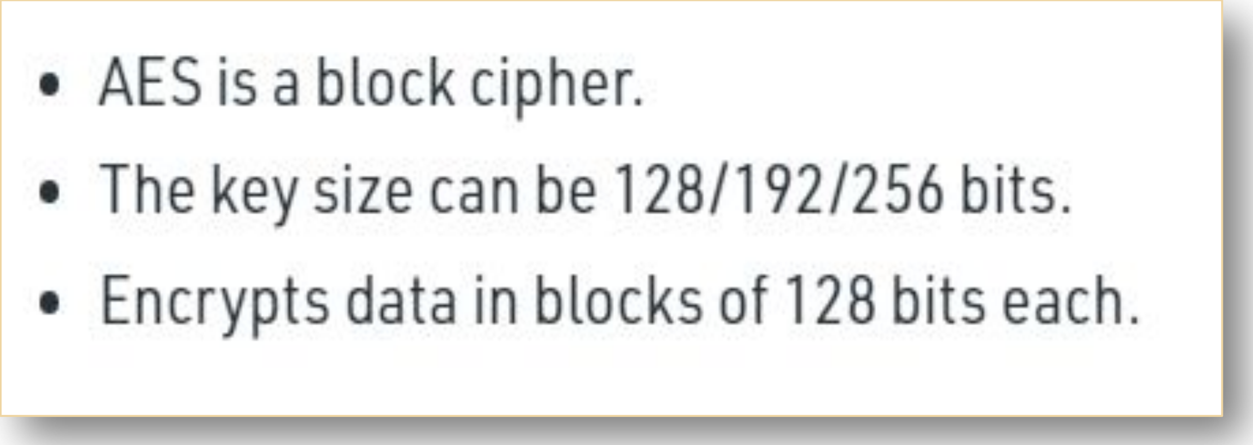
AES Algorithm

- **AES** stands for **Advanced Encryption Standard** and is a majorly used symmetric encryption algorithm.
- It is mainly used for encryption and protection of electronic data.
- It was used as the replacement of DES(Data encryption standard) as it is much faster and better than DES.
- AES consists of three block ciphers and these ciphers are used to provide encryption of data.



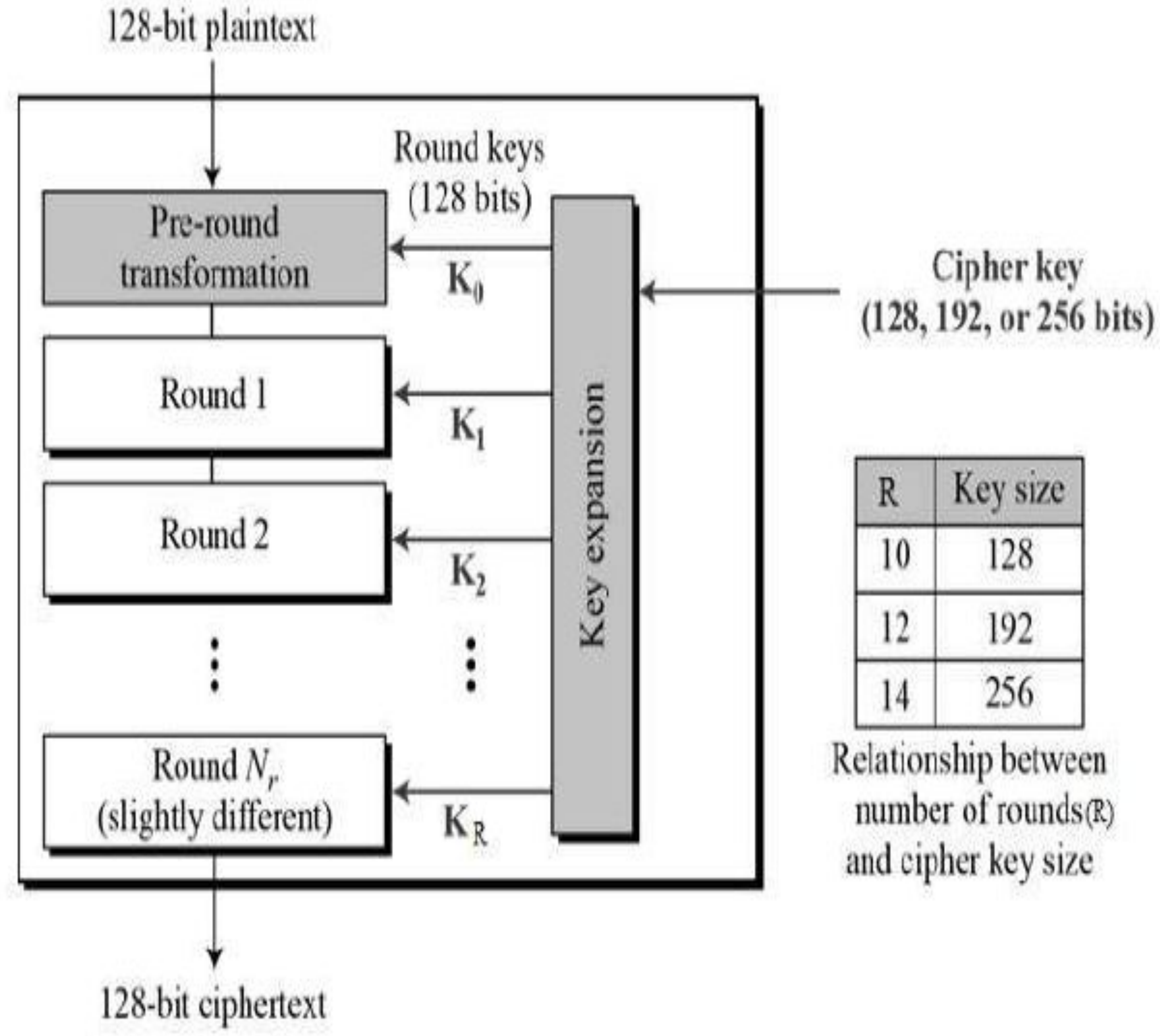
- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

Rounds	No. of bits in key	
10	128	**AES 128 Version**
12	192	**AES 192 Version**
14	256	**AES 256 Version**

128 bits i.e. 16 byte = 4 words

Since, 1 word = 32 bits

Working of AES Algorithm:



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

No. of keys generated by key expansion algorithm = No. of rounds + 1

Concepts to be known:

→ 128 bits i.e. 16 byte = 4 words

Since, 1 word = 32 bits

1 byte	Group of 8 bits
1 Words	4 bytes = 32 bits
Block Size =	128 bit data

State:

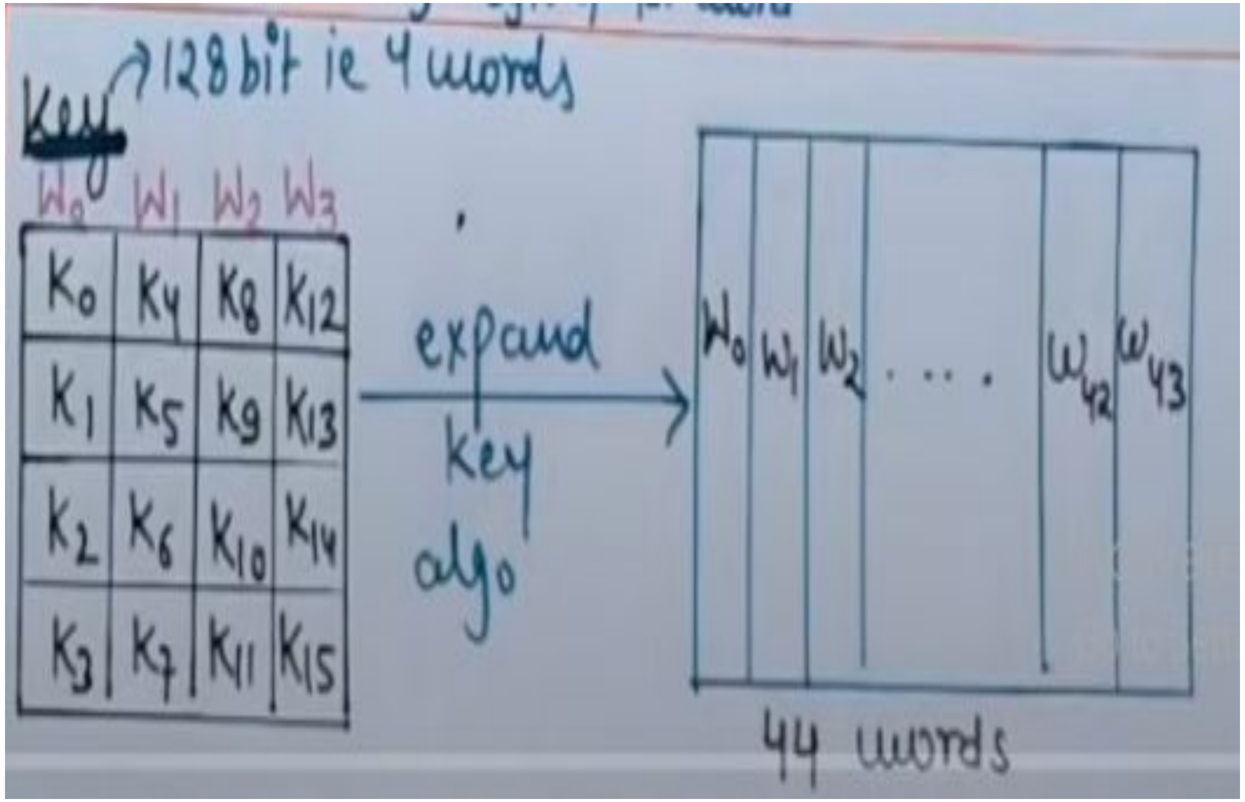
16 bytes (4 x 4).

Basically. It stores the intermediate result in matrix format after each step process.

Input matrix: 4 x 4 i.e. 16 bytes i.e. 128 bits OR 4 words

1 byte	1 byte	1 byte	1 byte
1 byte	1 byte	1 byte	1 byte
1 byte	1 byte	1 byte	1 byte
1 byte	1 byte	1 byte	1 byte

1 word = 4 bytes



Key → 128 bit ie 4 words

W_0	W_1	W_2	W_3
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}

State matrix: 4 x 4

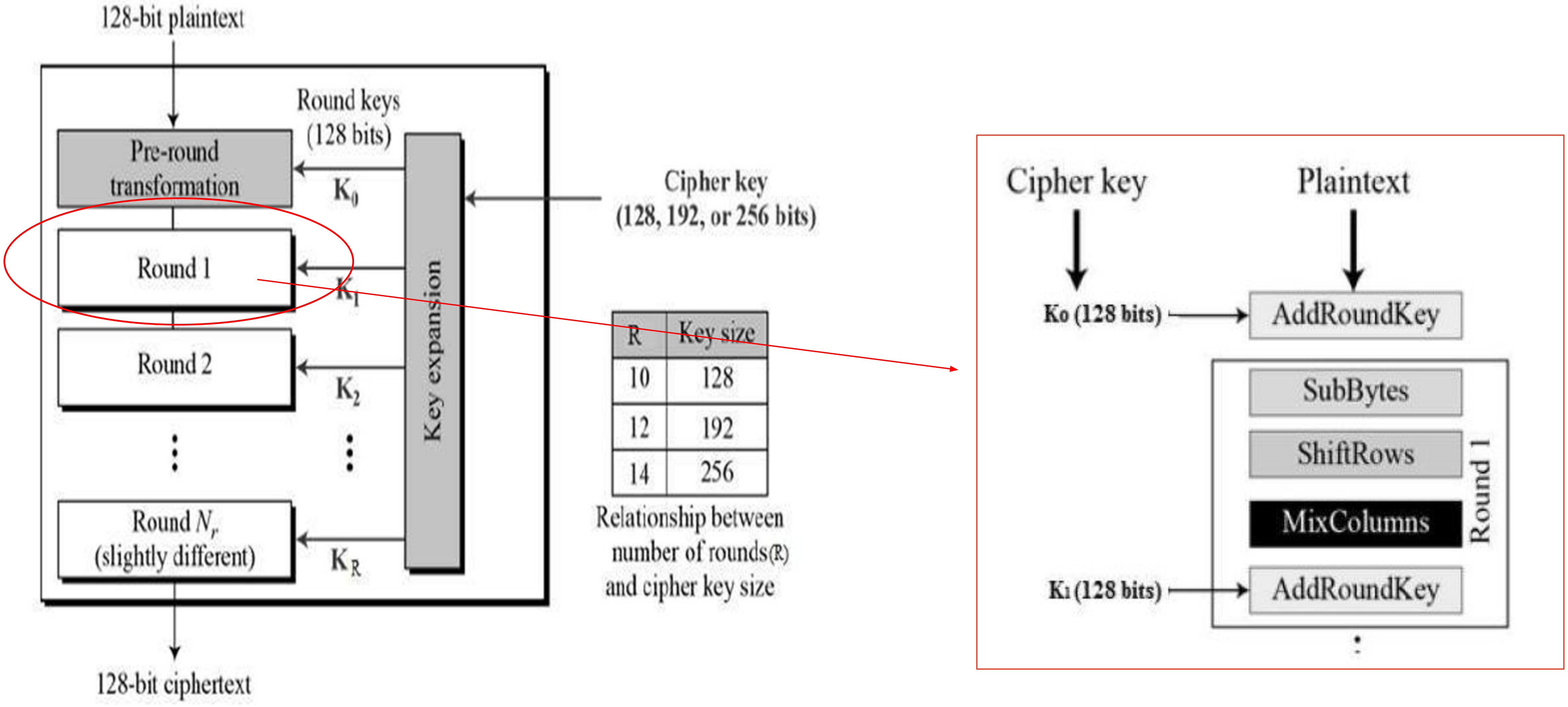
→ $[W_0 , W_1 , W_2 , W_3]$

1st byte of 0th word

2nd byte of 1st word

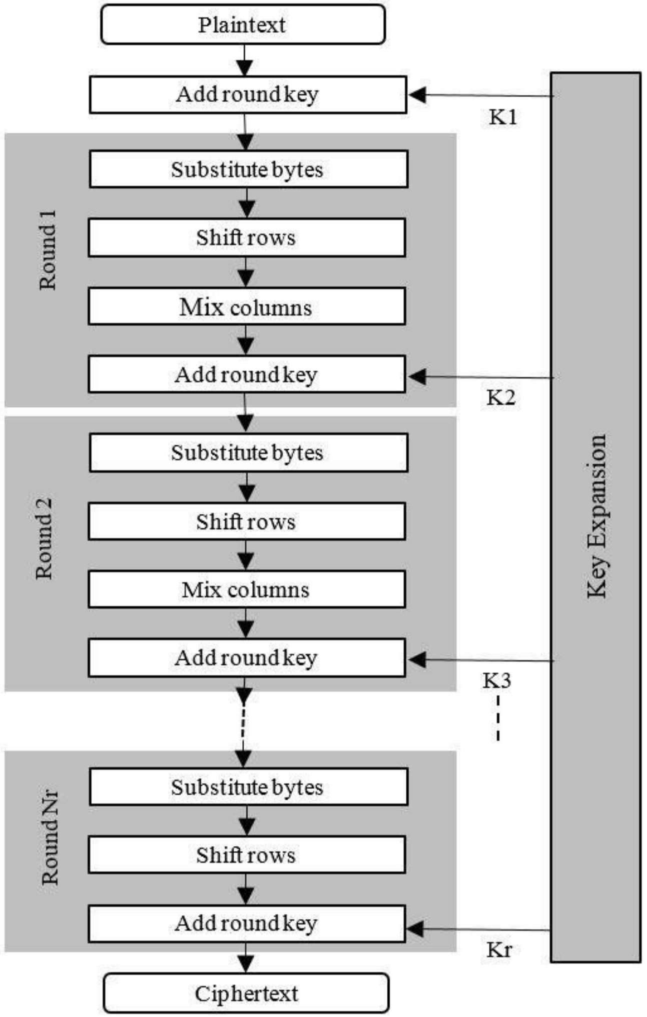
3rd byte of 2nd word

Rounds and its Transformation

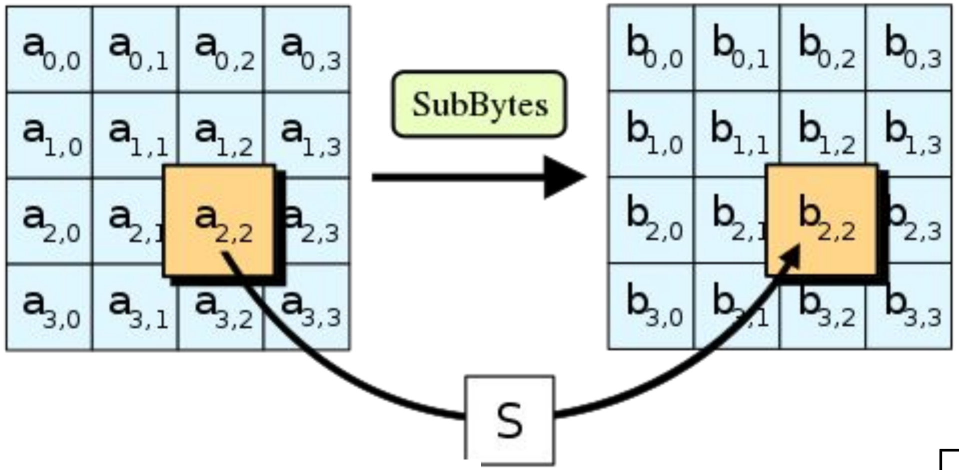


R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size


Plaintext
Add round key
K1
Round 1
Substitute bytes
Shift rows
Mix columns
Add round key
K2
Round 2
Substitute bytes
Shift rows
Mix columns
Add round key
K3
Round Nr
Substitute bytes
Shift rows
Add round key
Kr
Ciphertext
Key Expansion

1. Sub- bytes



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S

The *state* array is replaced with a SubByte using an 8-bit substitution box.
This S- box consist of hexa-decimal value i.e. 0 to 9 and A to F.

Eg. →

0000 0010 0 2

Row Column

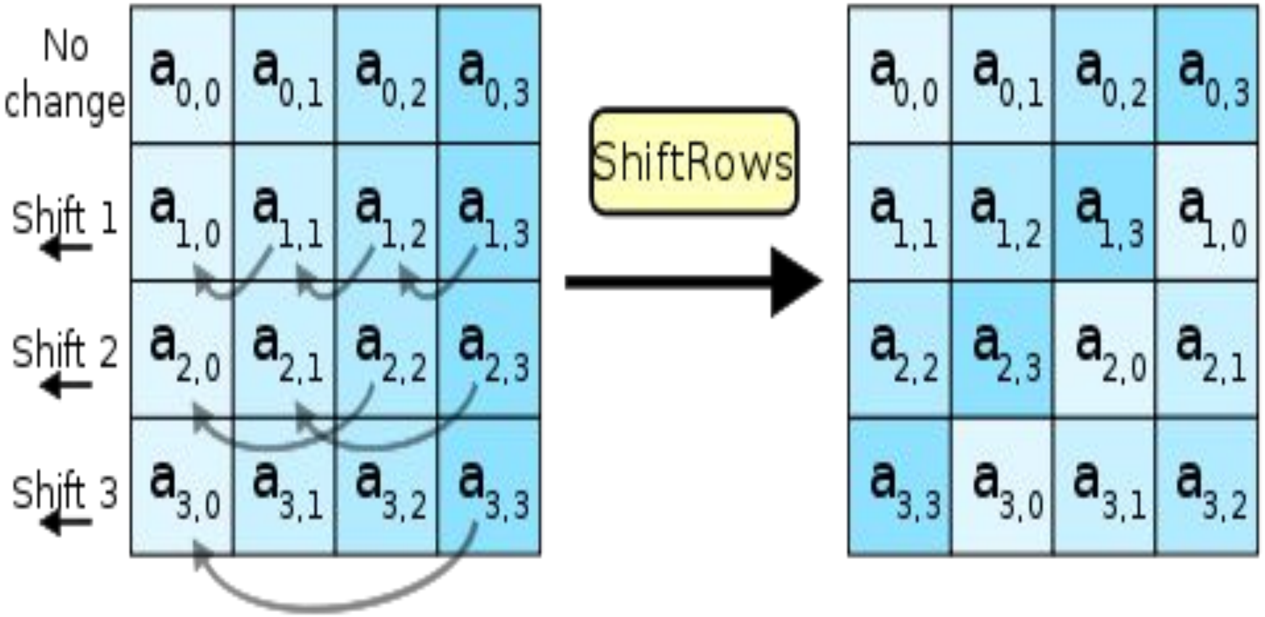
3 → 0000 0011

	0	1	2	9	A	....F
0			3			
1						
. . 9						
A						
. . F						

2. Shift Row

Shifting is done by left.
No. of shifting is depended upon the row of the state matrix.

In terms of row,
0th – no shifting
1st – 1 byte shifting
2nd - 2 byte shifting
3rd – 3 byte shifting



$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

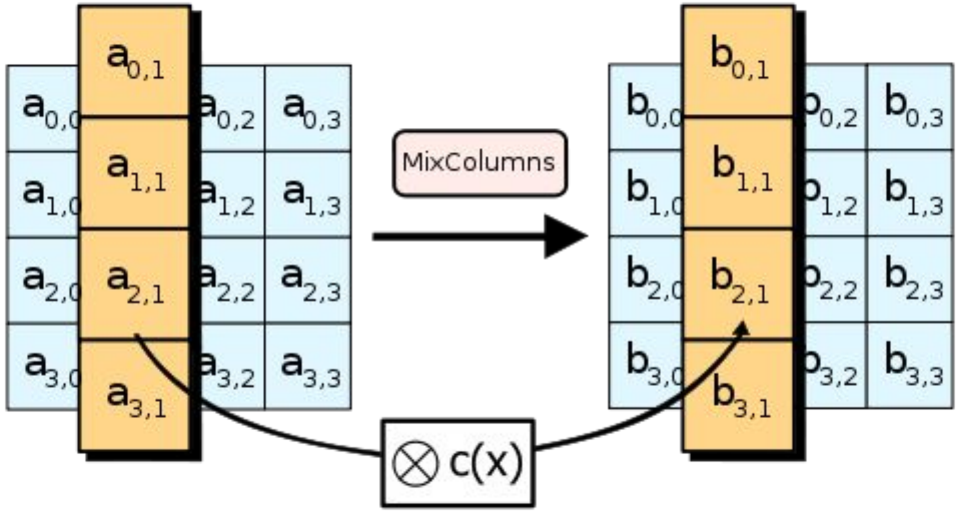
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

3. Mix Column

$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} \qquad 0 \leq j \leq 3$$

Constant Matrix
(4 x 4)

One column of
state (4 x 1)

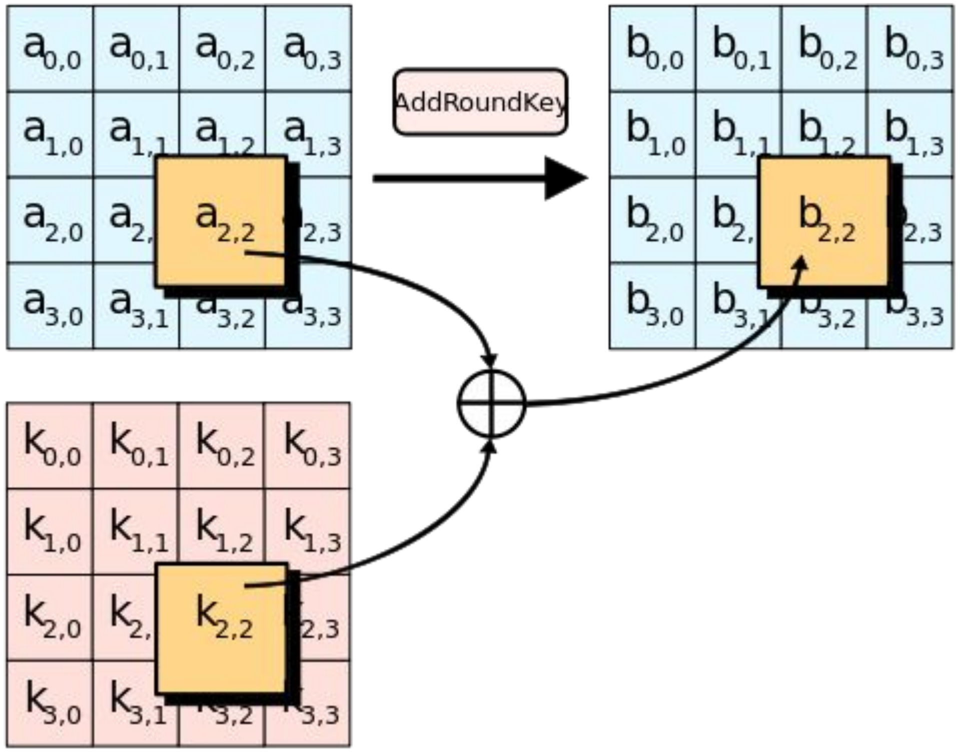


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes$ c(x)
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

Note: This step will be not performed in the last round.

4. Add Round Key


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$\oplus$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data.

Note: And the resultant matrix will be send to other round. Same process for other rounds as well. (Until now, the process was for round 1)

Characteristics

- AES has keys of three lengths which are of 128, 192, 256 bits.
- It is flexible and has implementation for software and hardware.
- It provides high security and can prevent many attacks.
- It doesn’t have any copyright so it can be easily used globally.
- It consists of 10 rounds of processing for 128 bit keys.

Advantages

- It can be implemented on both hardware and software.
- It provides high security to the users.
- It provides one of the best open source solutions for encryption.
- It is a very robust algorithm.

Disadvantages

- It requires many rounds for encryption.
- It is hard to implement on software.
- It needs much processing at different stages.
- It is difficult to implement when performance has to be considered.