Data Structures
ADVANCED ENCRYPTION STANDARD (AES)

In this assignment you have to implement AES algorithm. The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext. The AES algorithm can use cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

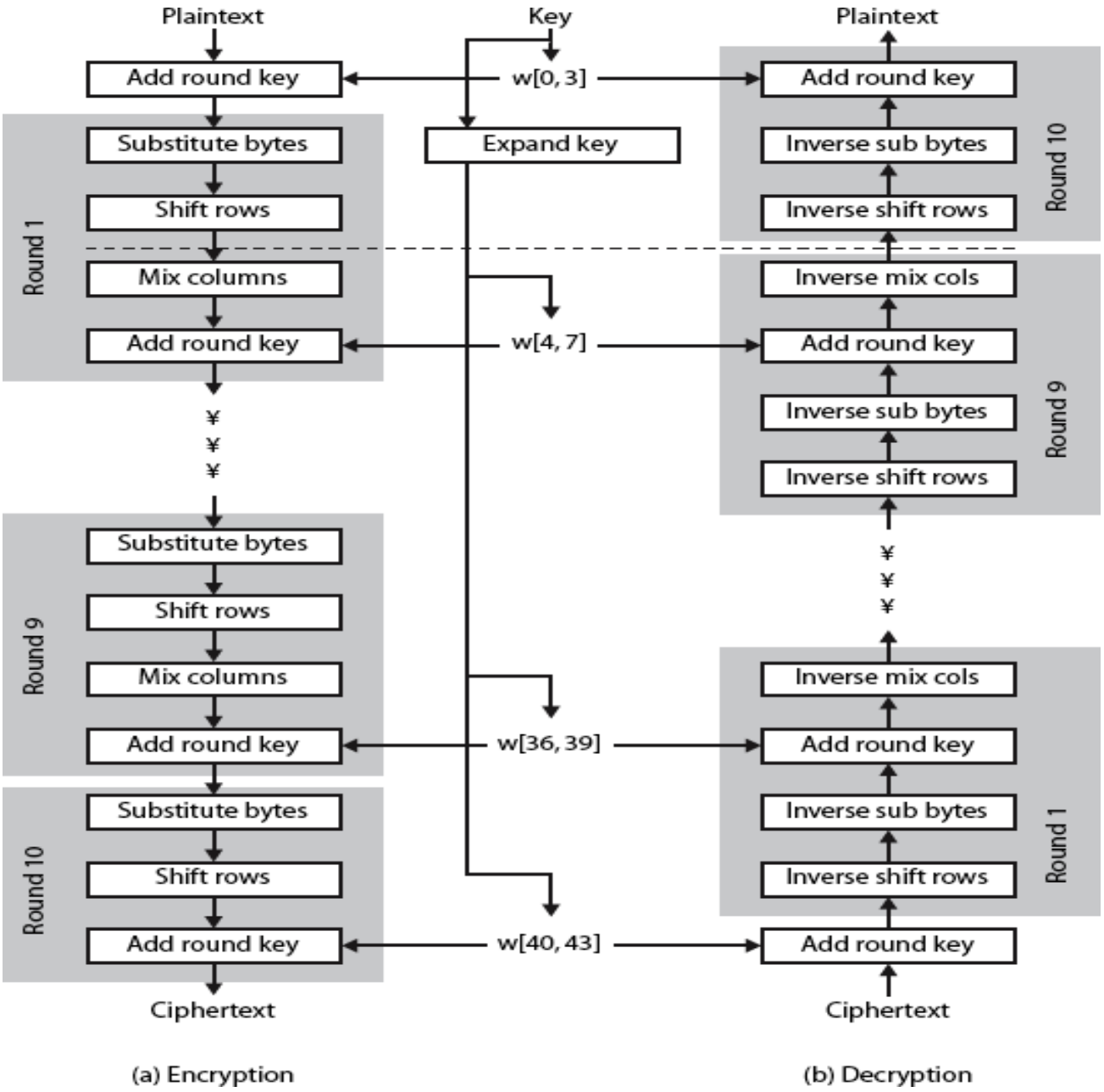
Operation of AES

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix.

AES is variable and depends on the length of the key. AES uses **10 rounds for 128-bit keys**, **12 rounds for 192-bit keys and 14 rounds for 256-bit keys**. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

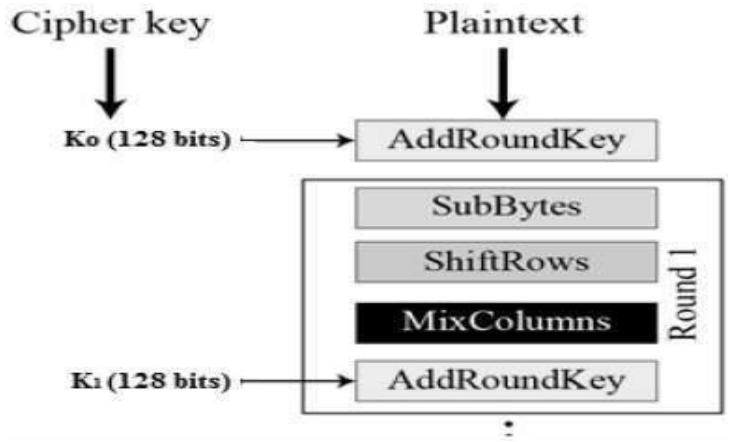
The schematic of AES structure is given in the following illustration



Note: There are only 3 steps in last round.

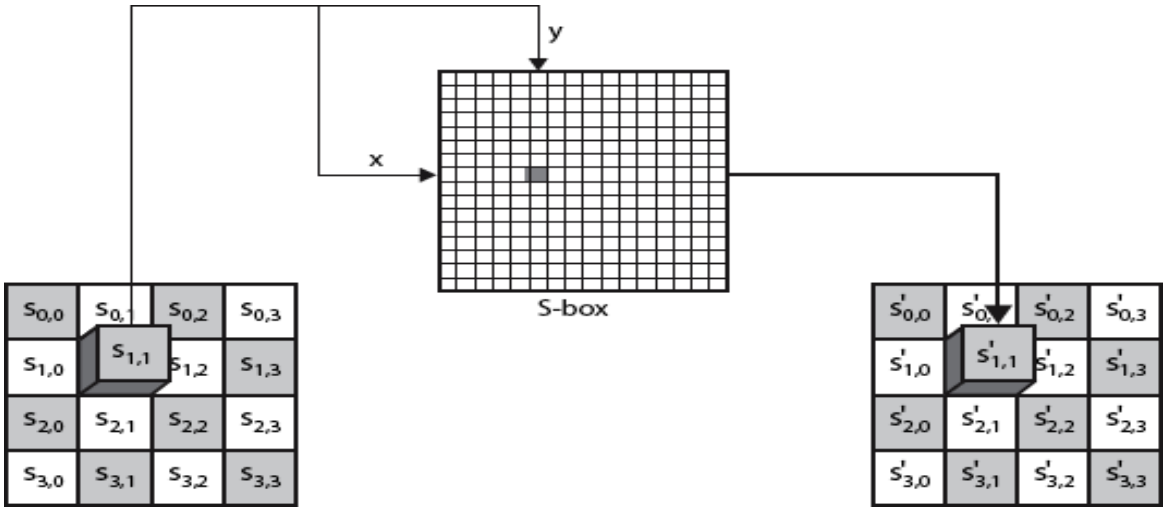
Encryption Process

AES consist of different rounds depends on size of key. Here, we restrict you to just implement 16-byte (128 bits) key which will have 10 rounds. Each round comprises of four sub-processes. The first-round process is depicted below:



ByteSubstitution ():

In this transformation step, each of the byte in the state matrix is replaced with another byte as per the S-box (Substitution Box) as its name implied. The result is in a matrix of four rows and four columns.



EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

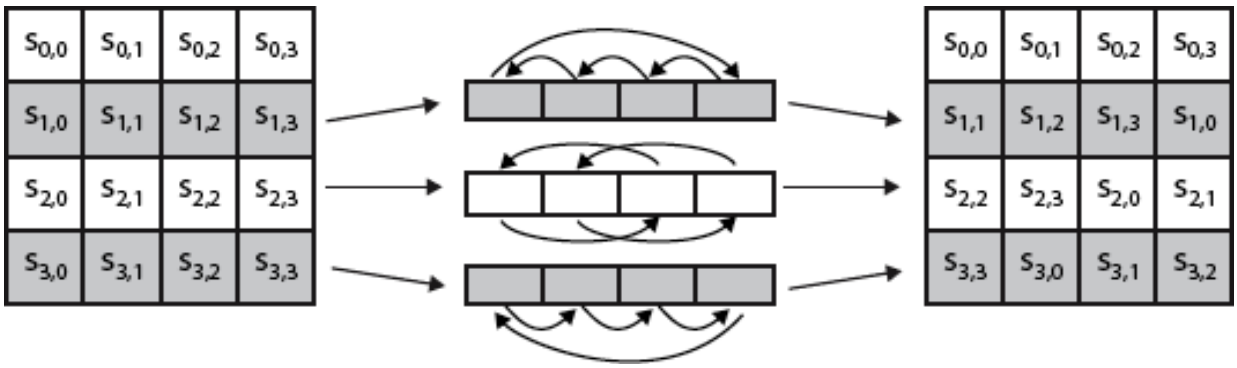
→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Shiftrows ():

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows −

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.



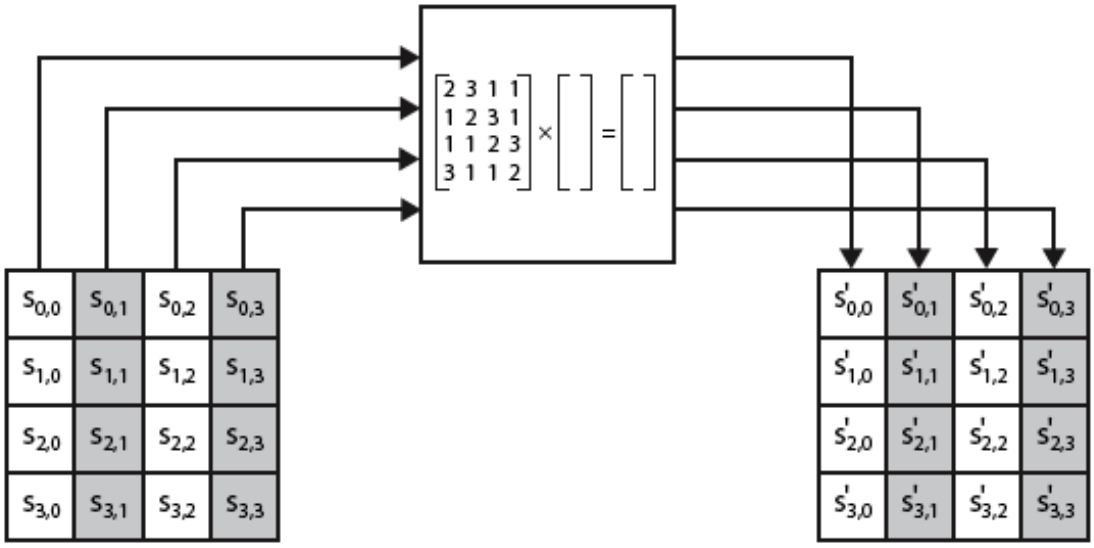
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

MixColumns ():

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.



Example:

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$		2	3	1	1		$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$		1	2	3	1		$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	X	1	1	2	3	=	$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$		3	1	1	2		$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

Note: The matrix with whom you have to multiply will remain same (given in above example).

Addroundkey ():

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round, then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$		w_i	w_{i+1}	w_{i+2}	w_{i+3}		$s'_{0,0}$	$s'_{0,1}$	$s'_{0,2}$	$s'_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$							$s'_{1,0}$	$s'_{1,1}$	$s'_{1,2}$	$s'_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$	⊕					=	$s'_{2,0}$	$s'_{2,1}$	$s'_{2,2}$	$s'_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$							$s'_{3,0}$	$s'_{3,1}$	$s'_{3,2}$	$s'_{3,3}$

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −

- InvAddround key ()
- InvMixcolumns ()
- InvShiftrows ()
- InvBytesubstitution ()

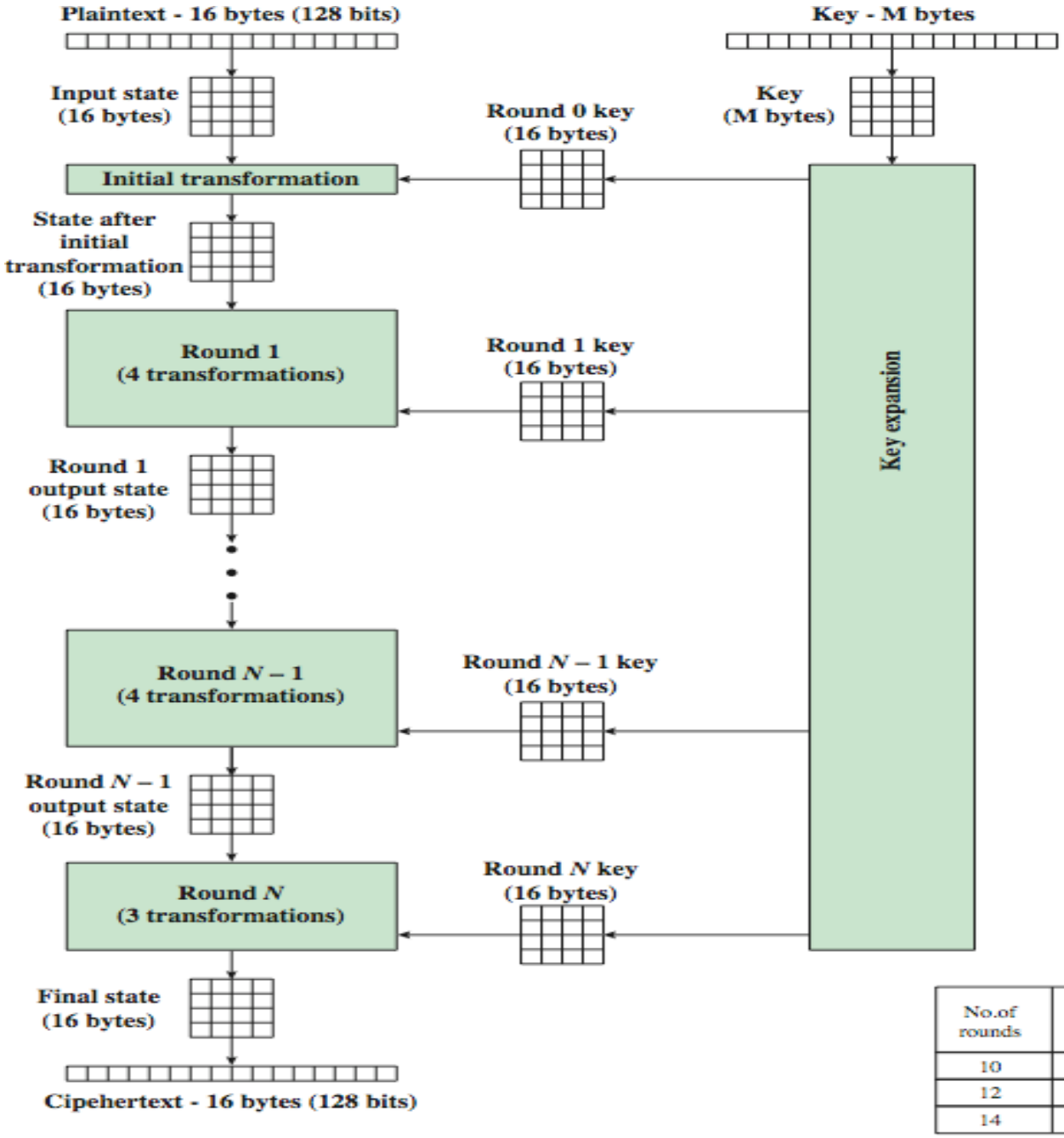
Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms need to be separately implemented, although they are very closely related. Round depends on key size. If key size is 16 bytes then round will ne 10, for 24 bytes rounds will be 12 and for 32 bytes its 14.

Key Expansion:

The AES algorithm takes the Cipher Key, K, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb (Nr + 1) words: the algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [wi], with i in the range 0 £ i < Nb(Nr + 1). Following is the Pseudo code for the key expansion:

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end

Note that Nk=4, 6, and 8 do not all have to be implemented;
they are all included in the conditional statement above for
conciseness.    Specific implementation requirements for the
Cipher Key are presented in Sec. 6.1.
```

A complete process model of AES encryption:



No.of rounds	Key Length (bytes)
10	16
12	24
14	32

***Note**: You can understand the process more clearly by running_example.pdf*

Following matrices will be fixed in all cases:

1. **S-BOX**

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

2. **KEY**

   Thats my Kung Fu (16 ASCII characters, 1 byte each)

3. **MATRIC IN MIX COLOUMN**

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Useful Links:

https://www.comparitech.com/blog/information-security/what-is-aes-encryption/

https://www.commonlounge.com/discussion/e32fdd267aaa4240a4464723bc74d0a5

https://cboard.cprogramming.com/c-programming/87805-[tutorial]-implementing-advanced-encryption-standard.html

happy coding □