

Name : Harsh Verdhan Singh

Roll No : 20mcs009

## ReadMe :

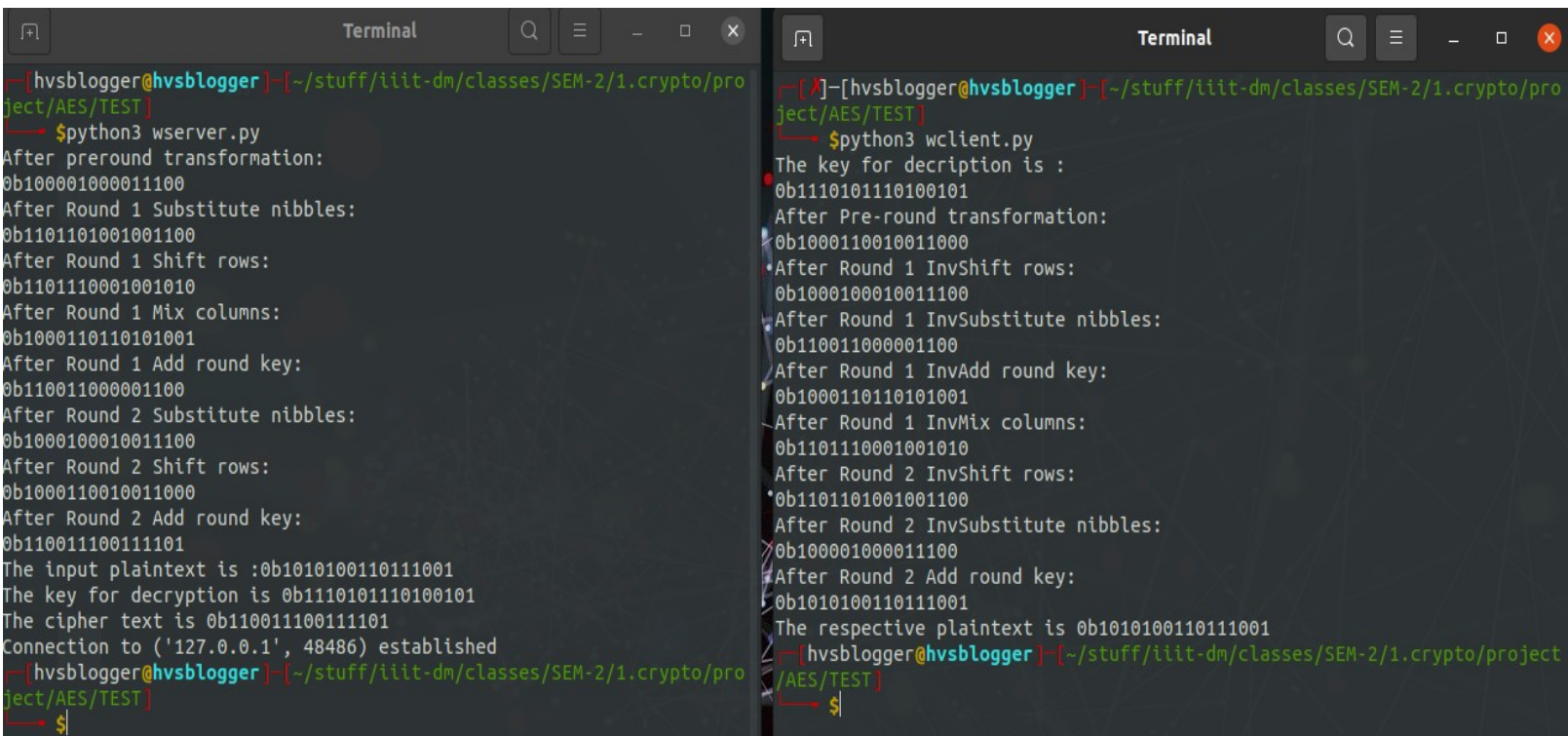
In this zip file there are two python file client.py and server.py. Client.py is used for client side socket programming and server.py is used for server side socket programming.

In both file the implementation of AES is done as given in the assignment.

For execution initially the server file must be run else on running the client file first it will show error of refused request.

Here the key and plaintext both are of 16 bit bin format.

The execution step is shown below : Here I also print each step that what each step is doing .



The image displays two terminal windows side-by-side, showing the execution of a server and a client for AES encryption/decryption.

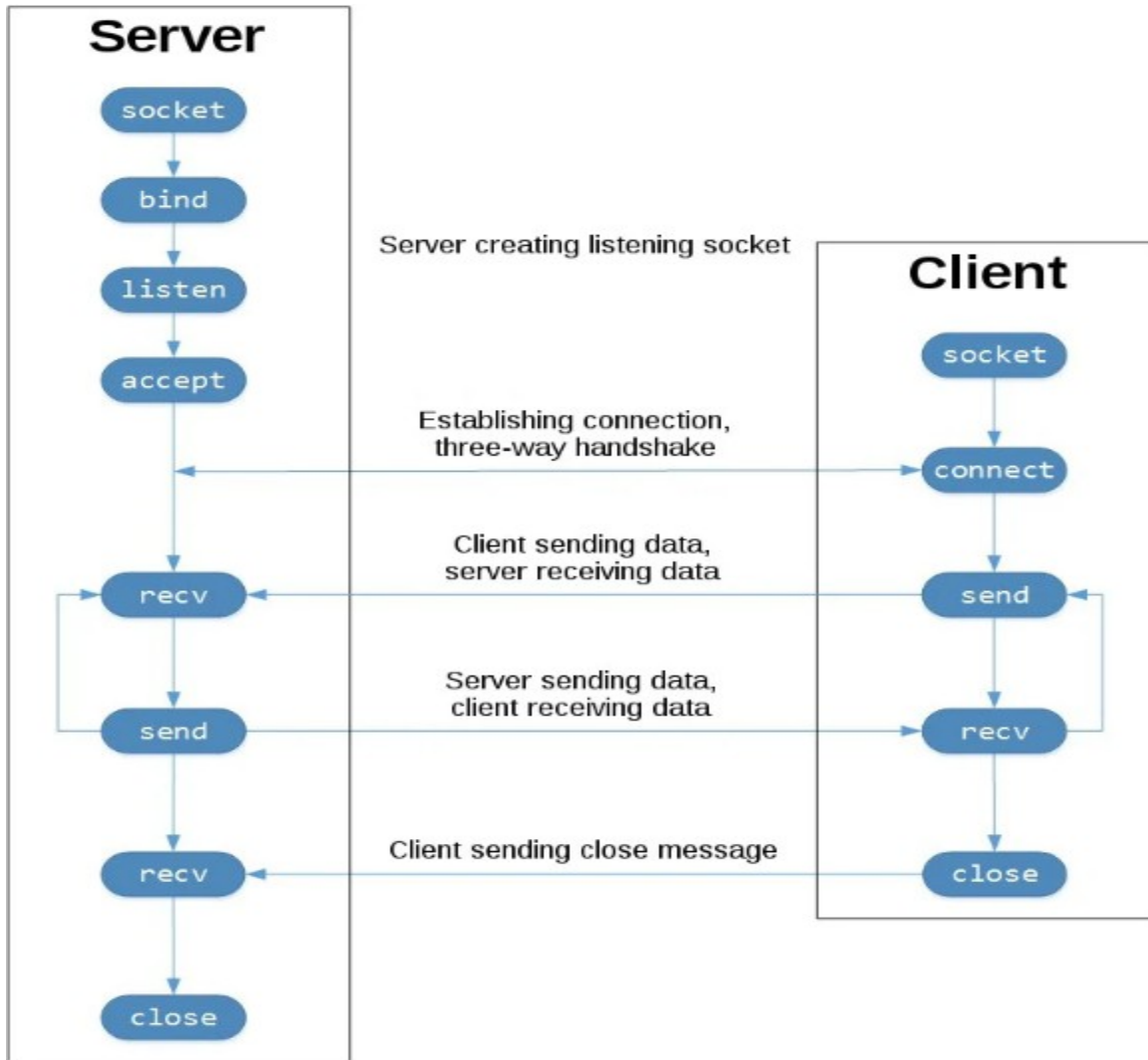
**Left Terminal (Server):**

```
hvsblogger@hvsblogger:~/stuff/iit-dm/classes/SEM-2/1.crypto/project/AES/TEST$ python3 wserver.py
After preround transformation:
0b100001000011100
After Round 1 Substitute nibbles:
0b1101101001001100
After Round 1 Shift rows:
0b1101110001001010
After Round 1 Mix columns:
0b1000110110101001
After Round 1 Add round key:
0b110011000001100
After Round 2 Substitute nibbles:
0b1000100010011100
After Round 2 Shift rows:
0b1000110010011000
After Round 2 Add round key:
0b110011100111101
The input plaintext is :0b1010100110111001
The key for decryption is 0b1110101110100101
The cipher text is 0b110011100111101
Connection to ('127.0.0.1', 48486) established
hvsblogger@hvsblogger:~/stuff/iit-dm/classes/SEM-2/1.crypto/project/AES/TEST$
```

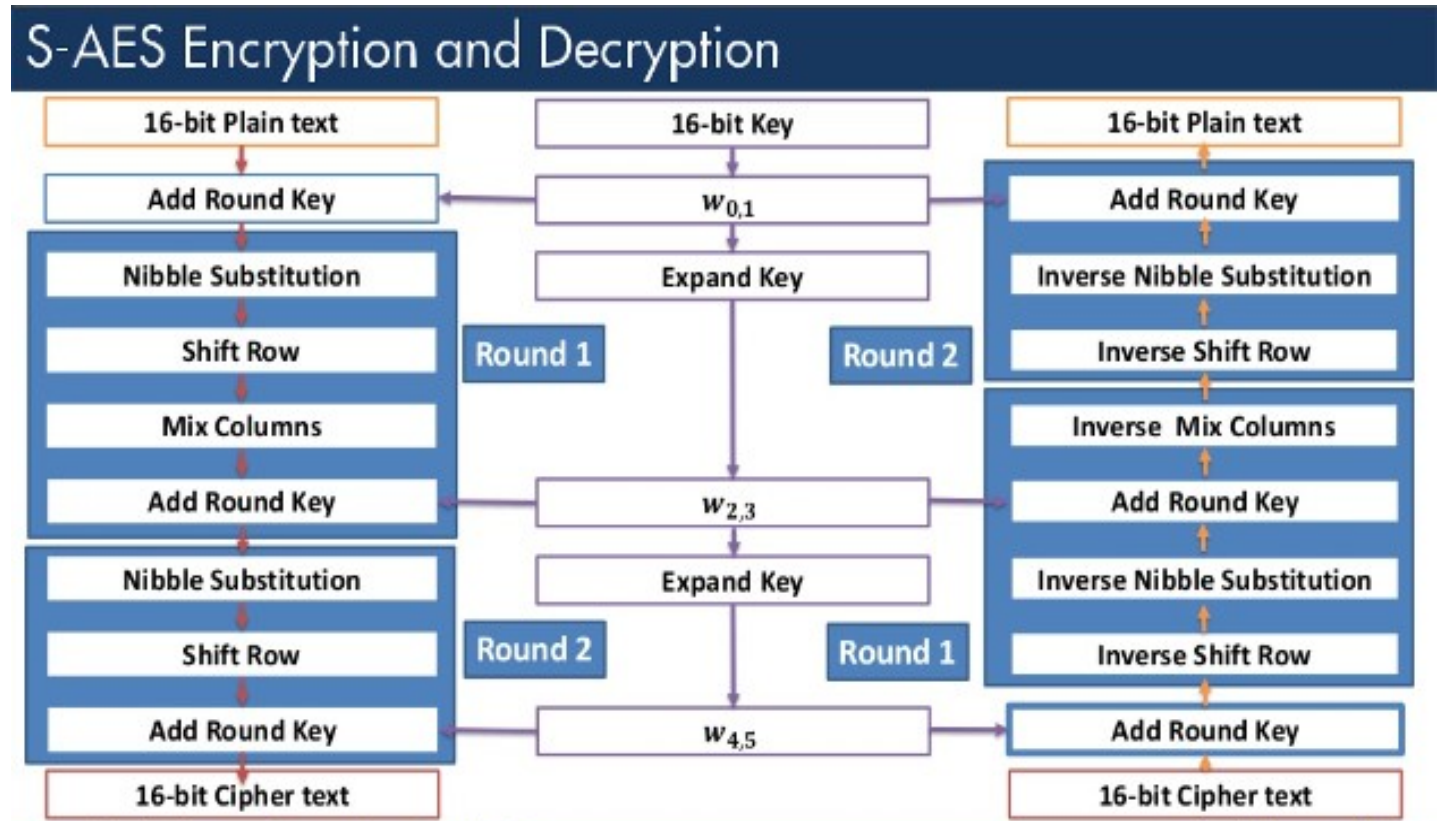
**Right Terminal (Client):**

```
hvsblogger@hvsblogger:~/stuff/iit-dm/classes/SEM-2/1.crypto/project/AES/TEST$ python3 wclient.py
The key for decryption is :
0b1110101110100101
After Pre-round transformation:
0b1000110010011000
After Round 1 InvShift rows:
0b1000100010011100
After Round 1 InvSubstitute nibbles:
0b110011000001100
After Round 1 InvAdd round key:
0b1000110110101001
After Round 1 InvMix columns:
0b1101110001001010
After Round 2 InvShift rows:
0b1101101001001100
After Round 2 InvSubstitute nibbles:
0b100001000011100
After Round 2 Add round key:
0b1010100110111001
The respective plaintext is 0b1010100110111001
hvsblogger@hvsblogger:~/stuff/iit-dm/classes/SEM-2/1.crypto/project/AES/TEST$
```

# Socket programming



# AES Implementation



16-bit block

16-bit key

4 x 4 S-box

Field 16f6

Modulus  $X^4 + X + 1$

2 rounds

**sbox** ; substitution box and substitution Inverse are initialized.

**sub\_word()** is defined to Substitute word Block with the help of sbox, substitute nibbles to another nibbles. Each nibble is replaced by nibble indexed by row (left 4-bits) & column (right 4-bits) of a 4x4 table

**gf\_mult(a, b):** Galois field multiplication of a and b in  $GF(2^4) / x^4 + x + 1$

a&b are respective numbers and will return the product of both a&b under  $GF(2^4)$

**int\_to\_state(n):** is converting an integer of 2B into a 4-element vector (state matrix) and it returns the state corresponding to the integer value

**state\_to\_int():** this is reverse of int\_to\_state(). It will convert the 4 element vector into a 2B int and return an integer corresponding to that state.

**add\_round\_key(s1, s2):** Add round keys in  $GF(2^4)$  ,(s1 -> First number),(s2 -> Second number) and it returns the Addition of both under  $GF(2^4)$

**sub\_nibbles(sbox, state):** block for Nibble substitution (sbox for Substitution box to use for transformation) ( State to perform sub nibbles transformation on , return the Resultant state

**shift\_rows(state):** Shift rows and inverse shift rows of state matrix (same) , State to perform shift rows transformation and returns Resultant state)

**Mix Columns(state):**Apply the matrix multiplication with the constant matrix, Me, using  $GF(24)$ . For  $GF(24)$ , the addition operation is simply an XOR, and for the multiplication operation you can use a lookup table.

**inverse\_mix\_columns(state):**Inverse mix columns transformation on state matrix, State to perform inverse mix columns transformation and Resultant the state

**encrypt(plaintext,key):** this function takes 2 arguments plaintext: the normal 16bit message and key 16bit. Then it calls the function one by one and encrypts the message.

**Decrypt(ciphertext,key):** this will takes 2 argument ciphertext ,key and returns the actual plaintext.