

# Algorithm AES – Rijndael

José de Jesús Angel Angel \*

## 1 Introduction

The **Rijndael Algorithm** is a block cipher, that encrypt blocks of 128, 192, or 256 bits using symmetric keys of 128, 192 or 256 bits. It consists of an initial round (**AddRoundKey**), and  $r$  standard rounds,  $r$  is 10,12 or 14 that depending on the block and key length. The first  $r - 1$  rounds are similar and they consist of 4 transformations, called:

- **ByteSub** – Substitution Bytes
- **ShiftRow** – Shift Rows
- **MixColumn** – Multiply columns
- **AddRoundKey** – Xor-ed by key

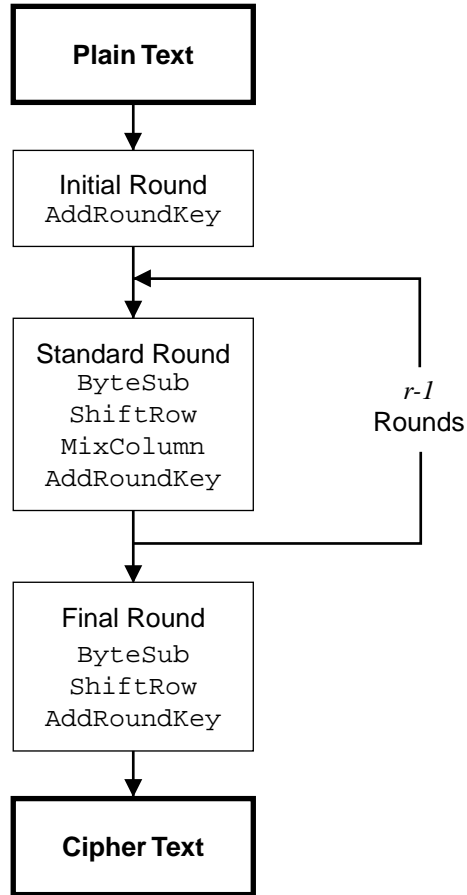
The last round only perform the transformations:

- **ByteSub**
- **ShiftRow**
- **AddRoundKey**

---

\*Please send any comments to [jesus@seguridata.com](mailto:jesus@seguridata.com)

The whole process is shown in the following diagram:



The transformations are applied to an array  $B$ , in matrix form  $B_{ji}$ , that is mapping in the following form: (This is the 128 bits case or 16 bytes)

$$B_{00} B_{10} B_{20} B_{30} B_{01} B_{11} B_{21} B_{31} B_{02} B_{12} B_{22} B_{32} B_{03} B_{13} B_{23} B_{33}$$

and the matrix is:

$$\begin{pmatrix} B_{00} & B_{10} & B_{20} & B_{30} \\ B_{01} & B_{11} & B_{21} & B_{31} \\ B_{02} & B_{12} & B_{22} & B_{32} \\ B_{03} & B_{13} & B_{23} & B_{33} \end{pmatrix}$$

In the 192 bits case, we have to add 2 columns more and in the 256 bits case, we have to add 4 columns more.

Notice that in  $B_{ji}$ ,  $j$  is the row and  $i$  the column.

All the transformations are applied to  $B$ , except `AddRoundKey` ( $B, K_r$ ), that uses the sub-key  $K_r[ji]$  as input.

One of the most important things in Rijndael is the key schedule: it generates the sub-keys  $K_r$  and load them into a linear array  $W$ . The length of the array  $W$  depends on the block and key length, and  $W$  provides the sub-keys  $K_r$  to every round  $r$ . Also the number of rounds  $r$  depends on the block and key length.

If at least the block or key length is 256 bits, then 14 rounds are needed. If both the block and key length are 128 bits, then 10 rounds are needed; otherwise 12 rounds are needed.

In this tutorial we will show step by step how Rijndael works, with a plain text  $B$  of 128 bits and a key  $K$  of 128 bits. The explanation is divided in two parts:

1. Key schedule to provide  $K_r$ .
2. Encryption (10 rounds in 128 bits case).

We are going to use the plain text  $B = 000102030405060708090A0B0C0D0E0F$  and the key  $K = 000102030405060708090A0B0C0D0E0F$ , both in hexadecimal.

This is the first vector test in the file `ecb_iv.txt`, taken from `rijndael-vals.zip`, available at <http://csrc.nist.gov/encryption/aes/rijndael/>

In every part we are going to explain in detail how it works, so the outputs will be very ease to check, in a few steps.

## 2 Key schedule

The key schedule is used to provide the sub-keys  $K_r$

Starting with the key  $K$ , we extend it and get the linear array  $W$ .  $W$  is an array with length  $N_b \times (N_r + 1)$ , where  $N_b$  is the number of columns of the matrix  $B_{ij}$  and  $N_r$  is the number of rounds.

In this case,  $N_b = 4$  and  $N_r = 10$ , so  $W$  have  $4 \times (10 + 1) = 44$  words of 32b. Every block of 4 words have 128 bits and it is the sub-key  $K_r$ . The first sub-key is  $K_0 = (W_0, W_1, W_2, W_3)$ , that is the copy of the key  $K$ .

The next sub-keys are obtained from the  $K_0$ . The rule is the following

$$\begin{aligned}
W_4 &= W_0 \text{ xor } temp_1 \\
W_5 &= W_1 \text{ xor } W_4 \\
W_6 &= W_2 \text{ xor } W_4 \\
W_7 &= W_3 \text{ xor } W_4 \\
W_8 &= W_4 \text{ xor } temp_2 \\
&\vdots \\
W_{43} &= W_{39} \text{ xor } W_{42} \\
W_{44} &= W_{40} \text{ xor } temp_{11}
\end{aligned}$$

Look at the cases of  $W_i$  where  $(i \bmod N_k \neq 0)$  and the rule is

$$W_i = W_{i-N_k} \text{ xor } W_{i-1}$$

with  $N_r = 4$ .

Where  $(i \bmod N_k = 0)$ , we have

$$W_i = W_{i-N_k} \text{ xor } temp_k$$

where  $temp_k = \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$ .

$\text{ByteSub}$  is applied to  $W_{i-1}$  with a byte shifted, and  $rcon_k$  is defined as  $rcon_k = (RC_k, 00, 00, 00)$ , with  $RC_1 = 1$ ,  $RC_k = X \times RC_{k-1} = X^{k-1}$  and  $RC_k \in GF(2^8)$  for  $k = i/4$  and  $i = 4, 8, 12, \dots, 44$ .

Consider the initial sub-key

$$K_0 = 00 \ 01 \ 02 \ 03 \ 04 \ 05 \ 06 \ 07 \ 08 \ 09 \ 0A \ 0B \ 0C \ 0D \ 0E \ 0F$$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_4 = W_0 \text{ xor } \text{ByteSub}(S_1 W_3)$$

$$\begin{aligned}
D7 &= 00 \text{ xor } D7 \\
AA &= 01 \text{ xor } AB \\
74 &= 02 \text{ xor } 76 \\
FD &= 03 \text{ xor } FE
\end{aligned}$$

$$\begin{aligned}
W_4[0] &= W_4[0] \text{ xor } rcon_1 \\
D6 &= D7 \text{ xor } 02
\end{aligned}$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$\begin{aligned}
W_i &= W_{i-4} \text{ xor } W_{i-1} \\
W_5 &= W_1 \text{ xor } W_4 & W_6 &= W_2 \text{ xor } W_5 & W_7 &= W_3 \text{ xor } W_6 \\
D2 &= 04 \text{ xor } D6 & DA &= 08 \text{ xor } D2 & D6 &= 0C \text{ xor } DA \\
AF &= 05 \text{ xor } AA & A6 &= 09 \text{ xor } AF & AB &= 0D \text{ xor } A6 \\
72 &= 06 \text{ xor } 74 & 78 &= 0A \text{ xor } 72 & 76 &= 0E \text{ xor } 78 \\
FA &= 07 \text{ xor } FD & F1 &= 0B \text{ xor } FA & FE &= 0F \text{ xor } F1
\end{aligned}$$

Sub-key  $K_1 = D6 \ AA \ 74 \ FD \ D2 \ AF \ 72 \ FA \ DA \ A6 \ 78 \ F1 \ D6 \ AB \ 76 \ FE$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$\begin{aligned}
W_8 &= W_4 \text{ xor } \text{ByteSub}(S_1 W_7) \\
B4 &= D6 \text{ xor } 62 \\
92 &= AA \text{ xor } 38 \\
CF &= 74 \text{ xor } BB \\
0B &= FD \text{ xor } F6 \\
W_8[0] &= W_8[0] \text{ xor } rcon_2 \\
B6 &= B4 \text{ xor } 04
\end{aligned}$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$\begin{aligned}
W_i &= W_{i-4} \text{ xor } W_{i-1} \\
W_9 &= W_5 \text{ xor } W_8 & W_{10} &= W_6 \text{ xor } W_9 & W_{11} &= W_7 \text{ xor } W_{10} \\
64 &= D2 \text{ xor } B6 & BE &= DA \text{ xor } 64 & 68 &= D6 \text{ xor } BE \\
3D &= AF \text{ xor } 92 & 9B &= A6 \text{ xor } 3D & 30 &= AB \text{ xor } 9B \\
BD &= 72 \text{ xor } CF & C5 &= 78 \text{ xor } BD & B3 &= 76 \text{ xor } C5 \\
F1 &= FA \text{ xor } 0B & 00 &= F1 \text{ xor } F1 & FE &= FE \text{ xor } 00
\end{aligned}$$

Sub-key  $K_2 = B6\ 92\ CF\ 0B\ 64\ 3D\ BD\ F1\ BE\ 9B\ C5\ 00\ 68\ 30\ B3\ FE$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_{12} = W_8 \text{ xor } \text{ByteSub}(S_1 W_{11})$$

$$B2 = B6 \text{ xor } 04$$

$$FF = 92 \text{ xor } 6D$$

$$74 = CF \text{ xor } BB$$

$$4E = 0B \text{ xor } 45$$

$$W_{12}[0] = W_{12}[0] \text{ xor } rcon_3$$

$$B6 = B2 \text{ xor } 08$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } W_{i-1}$$

$$W_{13} = W_9 \text{ xor } W_{12} \quad W_{14} = W_{10} \text{ xor } W_{13} \quad W_{15} = W_{11} \text{ xor } W_{14}$$

$$D2 = 64 \text{ xor } B6 \quad 6C = BE \text{ xor } D2 \quad 04 = 68 \text{ xor } 6C$$

$$C2 = 3D \text{ xor } FF \quad 59 = 9B \text{ xor } C2 \quad 69 = 30 \text{ xor } 59$$

$$C9 = BD \text{ xor } 74 \quad 0C = C5 \text{ xor } C9 \quad BF = B3 \text{ xor } 0C$$

$$BF = F1 \text{ xor } 4E \quad BF = 00 \text{ xor } BF \quad 41 = FE \text{ xor } BF$$

Sub-key  $K_3 = B6\ FF\ 74\ 4E\ D2\ C2\ C9\ BF\ 6C\ 59\ 0C\ BF\ 04\ 69\ BF\ 41$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_{16} = W_{12} \text{ xor } \text{ByteSub}(S_1 W_{15})$$

$$4F = B6 \text{ xor } F9$$

$$F7 = FF \text{ xor } 08$$

$$F7 = 74 \text{ xor } 83$$

$$BC = 4E \text{ xor } F2$$

$$\begin{aligned} W_{16}[0] &= W_{16}[0] \text{ xor } rcon_4 \\ 47 &= 4F \text{ xor } 10 \end{aligned}$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } W_{i-1}$$

$$\begin{aligned} W_{17} &= W_{13} \text{ xor } W_{16} & W_{18} &= W_{14} \text{ xor } W_{17} & W_{19} &= W_{15} \text{ xor } W_{18} \\ 95 &= D2 \text{ xor } 47 & F9 &= 6C \text{ xor } 95 & FD &= 04 \text{ xor } F9 \\ 35 &= C2 \text{ xor } F7 & 6C &= 59 \text{ xor } 35 & 05 &= 69 \text{ xor } 6C \\ 3E &= C9 \text{ xor } F7 & 32 &= 0C \text{ xor } 3E & 8D &= BF \text{ xor } 32 \\ 03 &= BF \text{ xor } BC & BC &= BF \text{ xor } 03 & FD &= 41 \text{ xor } BC \end{aligned}$$

Sub-key  $K_4 = 47 F7 F7 BC 95 35 3E 03 F9 6C 32 BC FD 05 8D FD$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_{20} = W_{16} \text{ xor } \text{ByteSub}(S_1 W_{19})$$

$$\begin{aligned} 2C &= 47 \text{ xor } 6B \\ AA &= F7 \text{ xor } 5D \\ A3 &= F7 \text{ xor } 54 \\ E8 &= BC \text{ xor } 54 \end{aligned}$$

$$\begin{aligned} W_{20}[0] &= W_{20}[0] \text{ xor } rcon_5 \\ 3C &= 2C \text{ xor } 20 \end{aligned}$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } W_{i-1}$$

$$\begin{aligned} W_{21} &= W_{17} \text{ xor } W_{20} & W_{22} &= W_{18} \text{ xor } W_{21} & W_{23} &= W_{19} \text{ xor } W_{22} \\ A9 &= 95 \text{ xor } 3C & 50 &= F9 \text{ xor } A9 & AD &= FD \text{ xor } 50 \\ 9F &= 35 \text{ xor } AA & F3 &= 6C \text{ xor } 9F & F6 &= 05 \text{ xor } F3 \\ 9D &= 3E \text{ xor } A3 & AF &= 32 \text{ xor } 9D & 22 &= 8D \text{ xor } AF \\ EB &= 03 \text{ xor } E8 & 57 &= BC \text{ xor } EB & AA &= FD \text{ xor } 57 \end{aligned}$$

Sub-key  $K_5 = 3C\ AA\ A3\ E8\ A9\ 9F\ 9D\ EB\ 50\ F3\ AF\ 57\ AD\ F6\ 22\ AA$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_{24} = W_{20} \text{ xor } \text{ByteSub}(S_1 W_{23})$$

$$7E = 3C \text{ xor } 42$$

$$39 = AA \text{ xor } 93$$

$$0F = A3 \text{ xor } AC$$

$$7D = E8 \text{ xor } 95$$

$$W_{24}[0] = W_{24}[0] \text{ xor } rcon_6$$

$$5E = 7E \text{ xor } 40$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } W_{i-1}$$

$$W_{25} = W_{21} \text{ xor } W_{24} \quad W_{26} = W_{22} \text{ xor } W_{25} \quad W_{27} = W_{23} \text{ xor } W_{26}$$

$$F7 = A9 \text{ xor } 5E \quad A7 = 50 \text{ xor } F7 \quad 0A = AD \text{ xor } A7$$

$$A6 = 9F \text{ xor } 39 \quad 55 = F3 \text{ xor } A6 \quad A3 = F6 \text{ xor } 55$$

$$92 = 9D \text{ xor } 0F \quad 3D = AF \text{ xor } 92 \quad 1F = 22 \text{ xor } 3D$$

$$96 = EB \text{ xor } 7D \quad C1 = 57 \text{ xor } 96 \quad 6B = AA \text{ xor } C1$$

Sub-key  $K_6 = 5E\ 39\ 0F\ 7D\ F7\ A6\ 92\ 96\ A7\ 55\ 3D\ C1\ 0A\ A3\ 1F\ 6B$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_{28} = W_{24} \text{ xor } \text{ByteSub}(S_1 W_{27})$$

$$54 = 5E \text{ xor } 0A$$

$$F9 = 39 \text{ xor } C0$$

$$70 = 0F \text{ xor } 7F$$

$$1A = 7D \text{ xor } 67$$



$$\begin{aligned} W_{28}[0] &= W_{28}[0] \text{ xor } rcon_7 \\ 14 &= 54 \text{ xor } 80 \end{aligned}$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } W_{i-1}$$

$$W_{29} = W_{25} \text{ xor } W_{28} \quad W_{30} = W_{26} \text{ xor } W_{29} \quad W_{31} = W_{27} \text{ xor } W_{30}$$

$$\begin{aligned} E3 &= F7 \text{ xor } 14 & 44 &= A7 \text{ xor } E3 & 4E &= 0A \text{ xor } 44 \\ 5F &= A6 \text{ xor } F9 & 0A &= 55 \text{ xor } 5F & A9 &= A3 \text{ xor } 0A \\ E2 &= 92 \text{ xor } 70 & DF &= 3D \text{ xor } E2 & C0 &= 1F \text{ xor } DF \\ 8C &= 96 \text{ xor } 1A & 4D &= C1 \text{ xor } 8C & 26 &= 6B \text{ xor } 4D \end{aligned}$$

Sub-key  $K_7 = 14 F9 70 1A E3 5F E2 8C 44 0A DF 4D 4E A9 C0 26$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_{32} = W_{28} \text{ xor } \text{ByteSub}(S_1 W_{31})$$

$$\begin{aligned} C7 &= 14 \text{ xor } D3 \\ 43 &= F9 \text{ xor } BA \\ 87 &= 70 \text{ xor } F7 \\ 35 &= 1A \text{ xor } 2F \end{aligned}$$

$$\begin{aligned} W_{32}[0] &= W_{32}[0] \text{ xor } rcon_8 \\ 47 &= C7 \text{ xor } 1B \end{aligned}$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } W_{i-1}$$

$$W_{33} = W_{29} \text{ xor } W_{32} \quad W_{34} = W_{30} \text{ xor } W_{33} \quad W_{35} = W_{31} \text{ xor } W_{34}$$

$$\begin{aligned} A4 &= E3 \text{ xor } 47 & E0 &= 44 \text{ xor } A4 & AE &= 4E \text{ xor } E0 \\ 1C &= 5F \text{ xor } 43 & 16 &= 0A \text{ xor } 1C & BF &= A9 \text{ xor } 16 \\ 65 &= E2 \text{ xor } 87 & BA &= DF \text{ xor } 65 & 7A &= C0 \text{ xor } BA \\ B9 &= 8C \text{ xor } 35 & F4 &= 4D \text{ xor } B9 & D2 &= 26 \text{ xor } F4 \end{aligned}$$

Sub-key  $K_8 = 47\ 43\ 87\ 35\ A4\ 1C\ 65\ B9\ E0\ 16\ BA\ F4\ AE\ BF\ 7A\ D2$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_{36} = W_{32} \text{ xor } \text{ByteSub}(S_1 W_{34})$$

$$4F = 47 \text{ xor } 08$$

$$99 = 43 \text{ xor } DA$$

$$32 = 87 \text{ xor } B5$$

$$D1 = 35 \text{ xor } E4$$

$$W_{36}[0] = W_{36}[0] \text{ xor } rcon_9$$

$$54 = 4F \text{ xor } 36$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } W_{i-1}$$

$$W_{37} = W_{33} \text{ xor } W_{36} \quad W_{38} = W_{34} \text{ xor } W_{37} \quad W_{39} = W_{35} \text{ xor } W_{38}$$

$$F0 = A4 \text{ xor } 54 \quad 10 = E0 \text{ xor } F0 \quad BE = AE \text{ xor } 10$$

$$85 = 1C \text{ xor } 99 \quad 93 = 16 \text{ xor } 85 \quad 2C = BF \text{ xor } 93$$

$$57 = 65 \text{ xor } 32 \quad ED = BA \text{ xor } 57 \quad 97 = 7A \text{ xor } ED$$

$$68 = B9 \text{ xor } D1 \quad 9C = F4 \text{ xor } 68 \quad 4E = D2 \text{ xor } 9C$$

Sub-key  $K_9 = 54\ 99\ 32\ D1\ F0\ 85\ 57\ 68\ 10\ 93\ ED\ 9C\ BE\ 2C\ 97\ 4E$

**Evaluation of  $W_i$  when  $i$  is multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } \text{ByteSub}(S_1 W_{i-1}) \text{ xor } rcon_k$$

and the calculated values are:

$$W_{40} = W_{36} \text{ xor } \text{ByteSub}(S_1 W_{39})$$

$$25 = 54 \text{ xor } 71$$

$$11 = 99 \text{ xor } 88$$

$$1D = 32 \text{ xor } 2F$$

$$7F = D1 \text{ xor } AE$$

$$\begin{aligned} W_{40}[0] &= W_{40}[0] \text{ xor } rcon_{10} \\ 13 &= 25 \text{ xor } 6C \end{aligned}$$

**Evaluation of  $W_i$  when  $i$  is not a multiple of 4.** It is obtained using the rule

$$W_i = W_{i-4} \text{ xor } W_{i-1}$$

$$W_{41} = W_{37} \text{ xor } W_{40} \quad W_{42} = W_{38} \text{ xor } W_{41} \quad W_{43} = W_{39} \text{ xor } W_{42}$$

$$\begin{array}{lll} E3 & = & F0 \text{ xor } 13 \quad F3 & = & 10 \text{ xor } E3 \quad 4D & = & BE \text{ xor } F3 \\ 94 & = & 85 \text{ xor } 11 \quad 07 & = & 93 \text{ xor } 94 \quad 2B & = & 2C \text{ xor } 07 \\ 4A & = & 57 \text{ xor } 1D \quad A7 & = & ED \text{ xor } 4A \quad 30 & = & 97 \text{ xor } A7 \\ 17 & = & 68 \text{ xor } 7F \quad 8B & = & 9C \text{ xor } 17 \quad C5 & = & 4E \text{ xor } 8B \end{array}$$

Sub-key  $K_10 = 13111D7FE3944A17F307A78B4D2B30C5$

### 3 Encryption

**Initial Round.** Key Addition.

$$\begin{array}{ccc} B_{ji} & & K_0[j, i] & & B_{ji} \\ \left( \begin{array}{cccc} 00 & 04 & 08 & 0C \\ 01 & 05 & 09 & 0D \\ 02 & 06 & 0A & 0E \\ 03 & 07 & 0B & 0F \end{array} \right) & \text{ xor } & \left( \begin{array}{cccc} 00 & 04 & 08 & 0C \\ 01 & 05 & 09 & 0D \\ 02 & 06 & 0A & 0E \\ 03 & 07 & 0B & 0F \end{array} \right) & = & \left( \begin{array}{cccc} 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \end{array} \right) \end{array}$$

This transformation applies a xor in every entry of the matrix  $B_{ji}$  and  $K_0[ji]$ , i.e.  $B_{ji} = B_{ij} \text{ xor } K_0[j, i]$ .

**Round 1.** Byte Substitution.

$$\left( \begin{array}{cccc} 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \end{array} \right) a_{ij}^{-1} = \left( \begin{array}{cccc} 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 \end{array} \right)$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the

irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.

### Linear Transformation $LT$

Here is the calculation for the entry  $(3, 3)$ ,  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation `ByteSub` .

### Round 1. ShiftRow.

$$\begin{pmatrix} 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \end{pmatrix} \text{ShiftRow}(a_j) = \begin{pmatrix} 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \end{pmatrix}$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of 192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

### Round 1. Mix-Column

$$\begin{pmatrix} 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \end{pmatrix} \text{MixColumn}(a_j) = \begin{pmatrix} 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \end{pmatrix}$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so **MixColumn** multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 1. Key Addition.**

$$\begin{pmatrix} 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \end{pmatrix} \text{ xor } \begin{pmatrix} D6 & D2 & DA & D6 \\ AA & AF & A6 & AB \\ 74 & 72 & 78 & 76 \\ FD & FA & F1 & FE \end{pmatrix} = \begin{pmatrix} B5 & B1 & B9 & B5 \\ C9 & CC & C5 & C8 \\ 17 & 11 & 1B & 15 \\ 9E & 99 & 92 & 9D \end{pmatrix}$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{ xor } K_r[j, i]$ . This is the final of the round  $r$ , and the partial vector test  $CT_r$  is

$$CT_1 = B5 C9 17 9E B1 CC 11 99 B9 C5 1B 92 B5 C8 15 9D$$

**Round 2. Byte Substitution.**

$$\begin{pmatrix} B5 & B1 & B9 & B5 \\ C9 & CC & C5 & C8 \\ 17 & 11 & 1B & 15 \\ 9E & 99 & 92 & 9D \end{pmatrix} a_{ij}^{-1} = \begin{pmatrix} 75 & E0 & 8E & 75 \\ 27 & 1B & D4 & A9 \\ 5F & B4 & CC & 2B \\ 89 & 14 & 32 & DC \end{pmatrix}$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.

**Round 2. Linear Transformation  $LT$ .**

Here is the calculation for the entry (3, 3),  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} D5 & C8 & 56 & D5 \\ DD & 4B & A6 & E8 \\ F0 & 82 & AF & 59 \\ 0B & EE & 4F & 5E \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation `ByteSub` .

**Round 2. ShiftRow.**

$$\left( \begin{array}{cccc} D5 & C8 & 56 & D5 \\ DD & 4B & A6 & E8 \\ F0 & 82 & AF & 59 \\ 0B & EE & 4F & 5E \end{array} \right) \text{ShiftRow}(a_j) = \left( \begin{array}{cccc} D5 & C8 & 56 & D5 \\ 4B & A6 & E8 & DD \\ AF & 59 & F0 & 82 \\ 5E & 0B & EE & 4F \end{array} \right)$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of 192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

**Round 2. Mix-Column**

$$\left( \begin{array}{c|c|c|c} D5 & C8 & 56 & D5 \\ 4B & A6 & E8 & DD \\ AF & 59 & F0 & 82 \\ 5E & 0B & EE & 4F \end{array} \right) \text{MixColumn}(a_j) = \left( \begin{array}{c|c|c|c} 9D & 28 & 91 & 00 \\ F7 & 7F & 78 & A6 \\ 39 & C1 & 6C & C6 \\ 3C & AA & 25 & A5 \end{array} \right)$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so `MixColumn` multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 2. Key Addition.**

$$\left( \begin{array}{cccc} 9D & 28 & 91 & 00 \\ F7 & 7F & 78 & A6 \\ 39 & C1 & 6C & C6 \\ 3C & AA & 25 & A5 \end{array} \right) \text{xor} \left( \begin{array}{cccc} B6 & 64 & BE & 68 \\ 92 & 3D & 9B & 30 \\ CF & BD & C5 & B3 \\ 0B & F1 & 00 & FE \end{array} \right) = \left( \begin{array}{cccc} 2B & 4C & 2F & 68 \\ 65 & 42 & E3 & 96 \\ F6 & 7C & A9 & 75 \\ 37 & 5B & 25 & 5B \end{array} \right)$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{ xor } K_r[j, i]$ . This is the final of the round  $r$ , and the parcial vector test  $CT_r$  is

$$CT_2 = 2B\ 65\ F6\ 37\ 4C\ 42\ 7C\ 5B\ 2F\ E3\ A9\ 25\ 68\ 96\ 75\ 5B$$

**Round 3.** Byte Substitution.

$$\begin{pmatrix} 2B & 4C & 2F & 68 \\ 65 & 42 & E3 & 96 \\ F6 & 7C & A9 & 75 \\ 37 & 5B & 25 & 5B \end{pmatrix} a_{ij}^{-1} = \begin{pmatrix} 15 & 54 & C2 & F4 \\ A6 & 37 & EB & 84 \\ 03 & A1 & C8 & B5 \\ 42 & F0 & 4D & F0 \end{pmatrix}$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.

**Round 3.** Linear Transformation  $LT$ .

Here is the calculation for the entry  $(3, 3)$ ,  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} F1 & 29 & 15 & 45 \\ 4D & 2C & 11 & 90 \\ 42 & 10 & D3 & 9D \\ 9A & 39 & 3F & 39 \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation ByteSub .

**Round 3.** ShiftRow.

$$\begin{pmatrix} F1 & 29 & 15 & 45 \\ 4D & 2C & 11 & 90 \\ 42 & 10 & D3 & 9D \\ 9A & 39 & 3F & 39 \end{pmatrix} \text{ShiftRow}(a_j) = \begin{pmatrix} F1 & 29 & 15 & 45 \\ 2C & 11 & 90 & 4D \\ D3 & 9D & 42 & 10 \\ 39 & 9A & 39 & 3F \end{pmatrix}$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of

192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

**Round 3. Mix-Column**

$$\left( \begin{array}{c|c|c|c} F1 & 29 & 15 & 45 \\ \hline 2C & 11 & 90 & 4D \\ \hline D3 & 9D & 42 & 10 \\ \hline 39 & 9A & 39 & 3F \end{array} \right) \text{MixColumn}(a_j) = \left( \begin{array}{c|c|c|c} 67 & 66 & FA & 72 \\ \hline FE & 2D & D1 & D0 \\ \hline 2B & AC & 4A & 69 \\ \hline 85 & D8 & 9F & EC \end{array} \right)$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so **MixColumn** multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 3. Key Addition.**

$$\left( \begin{array}{cccc} 67 & 66 & FA & 72 \\ FE & 2D & D1 & D0 \\ 2B & AC & 4A & 69 \\ 85 & D8 & 9F & EC \end{array} \right) \text{xor} \left( \begin{array}{cccc} B6 & D2 & 6C & 04 \\ FF & C2 & 59 & 69 \\ 74 & C9 & 0C & BF \\ 4E & BF & BF & 41 \end{array} \right) = \left( \begin{array}{cccc} D1 & B4 & 96 & 76 \\ 01 & EF & 88 & B9 \\ 5F & 65 & 46 & D6 \\ CB & 67 & 20 & AD \end{array} \right)$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{xor} K_r[j, i]$ . This is the final of the round  $r$ , and the parcial vector test  $CT_r$  is

$$CT_3 = D1\ 01\ 5F\ CB\ B4\ EF\ 65\ 67\ 96\ 88\ 46\ 20\ 76\ B9\ D6\ AD$$

**Round 4. Byte Substitution.**

$$\left( \begin{array}{cccc} D1 & B4 & 96 & 76 \\ 01 & EF & 88 & B9 \\ 5F & 65 & 46 & D6 \\ CB & 67 & 20 & AD \end{array} \right) a_{ij}^{-1} = \left( \begin{array}{cccc} 07 & 11 & 84 & BA \\ 01 & B3 & 9B & 8E \\ 17 & A6 & F5 & E2 \\ 04 & 43 & 3A & E7 \end{array} \right)$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.



**Round 4. Linear Transformation  $LT$ .**

Here is the calculation for the entry  $(3, 3)$ ,  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} 3E & 8D & 90 & 38 \\ 7C & DF & C4 & 56 \\ CF & 4D & 5A & F6 \\ 1F & 85 & B7 & 95 \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation **ByteSub** .

**Round 4. ShiftRow.**

$$\begin{pmatrix} 3E & 8D & 90 & 38 \\ 7C & DF & C4 & 56 \\ CF & 4D & 5A & F6 \\ 1F & 85 & B7 & 95 \end{pmatrix} \text{ShiftRow}(a_j) = \begin{pmatrix} 3E & 8D & 90 & 38 \\ \hline DF & C4 & 56 & 7C \\ \hline 5A & F6 & CF & 4D \\ \hline 95 & 1F & 85 & B7 \end{pmatrix}$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of 192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

**Round 4. Mix-Column**

$$\begin{pmatrix} 3E & 8D & 90 & 38 \\ DF & C4 & 56 & 7C \\ 5A & F6 & CF & 4D \\ 95 & 1F & 85 & B7 \end{pmatrix} \text{MixColumn}(a_j) = \begin{pmatrix} C9 & BF & 8B & 0E \\ E0 & 00 & F3 & A0 \\ F1 & 9F & D7 & 1C \\ F6 & 80 & 23 & 0C \end{pmatrix}$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so **MixColumn** multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 4.** Key Addition.

$$\begin{pmatrix} C9 & BF & 8B & 0E \\ E0 & 00 & F3 & A0 \\ F1 & 9F & D7 & 1C \\ F6 & 80 & 23 & 0C \end{pmatrix} \text{ xor } \begin{pmatrix} 47 & 95 & F9 & FD \\ F7 & 35 & 6C & 05 \\ F7 & 3E & 32 & 8D \\ BC & 03 & BC & FD \end{pmatrix} = \begin{pmatrix} 8E & 2A & 72 & F3 \\ 17 & 35 & 9F & A5 \\ 06 & A1 & E5 & 91 \\ 4A & 83 & 9F & F1 \end{pmatrix}$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{ xor } K_r[j, i]$ . This is the final of the round  $r$ , and the parcial vector test  $CT_r$  is

$$CT_4 = 8E 17 06 4A 2A 35 A1 83 72 9F E5 9F F3 A5 91 F1$$

**Round 5.** Byte Substitution.

$$\begin{pmatrix} 8E & 2A & 72 & F3 \\ 17 & 35 & 9F & A5 \\ 06 & A1 & E5 & 91 \\ 4A & 83 & 9F & F1 \end{pmatrix} a_{ij}^{-1} = \begin{pmatrix} B9 & 98 & 97 & 34 \\ 5F & 39 & 9A & B8 \\ 7B & 7C & 0E & 6A \\ AB & 80 & 9A & 23 \end{pmatrix}$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.

**Round 5.** Linear Transformation  $LT$ .

Here is the calculation for the entry  $(3, 3)$ ,  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} 19 & E5 & 40 & 0D \\ F0 & 96 & DB & 06 \\ 6F & 32 & D9 & 81 \\ D6 & EC & DB & A1 \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation `ByteSub` .

**Round 5. ShiftRow.**

$$\left( \begin{array}{cccc} 19 & E5 & 40 & 0D \\ \hline F0 & 96 & DB & 06 \\ \hline 6F & 32 & D9 & 81 \\ \hline D6 & EC & DB & A1 \end{array} \right) \text{ShiftRow}(a_j) = \left( \begin{array}{cccc} 19 & E5 & 40 & 0D \\ \hline 96 & DB & 06 & F0 \\ \hline D9 & 81 & 6F & 32 \\ \hline A1 & D6 & EC & DB \end{array} \right)$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of 192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

**Round 5. Mix-Column**

$$\left( \begin{array}{c|c|c|c} 19 & E5 & 40 & 0D \\ \hline 96 & DB & 06 & F0 \\ \hline D9 & 81 & 6F & 32 \\ \hline A1 & D6 & EC & DB \end{array} \right) \text{MixColumn}(a_j) = \left( \begin{array}{c|c|c|c} EB & F0 & 09 & F8 \\ \hline FF & 06 & 11 & 7B \\ \hline DE & 46 & B7 & EF \\ \hline 3D & D9 & 6A & 78 \end{array} \right)$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so `MixColumn` multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 5. Key Addition.**

$$\left( \begin{array}{cccc} EB & F0 & 09 & F8 \\ FF & 06 & 11 & 7B \\ DE & 46 & B7 & EF \\ 3D & D9 & 6A & 78 \end{array} \right) \text{xor} \left( \begin{array}{cccc} 3C & A9 & 50 & AD \\ AA & 9F & F3 & F6 \\ A3 & 9D & AF & 22 \\ E8 & EB & 57 & AA \end{array} \right) = \left( \begin{array}{cccc} D7 & 59 & 59 & 55 \\ 55 & 99 & E2 & 8D \\ 7D & DB & 18 & CD \\ D5 & 32 & 3D & D2 \end{array} \right)$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{ xor } K_r[j, i]$ . This is the final of the round  $r$ , and the partial vector test  $CT_r$  is

$$CT_5 = D7\ 55\ 7D\ D5\ 59\ 99\ DB\ 32\ 59\ E2\ 18\ 3D\ 55\ 8D\ CD\ D2$$

**Round 6.** Byte Substitution.

$$\begin{pmatrix} D7 & 59 & 59 & 55 \\ 55 & 99 & E2 & 8D \\ 7D & DB & 18 & CD \\ D5 & 32 & 3D & D2 \end{pmatrix} a_{ij}^{-1} = \begin{pmatrix} EA & 3E & 3E & 24 \\ 24 & 14 & D6 & 02 \\ FA & D5 & 58 & FC \\ DB & 92 & BB & AE \end{pmatrix}$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.

**Round 6.** Linear Transformation  $LT$ .

Here is the calculation for the entry  $(3, 3)$ ,  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} 0E & CB & CB & FC \\ FC & EE & 98 & 5D \\ FF & B9 & AD & BD \\ 03 & 23 & 27 & B5 \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation ByteSub .

**Round 6.** ShiftRow.

$$\begin{pmatrix} 0E & CB & CB & FC \\ FC & EE & 98 & 5D \\ FF & B9 & AD & BD \\ 03 & 23 & 27 & B5 \end{pmatrix} \text{ShiftRow}(a_j) = \begin{pmatrix} 0E & CB & CB & FC \\ EE & 98 & 5D & FC \\ AD & BD & FF & B9 \\ B5 & 03 & 23 & 27 \end{pmatrix}$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of

192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

**Round 6. Mix-Column**

$$\left( \begin{array}{c|c|c|c} 0E & CB & CB & FC \\ EE & 98 & 5D & FC \\ AD & BD & FF & B9 \\ B5 & 03 & 23 & 27 \end{array} \right) \text{MixColumn}(a_j) = \left( \begin{array}{c|c|c|c} 2D & 80 & B6 & 62 \\ 90 & 3F & 48 & E8 \\ 65 & 37 & 16 & 00 \\ 20 & 65 & A2 & 14 \end{array} \right)$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so **MixColumn** multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 6. Key Addition.**

$$\left( \begin{array}{c|c|c|c} 2D & 80 & B6 & 62 \\ 90 & 3F & 48 & E8 \\ 65 & 37 & 16 & 00 \\ 20 & 65 & A2 & 14 \end{array} \right) \text{xor} \left( \begin{array}{c|c|c|c} 5E & F7 & A7 & 0A \\ 39 & A6 & 55 & A3 \\ 0F & 92 & 3D & 1F \\ 7D & 96 & C1 & 6B \end{array} \right) = \left( \begin{array}{c|c|c|c} 73 & 77 & 11 & 68 \\ A9 & 99 & 1D & 4B \\ 6A & A5 & 2B & 1F \\ 5D & F3 & 63 & 7F \end{array} \right)$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{ xor } K_r[j, i]$ . This is the final of the round  $r$ , and the parcial vector test  $CT_r$  is

$$CT_6 = 73 \ A9 \ 6A \ 5D \ 77 \ 99 \ A5 \ F3 \ 11 \ 1D \ 2B \ 63 \ 68 \ 4B \ 1F \ 7F$$

**Round 7. Byte Substitution.**

$$\left( \begin{array}{c|c|c|c} 73 & 77 & 11 & 68 \\ A9 & 99 & 1D & 4B \\ 6A & A5 & 2B & 1F \\ 5D & F3 & 63 & 7F \end{array} \right) a_{ij}^{-1} = \left( \begin{array}{c|c|c|c} 85 & 3C & B4 & F4 \\ C8 & 14 & 40 & 13 \\ 91 & B8 & 15 & B2 \\ EC & 34 & D3 & 82 \end{array} \right)$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.

**Round 7. Linear Transformation  $LT$ .**

Here is the calculation for the entry  $(3, 3)$ ,  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} 8F & F5 & 82 & 45 \\ D3 & EE & A4 & B3 \\ 02 & 06 & F1 & C0 \\ 4C & 0D & FB & D2 \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation **ByteSub** .

**Round 7. ShiftRow.**

$$\begin{pmatrix} 8F & F5 & 82 & 45 \\ D3 & EE & A4 & B3 \\ 02 & 06 & F1 & C0 \\ 4C & 0D & FB & D2 \end{pmatrix} \text{ShiftRow}(a_j) = \begin{pmatrix} 8F & F5 & 82 & 45 \\ EE & A4 & B3 & D3 \\ F1 & C0 & 02 & 06 \\ D2 & 4C & 0D & FB \end{pmatrix}$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of 192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

**Round 7. Mix-Column**

$$\begin{pmatrix} 8F & F5 & 82 & 45 \\ EE & A4 & B3 & D3 \\ F1 & C0 & 02 & 06 \\ D2 & 4C & 0D & FB \end{pmatrix} \text{MixColumn}(a_j) = \begin{pmatrix} 0F & 8A & DE & 19 \\ 92 & B1 & F4 & 09 \\ F5 & 1E & 22 & 8C \\ 2A & F8 & 36 & F7 \end{pmatrix}$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so **MixColumn** multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 7.** Key Addition.

$$\begin{pmatrix} 0F & 8A & DE & 19 \\ 92 & B1 & F4 & 09 \\ F5 & 1E & 22 & 8C \\ 2A & F8 & 36 & F7 \end{pmatrix} \text{ xor } \begin{pmatrix} 14 & E3 & 44 & 4E \\ F9 & 5F & 0A & A9 \\ 70 & E2 & DF & C0 \\ 1A & 8C & 4D & 26 \end{pmatrix} = \begin{pmatrix} 1B & 69 & 9A & 57 \\ 6B & EE & FE & A0 \\ 85 & FC & FD & 4C \\ 30 & 74 & 7B & D1 \end{pmatrix}$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{ xor } K_r[j, i]$ . This is the final of the round  $r$ , and the parcial vector test  $CT_r$  is

$$CT_7 = 1B\ 6B\ 85\ 30\ 69\ EE\ FC\ 74\ 9A\ FE\ FD\ 7B\ 57\ A0\ 4C\ D1$$

**Round 8.** Byte Substitution.

$$\begin{pmatrix} 1B & 69 & 9A & 57 \\ 6B & EE & FE & A0 \\ 85 & FC & FD & 4C \\ 30 & 74 & 7B & D1 \end{pmatrix} a_{ij}^{-1} = \begin{pmatrix} CC & 47 & 9F & BF \\ DF & 1E & 41 & FB \\ 73 & CD & 1A & 54 \\ 2C & 10 & 06 & 07 \end{pmatrix}$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.

**Round 8.** Linear Transformation  $LT$ .

Here is the calculation for the entry  $(3, 3)$ ,  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} AF & F9 & B8 & 5B \\ 7F & 28 & BB & E0 \\ 97 & B0 & 54 & 29 \\ 04 & 92 & 21 & 3E \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation `ByteSub` .

**Round 8.** `ShiftRow`.

$$\left( \begin{array}{cccc} AF & F9 & B8 & 5B \\ 7F & 28 & BB & E0 \\ 97 & B0 & 54 & 29 \\ 04 & 92 & 21 & 3E \end{array} \right) \text{ShiftRow}(a_j) = \left( \begin{array}{cccc} AF & F9 & B8 & 5B \\ 28 & BB & E0 & 7F \\ 54 & 29 & 97 & B0 \\ 3E & 04 & 92 & 21 \end{array} \right)$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of 192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

**Round 8.** `Mix-Column`

$$\left( \begin{array}{c|c|c|c} AF & F9 & B8 & 5B \\ 28 & BB & E0 & 7F \\ 54 & 29 & 97 & B0 \\ 3E & 04 & 92 & 21 \end{array} \right) \text{MixColumn}(a_j) = \left( \begin{array}{c|c|c|c} 57 & 12 & 55 & A6 \\ 3D & EB & 53 & 4F \\ 6D & 1C & C0 & 3C \\ EA & 8A & 9B & 60 \end{array} \right)$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so `MixColumn` multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 8.** `Key Addition`.

$$\left( \begin{array}{cccc} 57 & 12 & 55 & A6 \\ 3D & EB & 53 & 4F \\ 6D & 1C & C0 & 3C \\ EA & 8A & 9B & 60 \end{array} \right) \text{xor} \left( \begin{array}{cccc} 47 & A4 & E0 & AE \\ 43 & 1C & 16 & BF \\ 87 & 65 & BA & 7A \\ 35 & B9 & F4 & D2 \end{array} \right) = \left( \begin{array}{cccc} 10 & B6 & B5 & 08 \\ 7E & F7 & 45 & F0 \\ EA & 79 & 7A & 46 \\ DF & 33 & 6F & B2 \end{array} \right)$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{ xor } K_r[j, i]$ . This is the final of the round  $r$ , and the parcial vector test  $CT_r$  is

$$CT_8 = 107E EA DF B6 F7 79 33 B5 45 7A 6F 08 F0 46 B2$$



**Round 9.** Byte Substitution.

$$\begin{pmatrix} 10 & B6 & B5 & 08 \\ 7E & F7 & 45 & F0 \\ EA & 79 & 7A & 46 \\ DF & 33 & 6F & B2 \end{pmatrix} a_{ij}^{-1} = \begin{pmatrix} 74 & 78 & 75 & E8 \\ 81 & 8C & 31 & 5B \\ D7 & 70 & D0 & F5 \\ 6B & 6C & 3B & 1F \end{pmatrix}$$

This transformation replace every entry of the input  $B_{ji}$  by the result of two operations: The first one maps an entry to their multiplicative inverse in  $GF(2^8)$ . Every entry (a byte) have a polynomial representation with the irreducible polynomial  $f(x) = x^8 + x^4 + x^3 + x + 1$ . The zero is mapped to zero.

**Round 8.** Linear Transformation  $LT$ .

Here is the calculation for the entry  $(3, 3)$ ,  $LT(a_{33}^{-1})$ :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad LT(a_{ji}^{-1}) = \begin{pmatrix} CA & 4E & D5 & 30 \\ F3 & 68 & 6E & 8C \\ 87 & B6 & DA & 5A \\ 9E & C3 & A8 & 37 \end{pmatrix}$$

The second operation apply a linear transformation from  $GF(2^8)$  to  $GF(2^8)$  in every entry (byte) of the matrix. The input are the bits  $x_7, \dots, x_0$  in  $a_{ji}^{-1}$  and the output are  $y_7, \dots, y_0$ , this is the new  $(j, i)$  element  $B_{ji}$ .

This is the final output in the transformation ByteSub .

**Round 9.** ShiftRow.

$$\begin{pmatrix} CA & 4E & D5 & 30 \\ F3 & 68 & 6E & 8C \\ 87 & B6 & DA & 5A \\ 9E & C3 & A8 & 37 \end{pmatrix} \text{ShiftRow}(a_j) = \begin{pmatrix} CA & 4E & D5 & 30 \\ 68 & 6E & 8C & F3 \\ DA & 5A & 87 & B6 \\ 37 & 9E & C3 & A8 \end{pmatrix}$$

This transformation is applied to the rows of  $B_{ji}$ , it shifts bytes of the rows.

The row 1 is not shifted, the row 2 is shifted 1 byte, the row 3 shifted 2 bytes and the row 4, 3 bytes (as is in our case of 128 bits). In the case of

192 bits is the same, and the case of 256 bits only changes the row 3,4, that are shifted 3 and 4 bytes respectively.

**Round 9. Mix-Column**

$$\left( \begin{array}{c|c|c|c} CA & 4E & D5 & 30 \\ \hline 68 & 6E & 8C & F3 \\ \hline DA & 5A & 87 & B6 \\ \hline 37 & 9E & C3 & A8 \end{array} \right) \text{MixColumn}(a_j) = \left( \begin{array}{c|c|c|c} DA & EA & 7A & 70 \\ \hline 58 & E2 & 87 & A4 \\ \hline 54 & 2D & 12 & 57 \\ \hline 99 & C1 & F2 & 5E \end{array} \right)$$

This transformation is applied to the columns of  $B_{ji}$ . We can see each column as a polynomial of  $GF(2^8)[x]$ , so **MixColumn** multiply each column by the constant polynomial  $c(x) = 03x^3 + 01x^2 + 01x + 02$ , modulus  $x^4 + 1$  their coefficients are in hexadecimal, and they are in  $GF(2^8)$ .

**Round 9. Key Addition.**

$$\left( \begin{array}{c|c|c|c} DA & EA & 7A & 70 \\ \hline 58 & E2 & 87 & A4 \\ \hline 54 & 2D & 12 & 57 \\ \hline 99 & C1 & F2 & 5E \end{array} \right) \text{xor} \left( \begin{array}{c|c|c|c} 54 & F0 & 10 & BE \\ \hline 99 & 85 & 93 & 2C \\ \hline 32 & 57 & ED & 97 \\ \hline D1 & 68 & 9C & 4E \end{array} \right) = \left( \begin{array}{c|c|c|c} 8E & 1A & 6A & CE \\ \hline C1 & 67 & 14 & 88 \\ \hline 66 & 7A & FF & C0 \\ \hline 48 & A9 & 6E & 10 \end{array} \right)$$

This transformation only Xor-ed every entry the matrix  $B_{ji}$  by every entry of  $K_r[j, i]$  from the Key schedule.  $B_{ji} = B_{ji} \text{ xor } K_r[j, i]$ . This is the final of the round  $r$ , and the partial vector test  $CT_r$  is

$$CT_9 = 8E C1 66 48 1A 67 7A A9 6A 14 FF 6E CE 88 C0 10$$

**Final Round. Byte Substitution.**

$$\left( \begin{array}{c|c|c|c} 8E & 1A & 6A & CE \\ \hline C1 & 67 & 14 & 88 \\ \hline 66 & 7A & FF & C0 \\ \hline 48 & A9 & 6E & 10 \end{array} \right) a_{ij}^{-1} = \left( \begin{array}{c|c|c|c} 19 & A2 & 02 & 8B \\ \hline 78 & 85 & FA & C4 \\ \hline 33 & DA & 16 & BA \\ \hline 52 & D3 & 9F & CA \end{array} \right)$$

**Final Round. ShiftRow.**

$$\left( \begin{array}{c|c|c|c} 19 & A2 & 02 & 8B \\ \hline 78 & 85 & FA & C4 \\ \hline 33 & DA & 16 & BA \\ \hline 52 & D3 & 9F & CA \end{array} \right) \text{ShiftRow}(a_j) = \left( \begin{array}{c|c|c|c} 19 & A2 & 02 & 8B \\ \hline 85 & FA & C4 & 78 \\ \hline 16 & BA & 33 & DA \\ \hline CA & 52 & D3 & 9F \end{array} \right)$$

**Final Round.** Key Addition.

$$\begin{pmatrix} 19 & A2 & 02 & 8B \\ 85 & FA & C4 & 78 \\ 16 & BA & 33 & DA \\ CA & 52 & D3 & 9F \end{pmatrix} \text{ xor } \begin{pmatrix} 13 & E3 & F3 & 4D \\ 11 & 94 & 07 & 2B \\ 1D & 4A & A7 & 30 \\ 7F & 17 & 8B & C5 \end{pmatrix} = \begin{pmatrix} 0A & 41 & F1 & C6 \\ 94 & 6E & C3 & 53 \\ 0B & F0 & 94 & EA \\ B5 & 45 & 58 & 5A \end{pmatrix}$$

*Cipher Text* = 0A 94 0B B5 41 6E F0 45 F1 C3 94 58 C6 53 EA 5A

In the decipher process, we apply the inverse transformations.

*Decipher Text* = 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F