Chapter 7

Advanced Encryption Standard (AES)

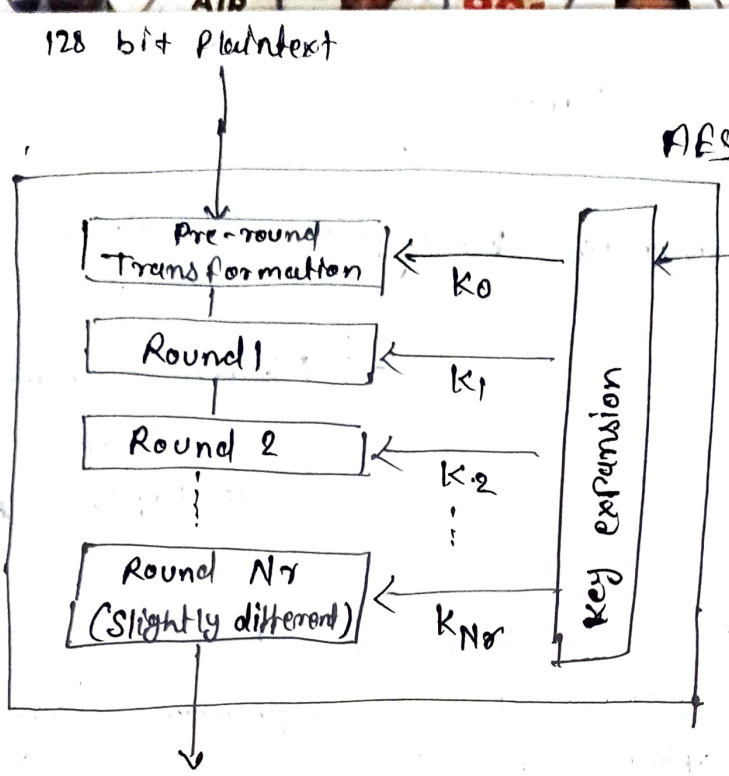
→ Modern Symmetric key cipher That Replace DES

Introduction :- The Advanced Encryption Standard (AES) is a symmetric key block cipher Published by NIST in Dec 2001.

History :- In 1997, NIST started looking for a replacement for DES, which would be called the Advanced Encryption Standard or AES.
The Block size is required for is 128 bits And 3 different key sizes of 128, 192, 256 bits

Rounds :-

1. AES is a non-feistel cipher that encrypts & decrypts a data block of 128 bits.
2. It uses 10, 12, or 14 rounds.
3. The key size, which can be 128, 192 or 256 bits. Depends on the number of rounds.
4. The Decryption algorithm is similar but the round keys are applied in the reverse order.



Nr	K Size
10	128
12	192
14	256

Relationship b/w Number of rounds & cipher key size

In this figure Nr defines the number of rounds.

Based on Number of rounds & key size we have 3 different version of AES.

they reffered to AES-128, AES-192, AES-256

However the key created by the key-expansion algorithm is always 128 bits Same size as plaintext & ciphertext block. (Means Round keys are of 128 bits)

The Number of round keys generated by the key-expansion algorithm is always one more than the number of rounds.

Number of round keys = Nr + 1

Data units :- AES uses five units of measurement to refer the data :- bits, bytes, words blocks and state.

The bit is the smallest and atomic unit. So that other units can be expressed in terms of ~~other unit~~ smaller unit.

Bit :- In AES, a bit is a binary digit with a value of 0 or 1, we use lowercase letter to refer to a bit.

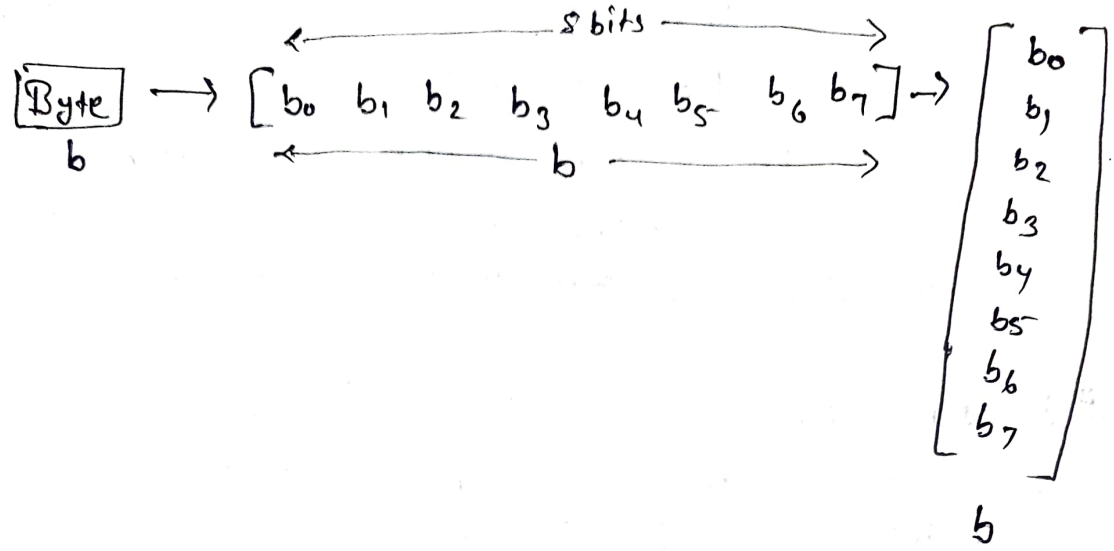
Byte :- It is a collection of 8 bits. we use lowercase bold letter to refer a byte.

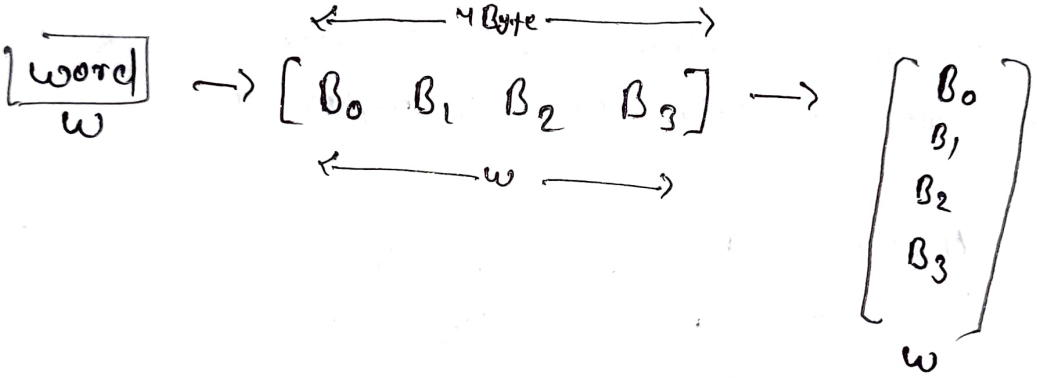
word :- A word is a group of 32 bits that can be treated as a single entity.
we use lowercase bold letter w to show a word.

Block :- AES encrypts and decrypts data blocks. A block in AES is a group of 128 bits

State :- AES uses several Rounds. in which each round is made up of several stages.
Data block is transformed from one stage to another. Before and after each stage the data block is referred to as a State.
we use uppercase bold letter to refer a State
State, like Block made up of 16 bytes.

Notations :-





Block → $[B_0\ B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7\ B_8\ B_9\ B_{10}\ B_{11}\ B_{12}\ B_{13}\ B_{14}\ B_{15}]$

State →
$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,3} & S_{3,3} \end{bmatrix} \rightarrow [w_0\ w_1\ w_2\ w_3]$$

Example :- How 16-character Block can be shown as 4×4 matrix. Assume that the text block is "AES uses a matrix".

Solve we add bogus character and ends with

AESUSESAMATRIXZZ

Now Each character is changed to its integer value then it is changed to its Hexadecimal value.

Like $S \rightarrow 18 \rightarrow 12_{16}$

Text [A E S U S E S A M A T R I X Z Z]

Hex value [00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19]

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$

Structure of each Round :-

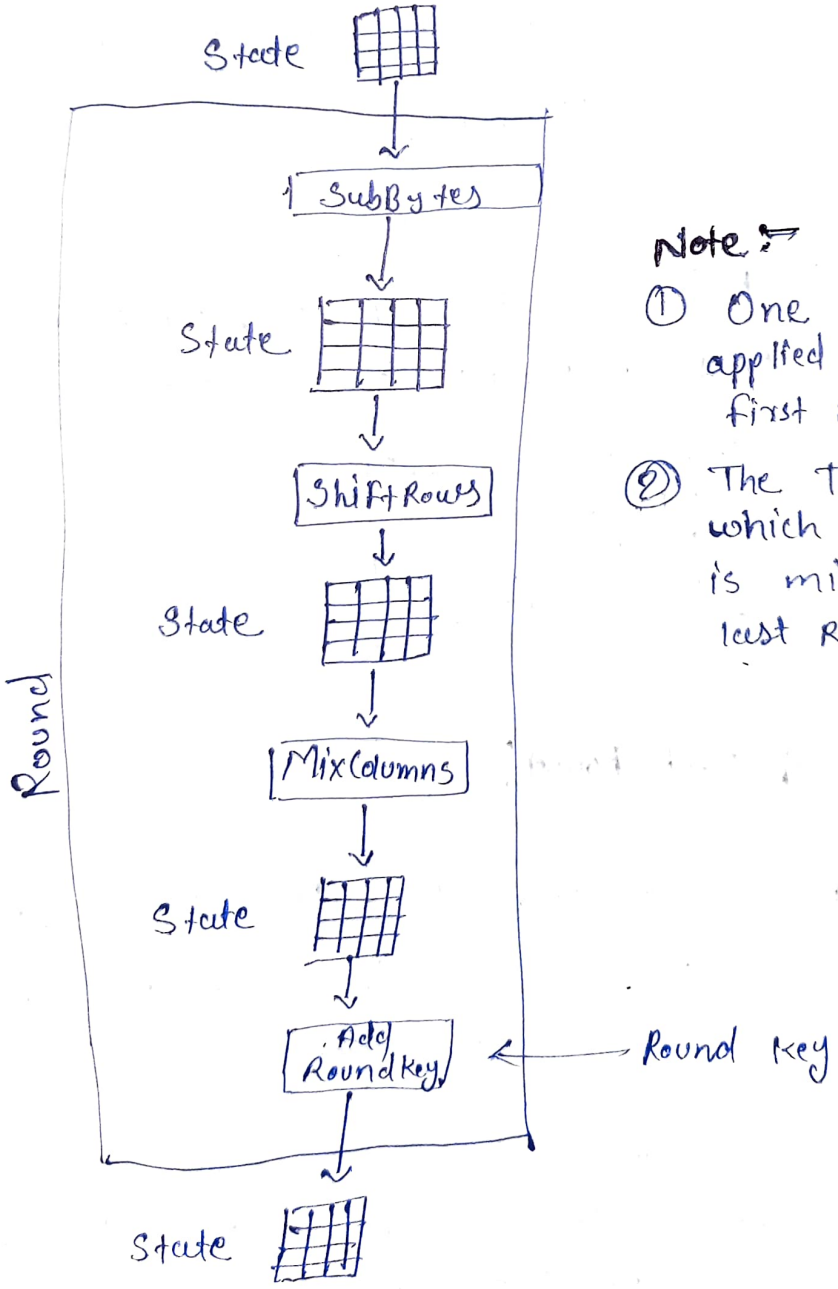
Each Round except the last Round, uses four transformations that are invertible. the last round has only 3 transformation.

Each transformation takes a state and creates another state to be used for the next transformation or the next round.

The Pre-round section uses only one transformation (Add Round key).

The last Round uses only three transformations (MixColumns is Missing in last Round)

→ Structure of each round at encryption side



Note :-

① One Add Round key is applied before the first Round.

② The third Transformation which is MixColumns is missing in the last Round.

At the decryption site, the inverse transformations are used:

InvSubByte, InvShiftRows, InvMixColumns.
& AddRoundKey (this one is self-invertible)

TRANSFORMATION :- To Provide Security, AES uses four types of transformations :-

Substitution, permutation, mixing, key-adding.

1. **Substitution** :- AES, like DES, uses substitution. However the machenism is Different.

① Substitution is done for each byte
② Only one table is used for transformation of every byte which means if two bytes are same then the transformation is also same.
③ Transformation is done By either a table lookup process or mathematical calculation in the $GF(2^8)$ field.

AES uses two invertible transformations.:-

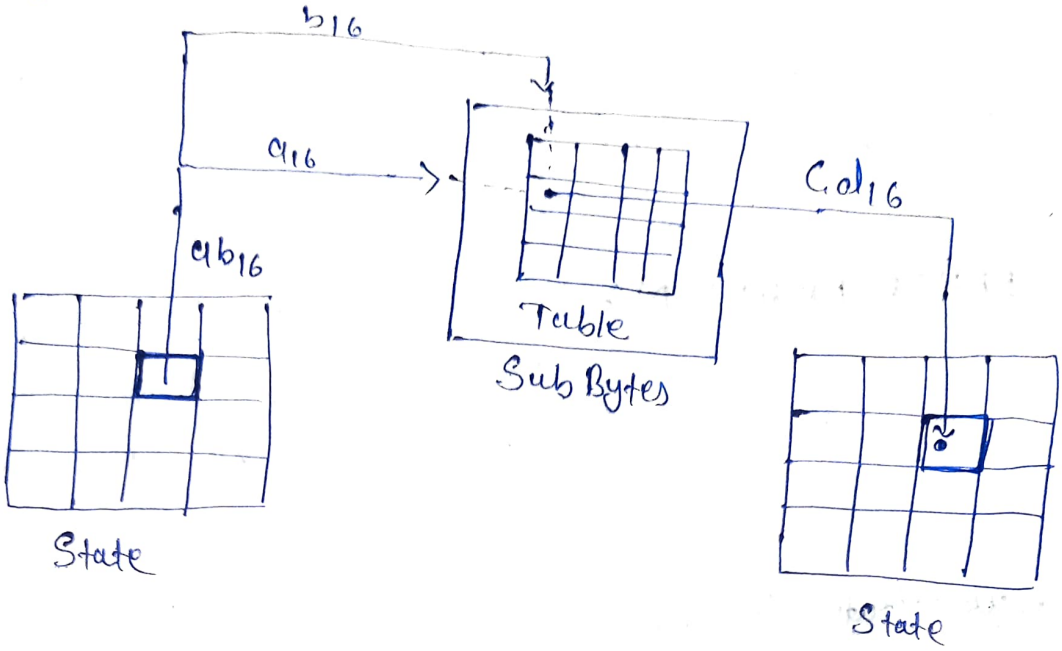
(i) **SubBytes** :- The first transformation, SubBytes, is used at encryption site.

To Substitute a byte, we interpret the byte as two hexadecimal digits.

The left digit define the row & the right digit defines the column of a Substitution table. The two hexadicimal digits at the Function of Row & column are the new byte.

Table lookup Method

Sub Bytes transformation



In this transformation, the State is treated as a 4X4 matrix of bytes.

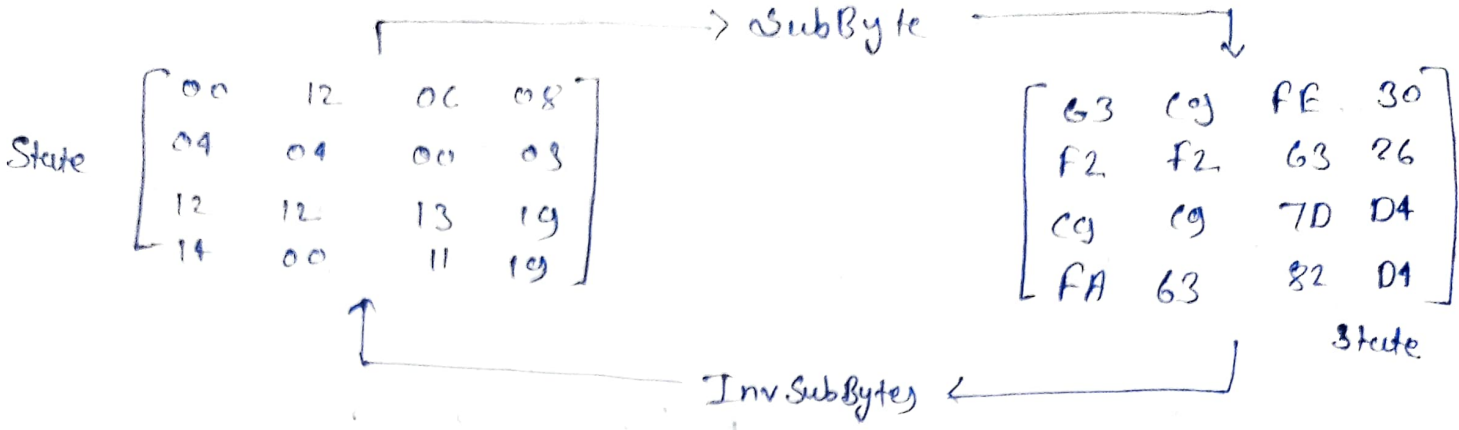
Transformation is done One byte at a time.

The content of each byte is changed, But the Arrangment of Bytes in the matrix remains the Same. In the Process each byte is transformed independently.

There are 16 byte-to-byte transformation.

The transformation definitely Provides confusion effect. for example, two bytes $5A_{16}$ and $5B_{16}$ which differ only one bit are transformed to BE_{16} & 89_{16} which differ in four bits

InvSubBytes :- InvSubBytes is a inverse of SubBytes.



NOTE :- If the two Bytes are same then their transformation is also same, Because Every Bytes uses the same table.
But it is Not possible in DES Because DES uses eight different S-boxes.

Transformation using the $GF(2^8)$ field :-

In the previous section we have done the transformation using table lookup.
AES also defines the transformation algebraically using the $GF(2^8)$ field with the irreducible Polynomial $(x^8 + x^4 + x^3 + x + 1)$.

In this there is also SubByte routine & InvSubByte Routine, which is Reapeated 16 times.

In SubByte Routine, the multiplicative inverse of the byte is found in $GF(2^8)$ with the irreducible polynomial $(x^8 + x^4 + x^3 + x + 1)$ as the modulus.

The inverted Byte is interpreted as a column matrix with the least significant bit at the top & the most significant bit at the bottom.

This column matrix is multiplied by constant square matrix X, ~~and the result, which~~

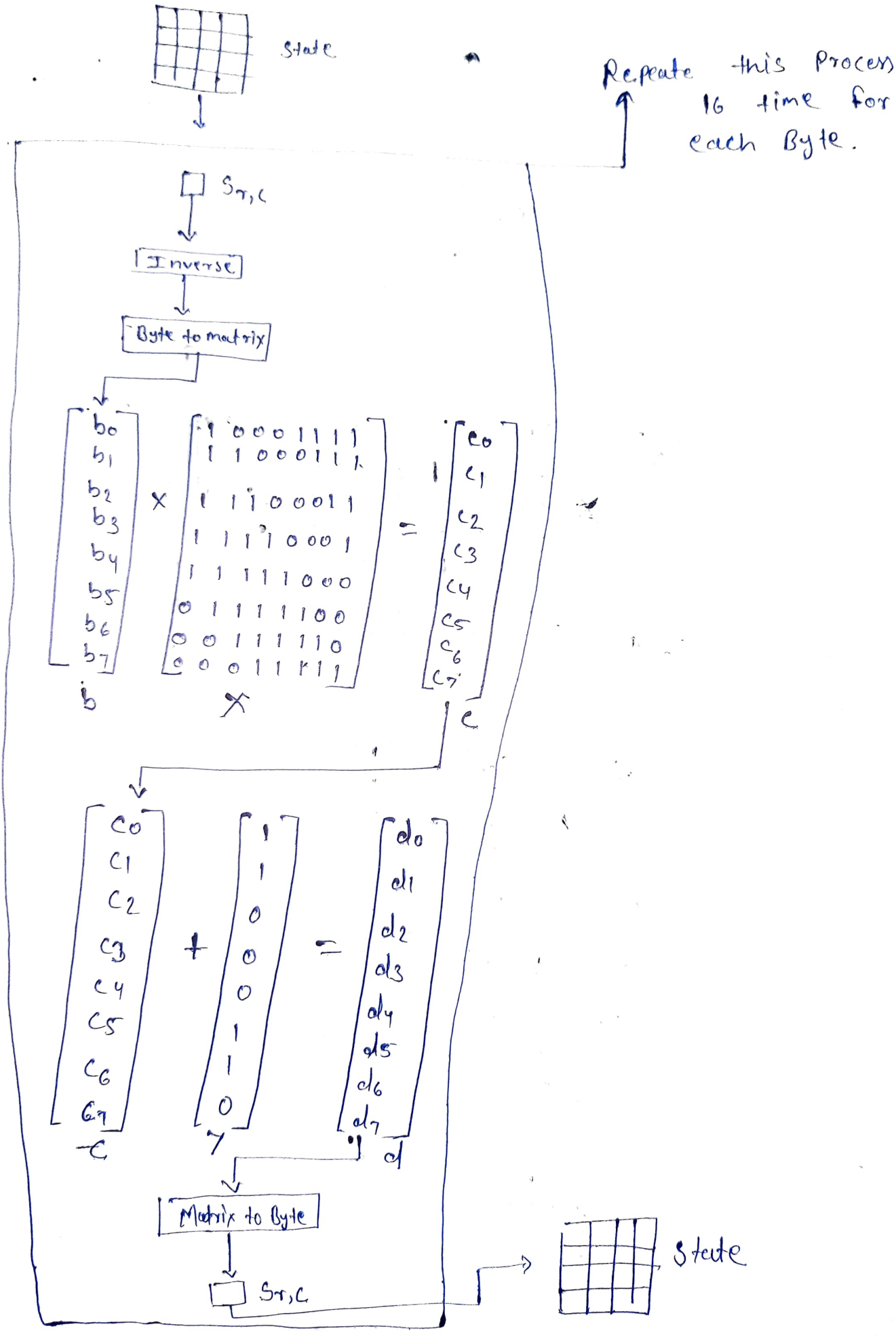
The result of this is a column matrix.

Then the resultant matrix is ~~Multiplied to~~ added with a constant column matrix Y.

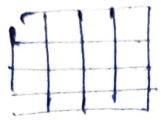
~~Ato~~ Multiplication & addition of Bit is done in $GF(2)$.

The ~~inverse~~ invSubByte is doing the same thing in reverse order.

After finding a Multiplicative inverse, it is like a Affine cipher.

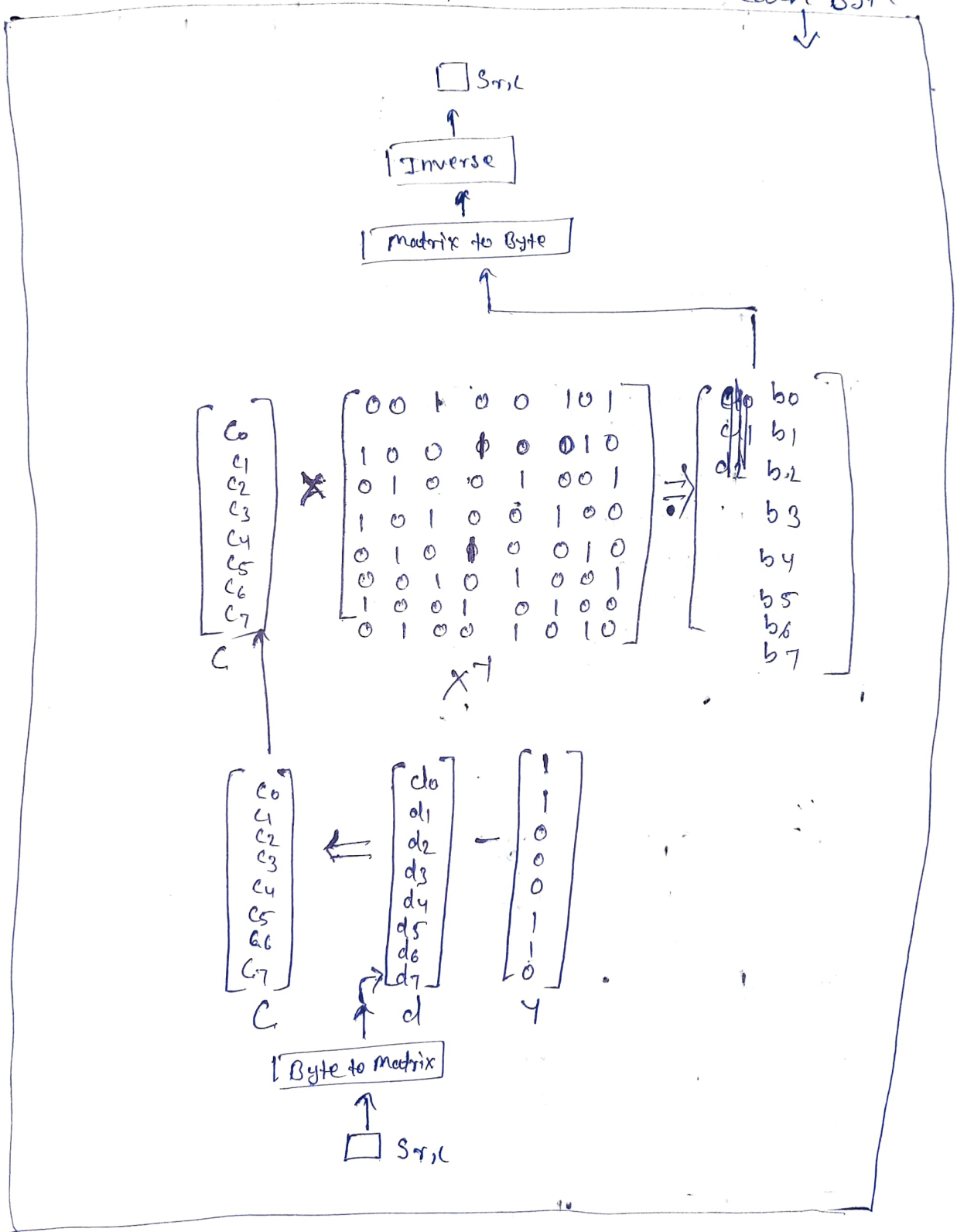

State
Repeate this Process 16 time for each Byte.
Sr,c
Inverse
Byte to matrix
Matrix to Byte
Sr,c
State

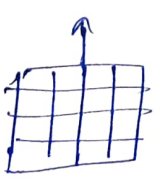
InvSubByte:-



State

Repeate this Process 16 time for each Byte





Ex:- transform 0C & also reverse of this :-

① SubByte :-

→ The Multiplicative inverse of 0C in GF(2^8) is B0, which means b = [10110000]

→ Multiply matrix X by this matrix result in C = (10011101)

→ The Result of XOR operation (Addition) which is FE in hexadecimal.

② InvSubByte :-

→ The Result of XOR operation (Substraction) which is C = (10011101)

→ Result of multiplying by matrix X^{-1} is (11010000) or. B0

→ The multiplicative inverse of B0 is 0C.

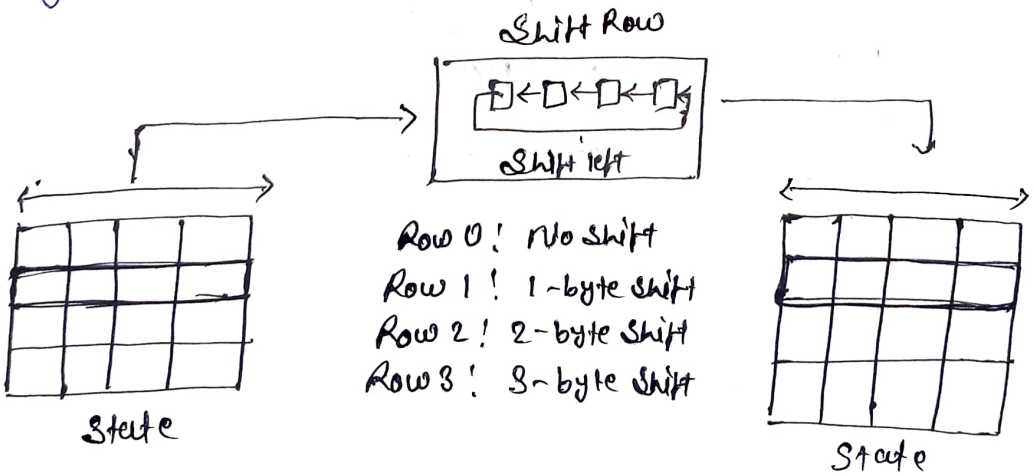
Permutation :- Another transformation found in a round is Shifting, which permutes the bytes. Unlike DES, in which permutation is done at the bit level, Shifting transformation in AES is done at the byte level.

The order of the bits in the byte is not changed.

Shift row :- In the encryption, the transformation is called ShiftRows and the Shifting is to the left.

The number of Shifts depends on the row number (0, 1, 2, 3) of the state matrix.

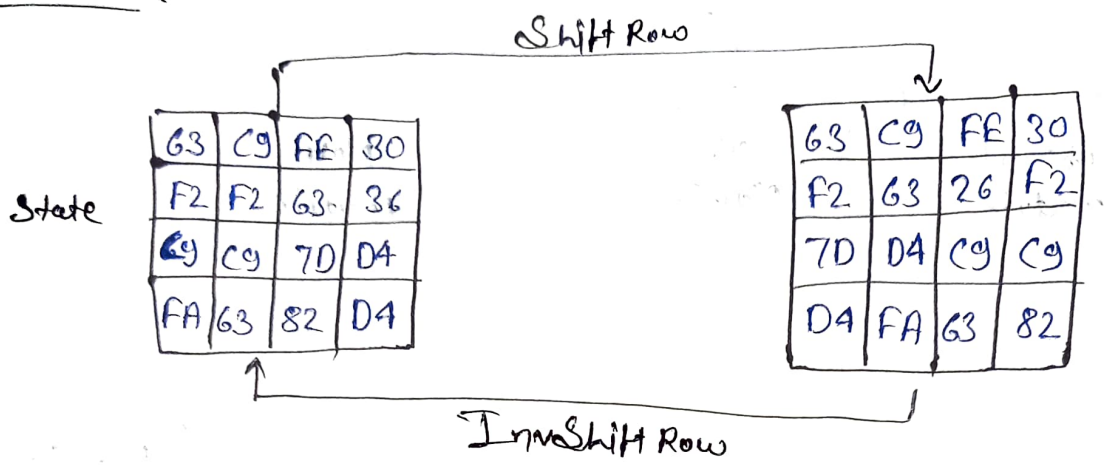
This means the row 0 is not Shifted at all and the last row is Shifted three bytes.



The ShiftRows transformation operates one row at a time.

InvShiftRows :- In the decryption, the transformation is called InvShiftRows and the shifting is to the right. The number of shifts is the same as the row number (0, 1, 2, & 3) of the State matrix.

Example :-



 # Mixing :- The mixing transformation changes the content of each byte by taking four bytes at a time and combining them to recreate four new bytes.

To guarantee that each new byte is different (even if all four bytes are the same) the combination process first multiplies each byte with a different constant & then mixes them.

The mixing is provided by matrix multiplication.

Each element in the new matrix depends on all four elements of the old matrix after they are multiplied by row values in the constant matrix

Mixing using matrix Multiplication:-

~~ax + by + cz + dt~~

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} = \begin{bmatrix} ax + by + cz + dt \\ ex + fy + gz + ht \\ ix + jy + kz + lt \\ mx + ny + oz + pt \end{bmatrix}$$

Constant matrix — Old Matrix — New matrix

AES defines a transformation, called "MixColumns" to achieve this goal.

there is also an inverse transformation, called InvMixColumns.

In this a constant matrix is used for transformation.

Constant matrix used by MixColumns and InvMixColumns are Inverse of each other

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \xleftrightarrow{\text{Inverse}} \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

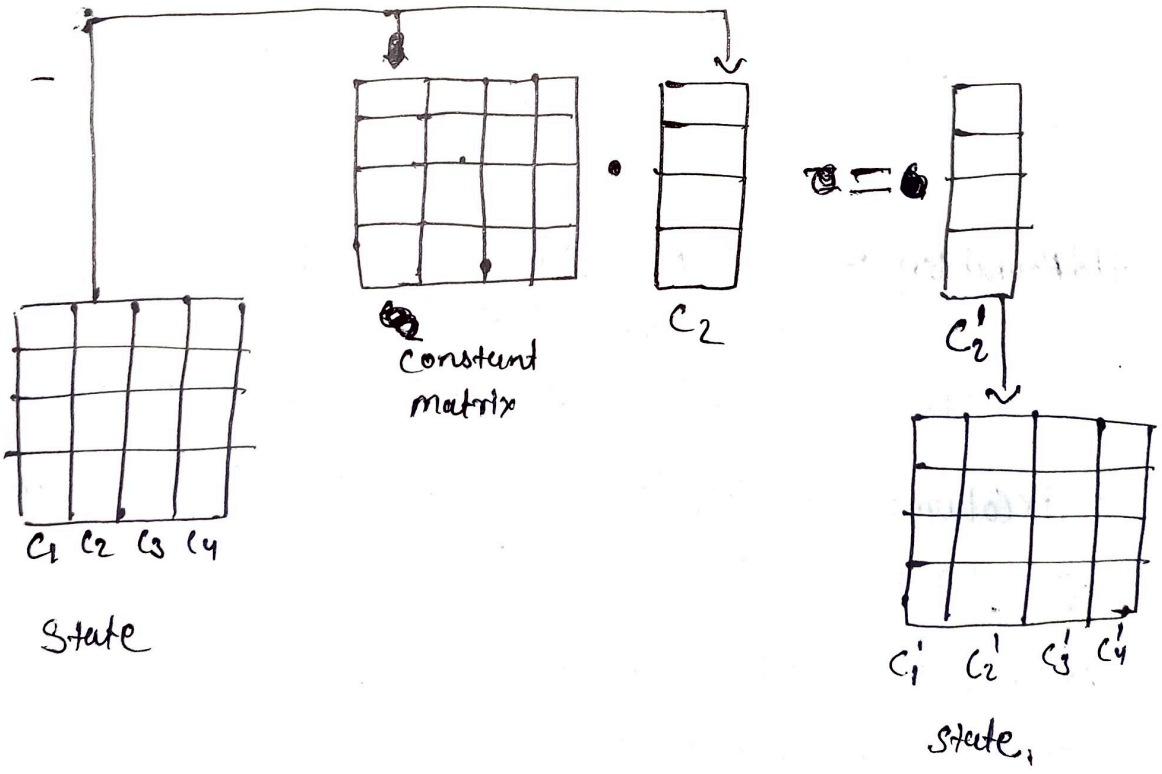
C $\qquad$ C^{-1}

Mix Columns :- The Mix Columns transformation operates at the column level. it transforms each column of State to a new column.

The transformation is actually the matrix multiplication of a State column by a constant square matrix.

The Bytes in the State column and constants matrix are interpreted as 8-bit words (or polynomials) with coefficients in $GF(2)$. And the multiplication of Bytes is done in $GF(2^8)$ with modulus (100011011) or $(x^8 + x^4 + x^3 + x + 1)$

& Addition is same as XORing a Bits.



Key Adding :- Probably the most important transformation is the one that includes the cipher key.

All Previous transformations use known algorithms that are invertible. If the cipher key is not added to the state at each round, It is very easy for the adversary to find the plaintext, given the ciphertext.

AES uses a process called key expansion that creates Nr+1 round keys from the cipher key. Each round key is 128 bit long It is treated as four 32-bit words.

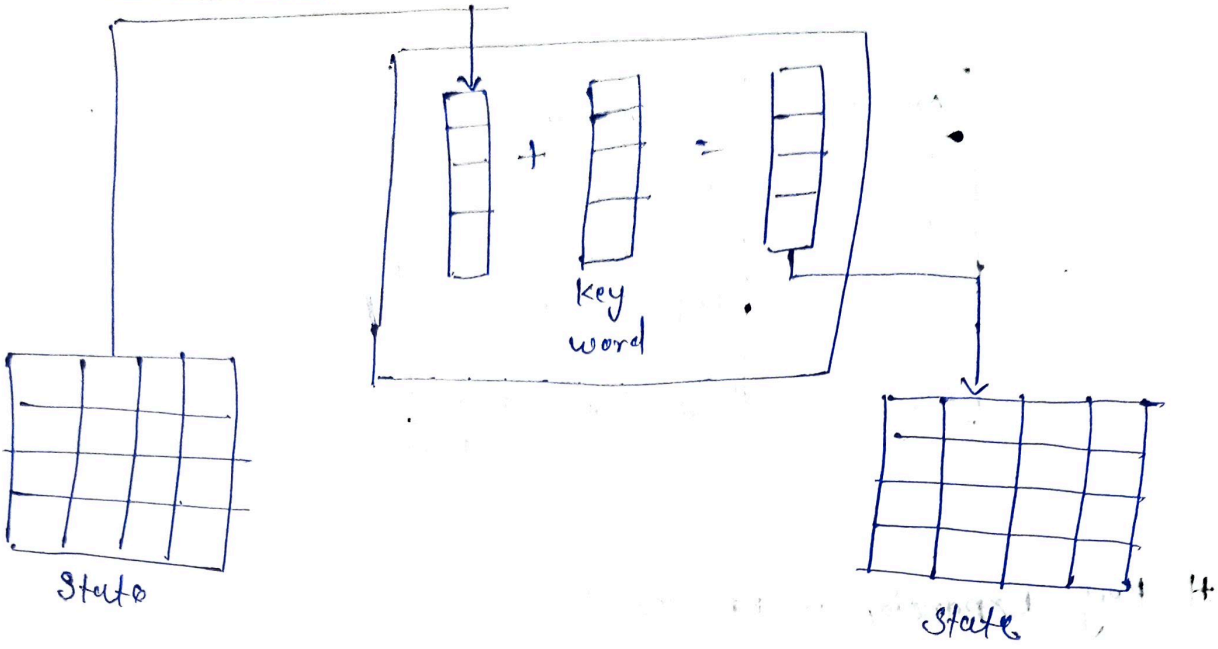
for the purpose of adding the key to the State, each word is considered as a column matrix.

AddRound key :- AddRoundkey also proceeds one column at a time.

It is similar to MixColumns in this. Respect MixColumn multiplies a constant Square matrix by each State column.

AddRoundkey adds a 'round key word' with each State column matrix.

That is the operation in MixColumn is Multiplication.
& the operation in AddRound key is Addition
So AddRound Key is transformation is inverse of itself.



Key Expansion :- To create a round key for each round AES uses a key-Expansion process.

If the Number of Rounds is Nr, the key-expansion creates Nr+1 128-bit round key from one single 128-bit cipher key.

The first round key is used for Pre-round transformation (AddRoundKey). the remaining round keys are used for the last transformation (Add round key) at the end of each round.

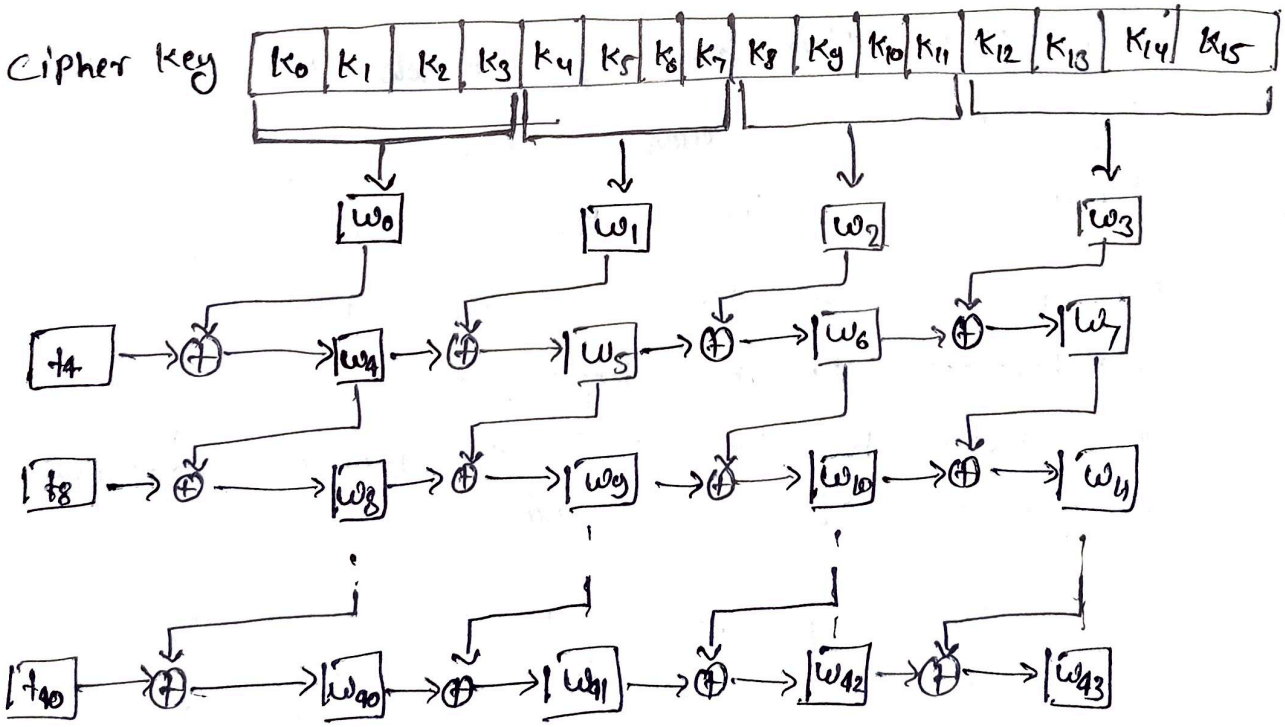
The key-Expansition creates round keys word by word, where a word is an array of four bytes.

In the AES-128 version (10 Rounds), there are 44 words. In the AES-192 version (12 Rounds), there are 52 words and in the AES-256 version (with 14 Rounds), there are 60 words. Each round key is made up of 4 words

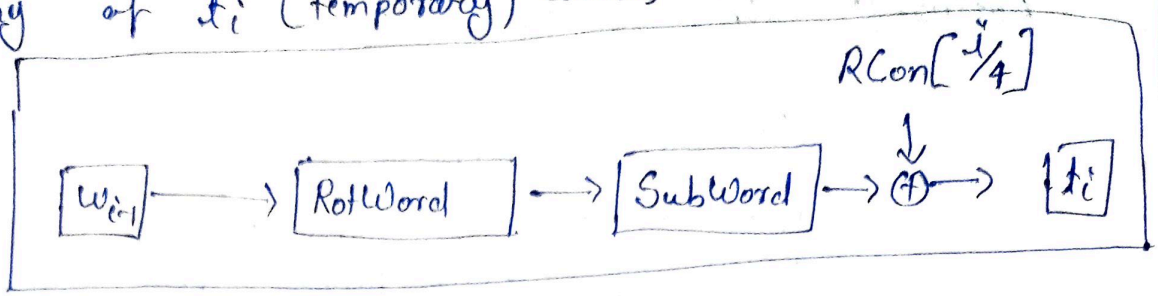
Rounds	words
Pre-Round	w_0 w_1 w_2 w_3
1	w_4 w_5 w_6 w_7
2	w_8 w_9 w_{10} w_{11}
...	
N_r	w_{4N_r} w_{4N_r+1} w_{4N_r+2} w_{4N_r+3}

Key Expansion in AES-128 :-

Let us show the creation of words for the AES-128 version, The Process for the other two versions are the same with some Slight Changes.



Making of t_i (temporary) words $i = 4 N_r$



The Process is as follow:-

① The first four words (w_0, w_1, w_2, w_3) are made from the cipher key.
The cipher key is an array of 16 (Bytes) (K_0 to K_{15}).
The first four Bytes (K_0 to K_3) become w_0.
the next four Bytes (K_4 to K_7) become w_1, and so on.
In other words, the concatenation of the words in this group replicates the cipher key.

② The rest of the words (w_i for $i = 4$ to 43) are made as follows:-

ⓐ If $(i \bmod 4) \neq 0$, $w_i = w_{i-1} \oplus w_{i-4}$
means word is made from the one at the left and the one at the top.

ⓑ If $(i \bmod 4) = 0$; $w_i = t \oplus w_{i-4}$,
Here t, a temporary word, is the result of applying two routines, SubWord and RotWord, on w_{i-1} and XORing the result with a round constants, RCon.
In other words, we have,

$$t = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{RCon}_{i/4}$$

RotWord:- The RotWord (rotate word) routine is similar to the ShiftRows transformation, but it is applied to only one row.

The routine takes a word as an array of four Bytes and shifts each byte to the left with wrapping.

SubWord:- The SubWord routine is similar to the SubBytes transformation, but it is applied only to four bytes.

The routine takes each byte in the word and substitutes another byte for it.

Round constants (RCon):- Each round constant, RCon is a 4-byte value in which the rightmost three bytes are always zero.

for AES-128 (10 Rounds) the values are

Round	Constant (RCon)	Round	Constant (RCon)
1	$(01\ 00\ 00\ 00)_{16}$	6	$(20\ 00\ 00\ 00)_{16}$
2	$(02\ 00\ 00\ 00)_{16}$	7	$(40\ 00\ 00\ 00)_{16}$
3	$(04\ 00\ 00\ 00)_{16}$	8	$(80\ 00\ 00\ 00)_{16}$
4	$(08\ 00\ 00\ 00)_{16}$	9	$(1B\ 00\ 00\ 00)_{16}$
5	$(10\ 00\ 00\ 00)_{16}$	10	$(36\ 00\ 00\ 00)_{16}$

OR we can calculate leftmost bit as below

$RC_1 \rightarrow x^{1-1} = x^0 \bmod \text{prime} = 1 = 0000\,0001 \rightarrow 1$

$RC_2 \rightarrow x^{2-1} = x \bmod \text{prime} = x = 0000\,0010 \rightarrow 2$

$RC_3 \rightarrow x^{3-1} = x^2 \bmod \text{prime} = x^2 = 0000\,0100 \rightarrow 04$

$RC_4 \rightarrow x^{4-1} = x^3 \bmod \text{prime} = x^3 = 0000\,1000 \rightarrow 08$

$RC_5 \rightarrow x^{5-1} = x^4 \bmod \text{prime} = x^4 = 0001\,0000 \rightarrow 10$

$RC_6 \longrightarrow x^{6-1} = x^5 \bmod \text{Prime} \longrightarrow 00100000 \longrightarrow 20$

$RC_7 \longrightarrow x^{7-1} = x^6 \bmod \text{Prime} \longrightarrow 01000000 \longrightarrow 40$

$RC_8 \longrightarrow x^{8-1} = x^7 \bmod \text{Prime} \longrightarrow 10000000 \longrightarrow 80$

$RC_9 \longrightarrow x^{9-1} = x^8 \bmod \text{Prime} \longrightarrow$
$x^4 + x^3 + x + 1 \longrightarrow 00011011 \longrightarrow 1B$

$RC_{10} \longrightarrow x^{10-1} = x^9 \bmod \text{prime} \longrightarrow x^5 + x^4 + x^2 + x \longrightarrow 00110110 \longrightarrow 36$

In other words The left most byte, which is called RC_i is actually x^{i-1}, where i is the round number. AES uses the irreducible polynomial $(x^8 + x^4 + x^3 + x + 1)$.

Key Expansion in AES-192 and AES-256 :-

Key-expansion algorithm in the AES-192 and AES-256 version are very similar to the key expansion algorithm in AES-128, with the following Differences:

① In AES-192, the words are generated in groups of six instead of four.

ⓐ The cipher key creates the first six words (w_0 to w_5)

ⓑ If $i \bmod 6 \neq 0$, $w_i \leftarrow w_{i-1} + w_{i-6}$

otherwise, $w_i \leftarrow t + w_{i-6}$

2. In AES-256 the words are generated in groups of eight instead of four.

a. The cipher key creates the first eight words (w_0 to w_7)

b. If $i \bmod 8 \neq 0$, $w_i \leftarrow w_{i-1} + w_{i-8}$; otherwise

$w_i \leftarrow t + w_{i-8}$

c. If $i \bmod 4 = 0$, but $i \bmod 8 \neq 0$ then

$$w_i = \text{SubWord}(w_{i-1}) + w_{i-8}$$

Key - Expansion Analysis :-

The key-expansion mechanism in AES has been designed to provide several features that thwart the cryptanalyst.

(1) If the attacker knows the part of key or word of key (cipher key)

He still need to find rest of the cipher key before finding the all round key.

This is because of the non-linearity provided by SubWord transformation. in the key expansion process.

(2) There is No serious weak keys in AES Unlike in DES

(3) key-expansion process can be easily implemented on all platforms.

④ Key-expansion routine can be implemented without storing a single table. all calculation can be done using the GF(2^8) & GF(2) fields

---

CIPHERS :-

→ AES is a Non-feistel cipher, which means that each transformation or group of transformation must be invertible. the cipher and the inverse cipher must use these operations in such a way that cancel each other.

→ The round key must be used in the reverse order.

→ Two different designs are suggests for different implimentation.

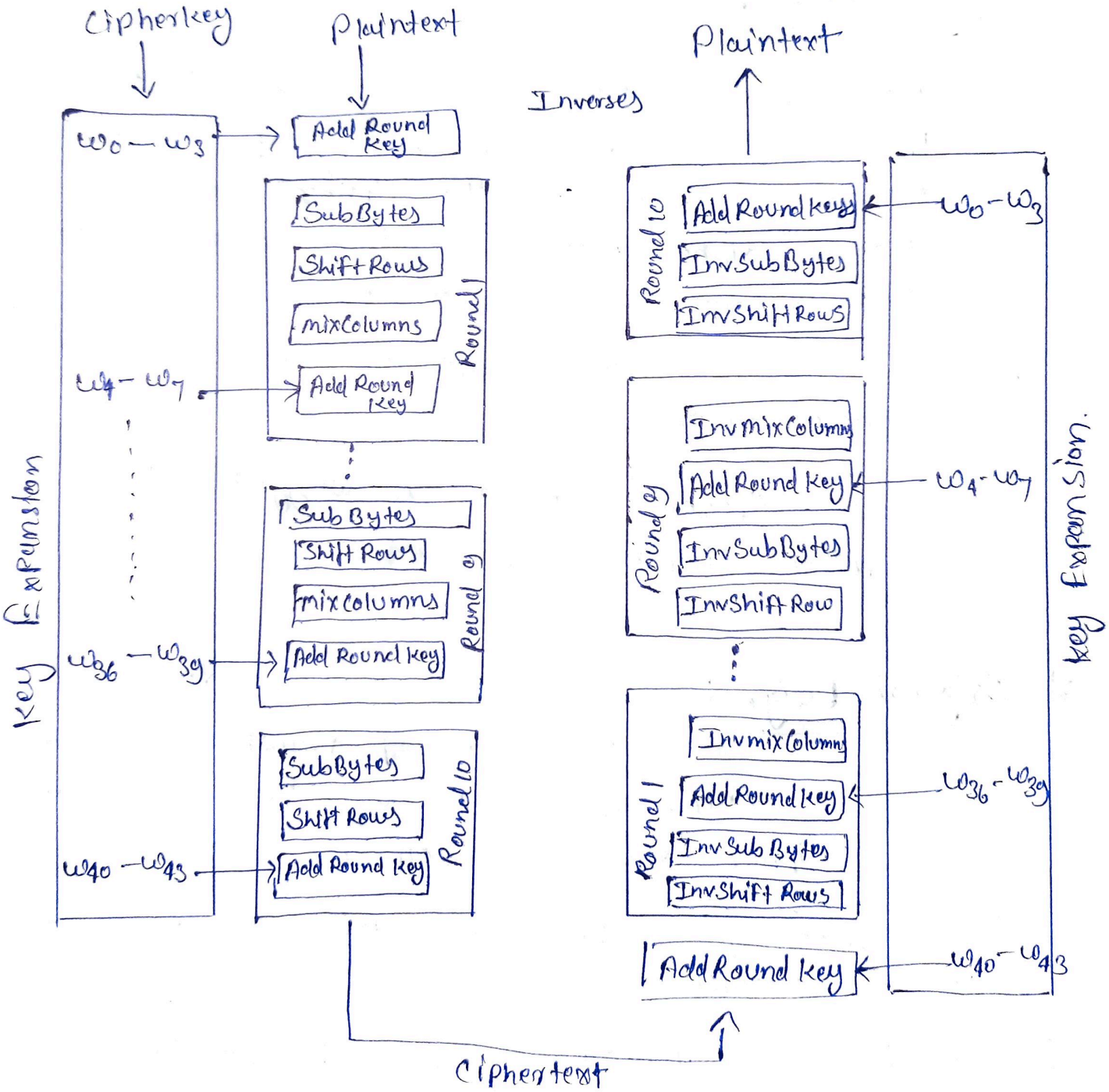
Original Design :- In the original design, the order of transformations in each round is not the same in the cipher and reverse cipher.

The order of SubBytes and ShiftRows is changed in the reverse cipher

& The order of MixColumns and AddRoundKey is changed in the reverse cipher.

This difference in the ordering is needed to make each transformation in the cipher aligned with its inverse in the reverse cipher. Round keys are used in reverse order.

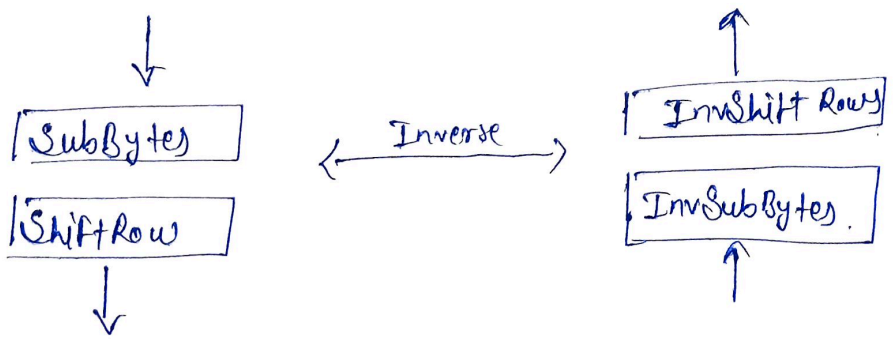
Encryption & Decryption Algorithm in the original design ~~is~~ are not similar.



Alternative Design :-

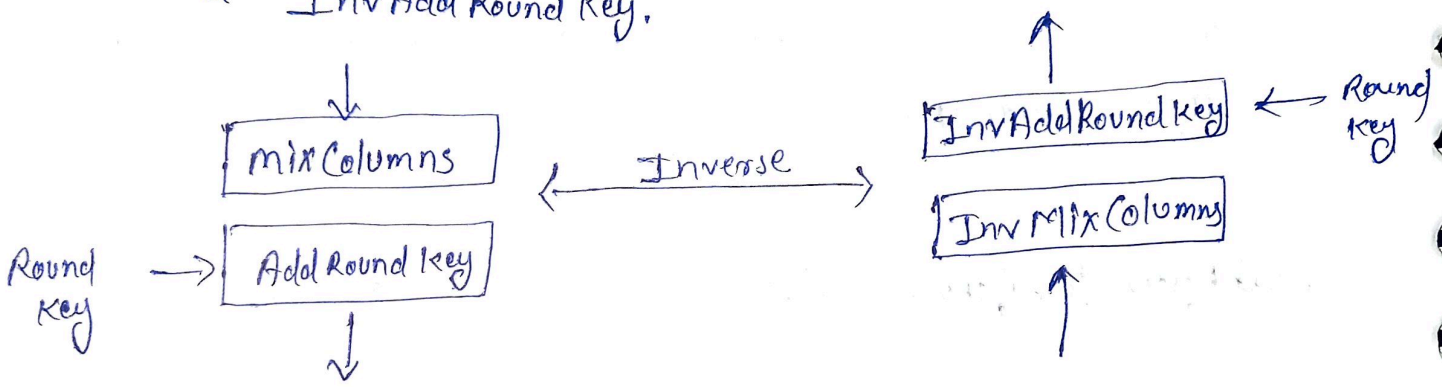
for those Application that prefer similar algorithms for encryption and decryption, a different inverse cipher was developed.
In this version, the transformations in the reverse cipher are rearranged to make the order of transformations the same in the cipher and reverse cipher. In this design, invertibility is Provided for a pair of transformations, not for single transformation..

SubBytes / ShiftRows Pairs :- SubBytes Change the contents of each bytes without changing the order of the bytes in the state. ShiftRows change the order of the bytes in the state without changing the content of the bytes.
This implies that we can change the order of these two transformation in the inverse cipher without affecting the invertibility of the whole algorithm.



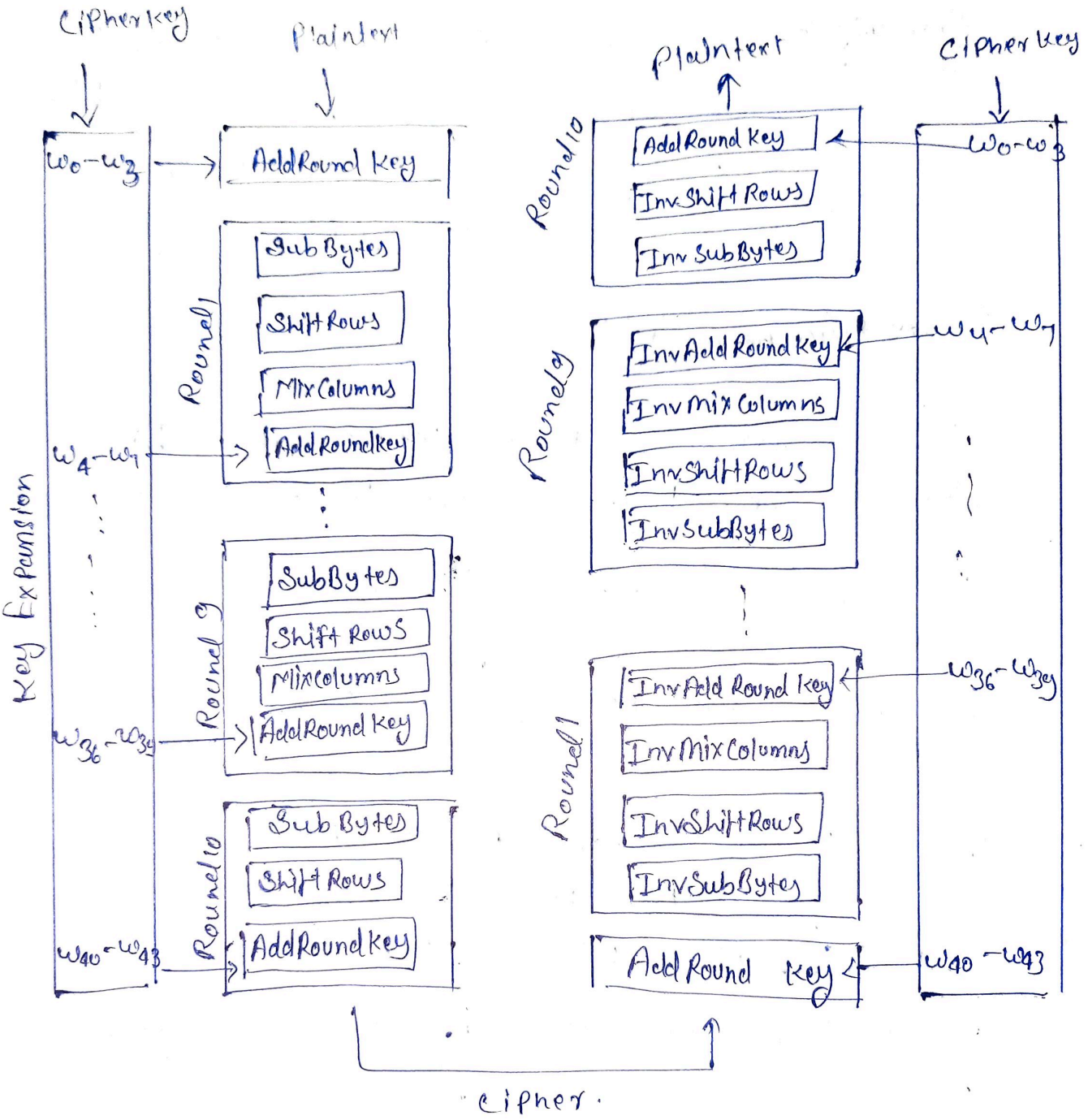
MixColumns / AddRoundKey Pair

Here the two involved transformations are of different nature. However, the pair can become inverses of each other if we multiply the key matrix by the inverse of the constant matrix used in MixColumns transformation. We call the new transformation a InvAddRoundKey.



Changing Key Expansion Algorithm :-

Instead of using InvRoundKey transformation in the reverse cipher, the key-expansion algorithm can be changed to create a different set of Round Key for the inverse cipher.


Cipher key
Plaintext
Key Expansion
$w_0 - w_3$
AddRound key
Round 1
SubBytes
ShiftRows
MixColumns
AddRoundkey
$w_4 - w_7$
Round 9
SubBytes
Shift Rows
MixColumns
AddRound key
$w_{36} - w_{39}$
Round 10
SubBytes
ShiftRows
AddRoundkey
$w_{40} - w_{43}$
Plaintext
Cipher key
Round 10
AddRound key
InvShiftRows
InvSubBytes
$w_0 - w_3$
Round 9
InvAddRound key
InvMixColumns
InvShiftRows
InvSubBytes
$w_4 - w_7$
$w_{36} - w_{39}$
Round 1
InvAddRound key
InvMixColumns
InvShiftRows
InvSubBytes
AddRound key
$w_{40} - w_{43}$
cipher

Analysis of AES :-

Security :- AES was designed after DES. ~~due to~~ Most of the known attacks on DES were already tested on AES. None of them has broken the security of AES so far.

Brute force Attack :-

AES is definitly more secure then DES due to the larger-size-key (128, 192 and 256) bits

for DES we need 2^{56} tests to find key.

for AES we need 2^{128} tests to find key

This means If someone Breaks DES in 1 Second then he need $(2^{72} \times 1)$ seconds to ~~to~~ Break AES. which is impossible.

The lack of weak keys is another advantage of AES over DES.

Statical Attack :- ~~AES was designed after DES~~.

The Strong diffusion & confusion Provided by the combination of the SubBytes, ShiftRows, and MixColumns transformations remores any frequency pattern in the plaintext. Numerous tests have failed to do statical analysis of the ciphertext.