Question Number 4

- Consider that there are two rounds of AES
  - Using openssl generate a 128-bit message to initialize the AES state matrix.
  - Now modify any byte of this state to get another state.
  - Run two rounds of AES on these states
  - For rounds keys you can use output of Problem-3
  - After every operation take the XOR of the states
  - In the main assignment show the two states used. Then show how the XOR of the states propagates though each step of the round.
  - Verify if you can observe a distinguishing pattern in the output.
  - Write a code for the above.
- Also see how the pattern changes as you change the location of the modified byte for the second state.

Solution.

Observations

- **Output**

  To get output for observation of xor variation run the python file named as `AES_2Round.py` this will dump the output in termial. Ouput will be as follows

```
============================================================

location 1,0

Message 1 :  11f019ef716db40688b1dbdfcd381c29
Message 2 :  11f019ef716db45288b1dbdfcd381c29
Cipher  1 :  e2b231ae32648ddc15228f591ec22607
Cipher  2 :  e2b2311d3264d7dc150c8f59b9c22607

Xor Variation

Message                :  00000000ed00000000000000000000
Round Key Apply        :  00000000ed00000000000000000000
First Round Sub        :  00000000540000000000000000000000
First Round Shift      :  00000000000000540000000000000000
First Round Mix Col    :  000000fc000000a80000005400000054
First Round Key Apply  :  000000fc000000a80000005400000054
Second Round Sub       :  000000b30000005a0000002e000000a7
Second Round Shift     :  000000b300005a00002e0000a7000000
Second Round Key Apply :  000000b300005a00002e0000a7000000
CipherText             :  000000b300005a00002e0000a7000000

Cipher Diff
[(3, 'ae', '1d'), (6, '8d', 'd7'), (9, '22', '0c'), (12, '1e', 'b9')]

============================================================
```

- **Distinguishing Pattern**
  - In message pair only one byte is different, So only byte is not zero in xor diff.
    00000000ed00000000000000000000000
  - In Row Shift Operation Xor diff is shifted by some position

    diff before shifting 00000000005e00000000000000000000
    diff after shifting 000000005e00000000000000000000000
  - In Mix column diff increases in more positions, than previous one

    diff before mix column 000000005e00000000000000000000000
    diff after mix column e2000000bc0000005e0000005e000000
  - Diff in Operation like state xor with key and Substitution diff not changes it is same as previous diff

1. Verifying a Distinguishing Pattern in the Output

A distinguishing pattern is any observable behavior during the AES transformations (e.g., XOR differences) that allows one to differentiate AES from a truly random permutation. Based on the implementation and results, the following observations were made:

- The XOR of the ciphertexts consistently has exactly **4 non-zero bytes** for the given input difference.
- The propagation of the XOR difference through the AES transformations (`SubBytes`, `ShiftRows`, and `MixColumns`) shows deterministic behavior, reflecting the structured nature of AES.
- Unlike a truly random permutation, which would not produce consistent patterns, AES exhibits predictable propagation of input differences.

This deterministic propagation of XOR differences through the AES steps serves as the **distinguishing pattern**.

Example: The following example illustrates the distinguishing pattern:

- **Messages:**

  Message 1: `59d0b9515b009b6c69aa9654339a2d8b`
  Message 2: `59d0b9515b009b6c69aa9654339a2dff`

- **Ciphertexts:**

  Cipher 1: `992b44622099b43ee6539288017123b6`
  Cipher 2: `c32b44622099b409e653e788012223b6`

- **XOR of Ciphertexts:**

  Cipher XOR: `5a00000000000037000075000053000`

  The XOR difference shows exactly **4 non-zero bytes**, which is a consistent observation.

Propagation of XOR Difference: The following table illustrates how the XOR difference propagates through each AES step for the above input:

Step	XOR Difference
Message XOR	00000000000000000000000000000074
Round Key Apply	00000000000000000000000000000074
First Round `SubBytes`	000000000000000000000000000000b9
First Round `ShiftRows`	00000000000000000000000b9000000
First Round `MixColumns`	b9000000b9000000d000000069000000
First Round Key Apply	b9000000b9000000d000000069000000
Second Round `SubBytes`	5a000000370000007500000053000000
Second Round `ShiftRows`	5a000000000000370000750000530000
Second Round Key Apply	5a000000000000370000750000530000
CipherText	5a000000000000370000750000530000

Table 1: Propagation of XOR Difference Through Each AES Step

2. How the Pattern Changes with Modified Byte Location

When the location of the modified byte in the second input message is changed, the propagation of the XOR difference changes as follows:

- The positions of the non-zero XOR bytes in the ciphertext vary based on the modified byte's location in the input.
- During the `ShiftRows` step, the XOR difference is shifted to different positions based on the row-wise shifts.
- During the `MixColumns` step (first round only), the XOR difference spreads to multiple bytes within the same column, resulting in more non-zero bytes in the XOR.

Empirical Observations: For example, modifying the byte at position `(3, 3)` in the input message produced the following results:

- **Ciphertext XOR:**

  Cipher XOR: `5a000000000000370000750000530000`

  The XOR difference has **4 non-zero bytes** at fixed positions: `(0, 1)`, `(14, 15)`, `(20, 21)`, `(26, 27)`.
- When the modified byte is moved to position `(2, 2)`, the non-zero XOR bytes in the ciphertext appear at different positions but follow the same deterministic propagation pattern.