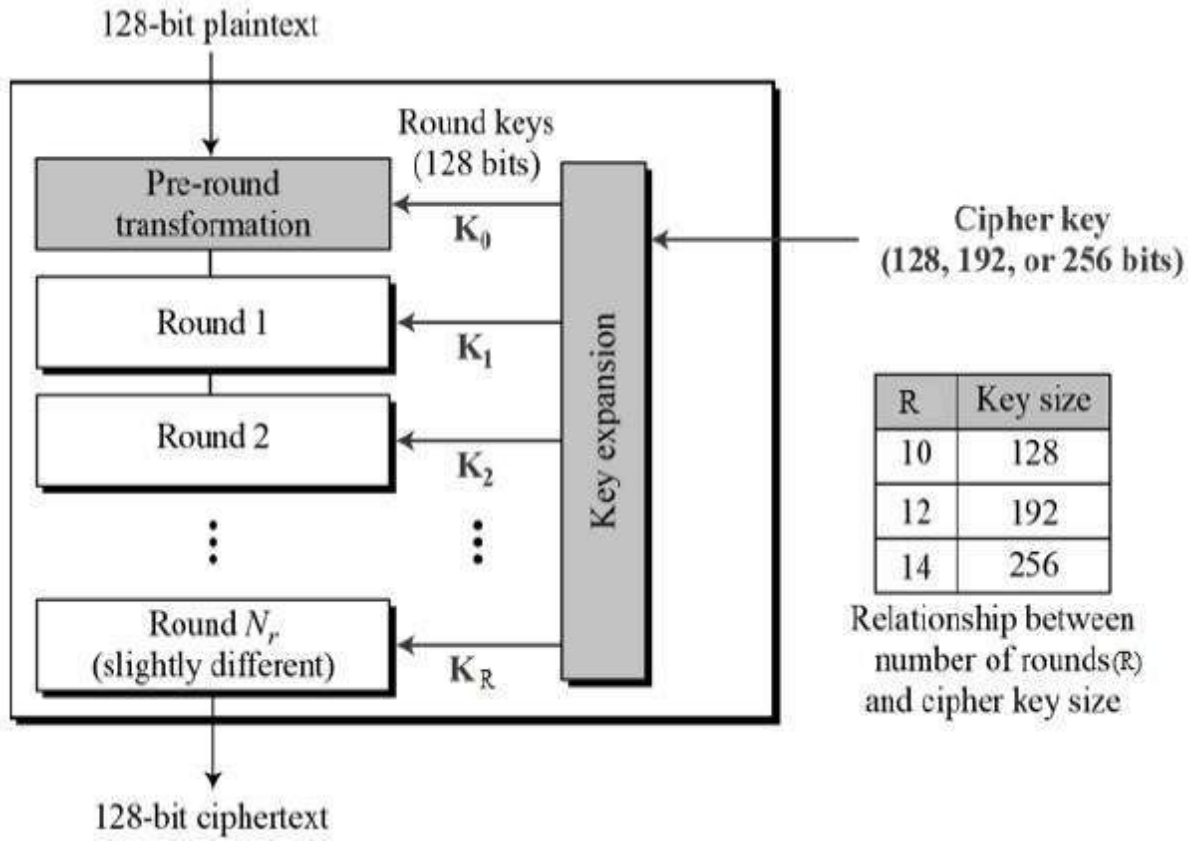
Programming Exercise -2

Advanced Encryption Standard (AES)

Manavjeet Singh (2018295)
Rhythm Gupta (2018082)

ABOUT ADVANCED ENCRYPTION STANDARD (AES)

AES is a symmetric block cipher which works on fixed length input and keys.

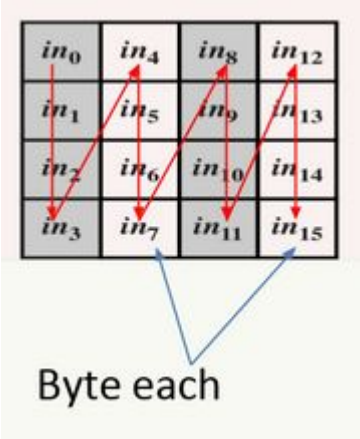


R	Key size
10	128
12	192
14	256

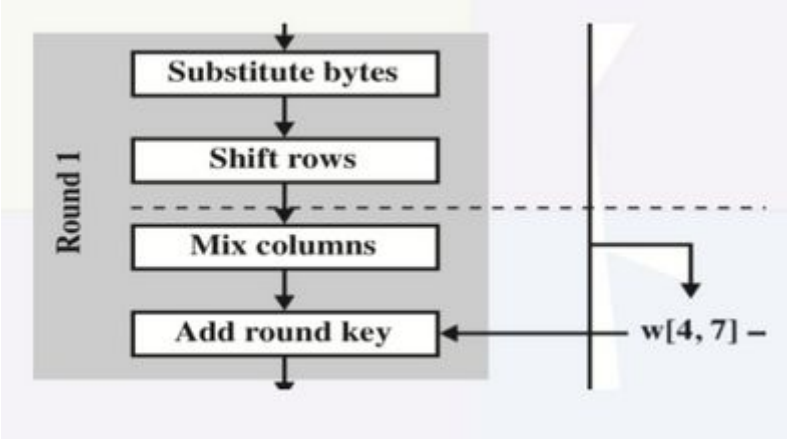
Relationship between number of rounds (R) and cipher key size

Pre Round Transformation

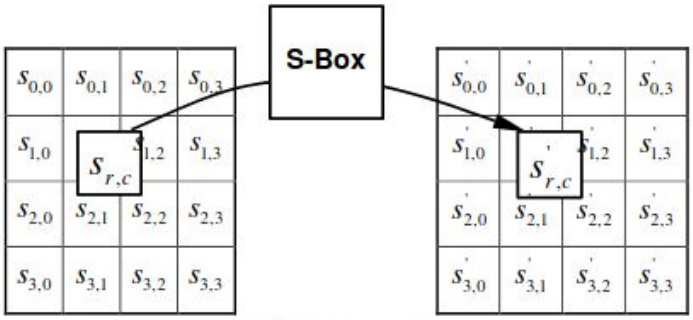
1. Convert 128 bit plaintext to 4*4 array of 1 byte entries
2. Perform addRoundKey() as described later in this document



Description of each round



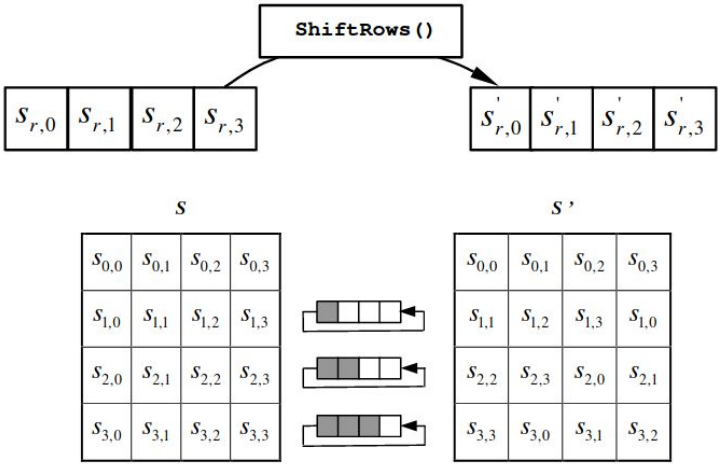
Substitute Bytes



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

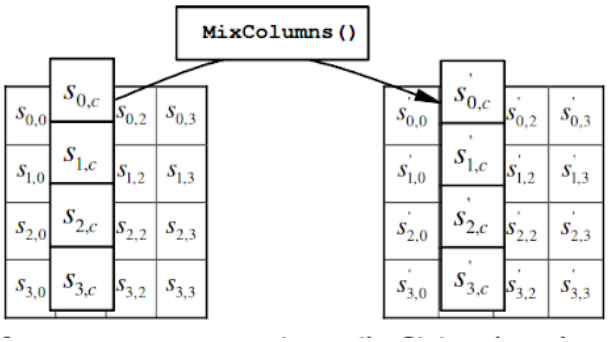
Fig: Sbox , substitute 0xab by Sbox[0xa][0xb] where a, b are 4 bit hexadecimal digits

Shift Rows



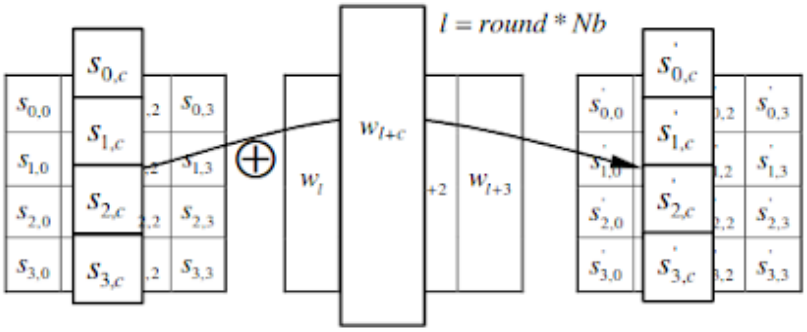
Mix Columns

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < \boldsymbol{Nb}.$$

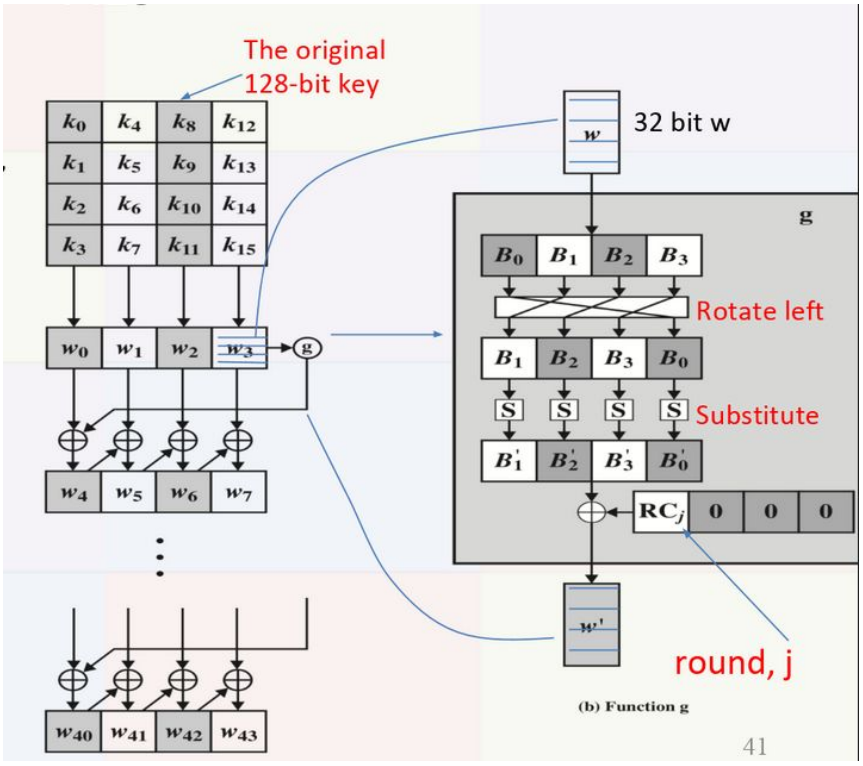


Add Round key

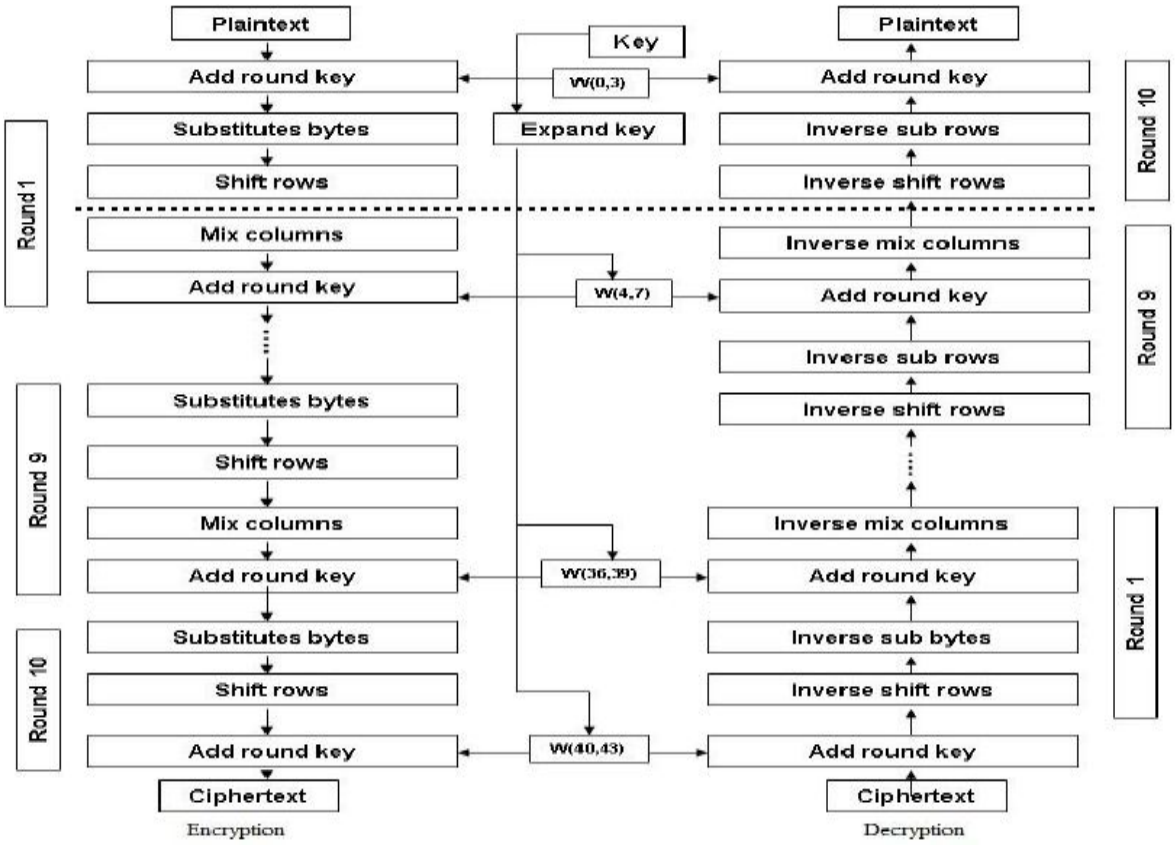
$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*Nb+c}] \quad \text{for } 0 \le c < Nb,$$



Sub Key Generation



Decryption



Decryption steps are just opposite of what we did for encryption, with a few changes.

1. Shift rows is performed in opposite direction
2. Mix columns and substitute bytes use the inverse of matrices used in encryption

IMPLEMENTATION DETAILS

We have used 128 bit plaintext and 128 bit keys.
Our code is based on python3
For effectively managing 128 bit numbers, we have used bitarray library.
For carrying out multiplication in GF (2^8), we have used pyfinite library.

AES_Encryptor.py -> Encrypts the given 128 bit plaintext into 128 bit ciphertext
AES_Decryptor.py -> Decrypts the given 128 bit ciphertext into 128 bit plaintext
main.py -> Wrapper file

EXAMPLE 1

Key: 2b7e151628aed2a6abf7158809cf4f3c
Plaintext: 3243f6a8885a308d313198a2e0370734
All States(**encryption**):
Input
0x32 0x88 0x31 0xe0
0x43 0x5a 0x31 0x37
0xf6 0x30 0x98 0x7
0xa8 0x8d 0xa2 0x34

After initial add key round
0x19 0xa0 0x9a 0xe9
0x3d 0xf4 0xc6 0xf8
0xe3 0xe2 0x8d 0x48
0xbe 0x2b 0x2a 0x8

Round 1
0xa4 0x68 0x6b 0x2
0x9c 0x9f 0x5b 0x6a
0x7f 0x35 0xea 0x50
0xf2 0x2b 0x43 0x49

Round 2
0xaa 0x61 0x82 0x68
0x8f 0xdd 0xd2 0x32
0x5f 0xe3 0x4a 0x46
0x3 0xef 0xd2 0x9a

Round 3
0x48 0x67 0x4d 0xd6
0x6c 0x1d 0xe3 0x5f
0x4e 0x9d 0xb1 0x58
0xee 0xd 0x38 0xe7

Round 4
0xe0 0xc8 0xd9 0x85
0x92 0x63 0xb1 0xb8
0x7f 0x63 0x35 0xbe
0xe8 0xc0 0x50 0x1

Round 5
0xf1 0xc1 0x7c 0x5d
0x0 0x92 0xc8 0xb5
0x6f 0x4c 0x8b 0xd5
0x55 0xef 0x32 0xc

Round 6
0x26 0x3d 0xe8 0xfd
0xe 0x41 0x64 0xd2
0x2e 0xb7 0x72 0x8b

0x17 0x7d 0xa9 0x25

Round 7
0x5a 0x19 0xa3 0x7a
0x41 0x49 0xe0 0x8c
0x42 0xdc 0x19 0x4
0xb1 0x1f 0x65 0xc

Round 8
0xea 0x4 0x65 0x85
0x83 0x45 0x5d 0x96
0x5c 0x33 0x98 0xb0
0xf0 0x2d 0xad 0xc5

Round 9
0xeb 0x59 0x8b 0x1b
0x40 0x2e 0xa1 0xc3
0xf2 0x38 0x13 0x42
0x1e 0x84 0xe7 0xd2

Output
0x39 0x2 0xdc 0x19
0x25 0xdc 0x11 0x6a
0x84 0x9 0x85 0xb
0x1d 0xfb 0x97 0x32

Ciphertext: 3925841d02dc09fbdc118597196a0b32

All States(**Decryption**)
Input
0x39 0x2 0xdc 0x19
0x25 0xdc 0x11 0x6a
0x84 0x9 0x85 0xb
0x1d 0xfb 0x97 0x32

After round 1
0xeb 0x59 0x8b 0x1b
0x40 0x2e 0xa1 0xc3
0xf2 0x38 0x13 0x42
0x1e 0x84 0xe7 0xd2

Round 2
0xea 0x4 0x65 0x85
0x83 0x45 0x5d 0x96
0x5c 0x33 0x98 0xb0
0xf0 0x2d 0xad 0xc5

Round 3
0x5a 0x19 0xa3 0x7a
0x41 0x49 0xe0 0x8c

0x42 0xdc 0x19 0x4
0xb1 0x1f 0x65 0xc

Round 4
0x26 0x3d 0xe8 0xfd
0xe 0x41 0x64 0xd2
0x2e 0xb7 0x72 0x8b
0x17 0x7d 0xa9 0x25

Round 5
0xf1 0xc1 0x7c 0x5d
0x0 0x92 0xc8 0xb5
0x6f 0x4c 0x8b 0xd5
0x55 0xef 0x32 0xc

Round 6
0xe0 0xc8 0xd9 0x85
0x92 0x63 0xb1 0xb8
0x7f 0x63 0x35 0xbe
0xe8 0xc0 0x50 0x1

Round 7
0x48 0x67 0x4d 0xd6
0x6c 0x1d 0xe3 0x5f
0x4e 0x9d 0xb1 0x58
0xee 0xd 0x38 0xe7

Round 8
0xaa 0x61 0x82 0x68
0x8f 0xdd 0xd2 0x32
0x5f 0xe3 0x4a 0x46
0x3 0xef 0xd2 0x9a

Round 9
0xa4 0x68 0x6b 0x2
0x9c 0x9f 0x5b 0x6a
0x7f 0x35 0xea 0x50
0xf2 0x2b 0x43 0x49

Round 10
0x19 0xa0 0x9a 0xe9
0x3d 0xf4 0xc6 0xf8
0xe3 0xe2 0x8d 0x48
0xbe 0x2b 0x2a 0x8

After last add key
0x32 0x88 0x31 0xe0
0x43 0x5a 0x31 0x37
0xf6 0x30 0x98 0x7
0xa8 0x8d 0xa2 0x34

The result of decryption is equal to plaintext.

EXAMPLE 2
Key: 000102030405060708090a0b0c0d0e0f
Plaintext: 00112233445566778899aabbccddeeff
All States(**Encryption**)

Input
0x0 0x44 0x88 0xcc
0x11 0x55 0x99 0xdd
0x22 0x66 0xaa 0xee
0x33 0x77 0xbb 0xff

After initial add key round
0x0 0x40 0x80 0xc0
0x10 0x50 0x90 0xd0
0x20 0x60 0xa0 0xe0
0x30 0x70 0xb0 0xf0

Round 1
0x89 0x85 0x2d 0xcb
0xd8 0x5a 0x18 0x12
0x10 0xce 0x43 0x8f
0xe8 0x68 0xd8 0xe4

Round 2
0x49 0x55 0xda 0x1f
0x15 0xe5 0xca 0xa
0x59 0xd7 0x94 0x63
0x8f 0xa0 0xfa 0xf7

Round 3
0xfa 0x25 0x40 0x57
0x63 0xb3 0x66 0x24
0x6a 0x39 0x8a 0x4d
0x28 0xc9 0x31 0x17

Round 4
0x24 0x69 0x6e 0x88
0x72 0x66 0xd2 0x42
0x40 0xb3 0x75 0x5b
0x23 0xfa 0x32 0x6c

Round 5
0xc8 0x9b 0x25 0xb0
0x16 0x7a 0x2 0x26
0x77 0xc9 0x79 0x19
0xbc 0x3b 0x92 0x96

Round 6
0xc6 0xf7 0xcc 0x84
0x2f 0x5e 0x79 0xf9
0xe1 0xed 0x39 0xcf
0x9 0xc3 0x5d 0x5d

Round 7
0xd1 0x79 0xb4 0xd6
0x87 0xc4 0x55 0x6f
0x6c 0x30 0x94 0xf4
0xf 0xa 0xad 0x1f

Round 8
0xfd 0x5 0x35 0xf1
0xe3 0xe5 0x47 0xfe
0xba 0xd0 0x96 0x37
0xd2 0xd7 0x4e 0xf1

Round 9
0xbd 0xf2 0xb 0x8b
0x6e 0xb5 0x61 0x10
0x7c 0x77 0x21 0xb6
0x3d 0x9e 0x6e 0x89

Round 10
0x69 0x6a 0xd8 0x70
0xc4 0x7b 0xcd 0xb4
0xe0 0x4 0xb7 0xc5
0xd8 0x30 0x80 0x5a

Ciphertext: 69c4e0d86a7b0430d8cdb78070b4c55a
All States(**Decryption**)
Input
0x69 0x6a 0xd8 0x70
0xc4 0x7b 0xcd 0xb4
0xe0 0x4 0xb7 0xc5
0xd8 0x30 0x80 0x5a

After round 1
0xbd 0xf2 0xb 0x8b
0x6e 0xb5 0x61 0x10
0x7c 0x77 0x21 0xb6
0x3d 0x9e 0x6e 0x89

Round 2
0xfd 0x5 0x35 0xf1
0xe3 0xe5 0x47 0xfe
0xba 0xd0 0x96 0x37
0xd2 0xd7 0x4e 0xf1

Round 3
0xd1 0x79 0xb4 0xd6
0x87 0xc4 0x55 0x6f
0x6c 0x30 0x94 0xf4
0xf 0xa 0xad 0x1f

Round 4
0xc6 0xf7 0xcc 0x84
0x2f 0x5e 0x79 0xf9
0xe1 0xed 0x39 0xcf
0x9 0xc3 0x5d 0x5d

Round 5
0xc8 0x9b 0x25 0xb0
0x16 0x7a 0x2 0x26
0x77 0xc9 0x79 0x19
0xbc 0x3b 0x92 0x96

Round 6
0x24 0x69 0x6e 0x88
0x72 0x66 0xd2 0x42
0x40 0xb3 0x75 0x5b
0x23 0xfa 0x32 0x6c

Round 7
0xfa 0x25 0x40 0x57
0x63 0xb3 0x66 0x24
0x6a 0x39 0x8a 0x4d
0x28 0xc9 0x31 0x17

Round 8
0x49 0x55 0xda 0x1f
0x15 0xe5 0xca 0xa
0x59 0xd7 0x94 0x63
0x8f 0xa0 0xfa 0xf7

Round 9
0x89 0x85 0x2d 0xcb
0xd8 0x5a 0x18 0x12
0x10 0xce 0x43 0x8f
0xe8 0x68 0xd8 0xe4

Round 10
0x0 0x40 0x80 0xc0
0x10 0x50 0x90 0xd0
0x20 0x60 0xa0 0xe0
0x30 0x70 0xb0 0xf0

After last add key
0x0 0x44 0x88 0xcc

0x11 0x55 0x99 0xdd
0x22 0x66 0xaa 0xee
0x33 0x77 0xbb 0xff

The deciphered text is equal to the plaintext.

Both the examples could be verified from (https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf), appendix B and C.

HOW TO RUN

Installing requirements:

- python3
- pip3 install bitarray
- pip3 install pyfinite

Creating random key(128 bits): Using the python script *'genRandKey.py'* user can generate pseudo random 128 bit keys.

Prerequisites
Create an ASCII text file with the first line of a file containing the key and the second line containing the Plaintext/Ciphertext in the same folder as *'main.py'*.

Encrypting OR Decrypting:

- Run script *'main.py'* and enter the name of the text file you want to import.
- Enter 1 to encrypt the imported plaintext or 2 to decrypt the imported ciphertext with the imported key.

References and Image Credits

1. CSE 350 slides for IIITD monsoon 2020 offering
2. AES official paper : https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf