Nirbhay Sharma (B19CSE114)

Cryptography - Assignment - 4

que1

part-a

- encryption and decryption function has been implemented using python Crypto library and the above mentioned code is available in the file b19cse114-code-1.py

part-b

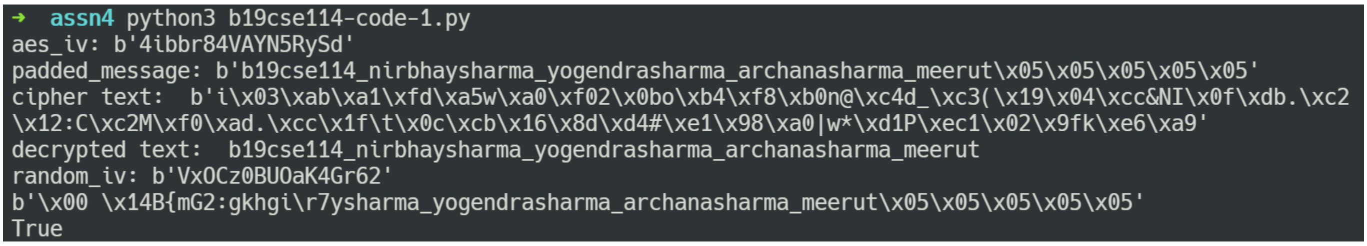
- oracle function has been implemented by first decrypting the cipher text and then checking the last b bits where b is the padding value required to make it a multiple of 16.

part-c

- output of padded plaintext is shown in figure

part-d

- output of encrypted cipher text is shown in figure



```
➜  assn4 python3 b19cse114-code-1.py
aes_iv: b'4ibbr84VAYN5RySd'
padded_message: b'b19cse114_nirbhaysharma_yogendrasharma_archanasharma_meerut\x05\x05\x05\x05\x05'
cipher text:  b'i\x03\xab\xa1\xfd\xa5w\xa0\xf02\x0bo\xb4\xf8\xb0n@\xc4d_\xc3(\x19\x04\xcc&NI\x0f\xdb.\xc2
\x12:C\xc2M\xf0\xad.\xcc\x1f\t\x0c\xcb\x16\x8d\xd4#\xe1\x98\xa0|w*\xd1P\xec1\x02\x9fk\xe6\xa9'
decrypted text:  b19cse114_nirbhaysharma_yogendrasharma_archanasharma_meerut
random_iv: b'VxOCz0BUOaK4Gr62'
b'\x00 \x14B{mG2:gkhgi\r7ysharma_yogendrasharma_archanasharma_meerut\x05\x05\x05\x05\x05'
True
```

que2

part-a

- output provided at figure

part-b

- output provided at figure

part-c

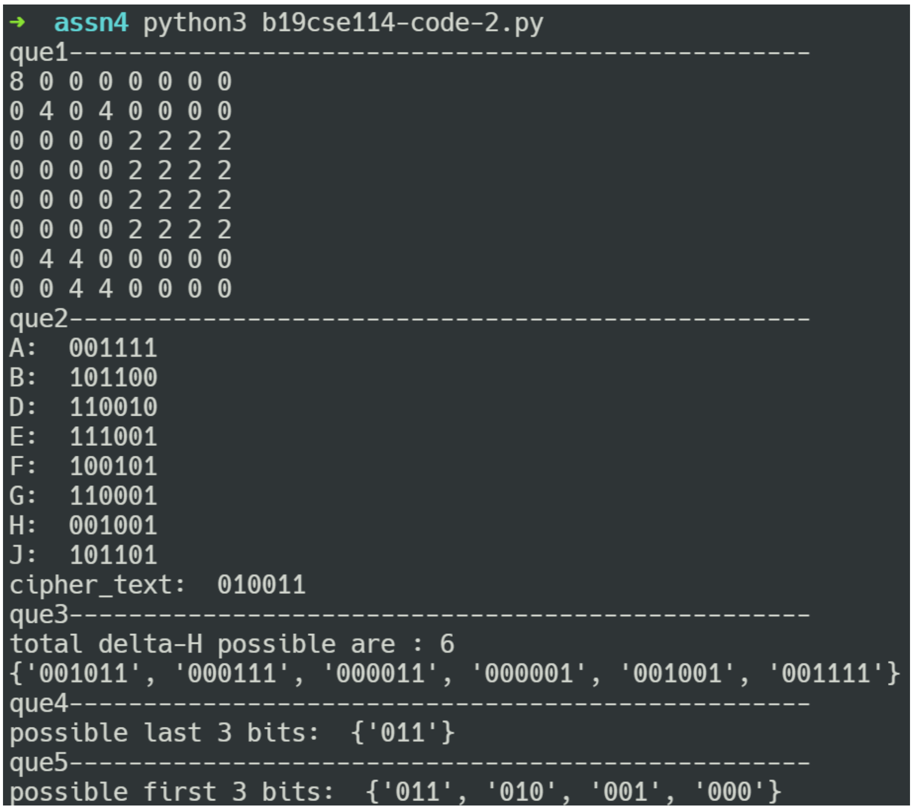
- brute force has been used to find δh first all the values of p and p' for which $p \oplus p' = 000001$ and there are 32 such pairs (actually there are 64 but it won't matter because in δh, order of p and p' won't matter)
- out of the those 32 pairs each of the pair is passed through SPN and δh is found using $h \oplus h'$ and stored in set and found to be 6 unique values of δh with detail output shown in figure

part-d

- attack algorithm has been developed as follows
    - observe that two pairs of plain text has the same property as in part-c i.e $p \oplus p' = 000001$
    - also observe that δh is invarient of keys (k_1, k_2, k_3, k_4) and hence we can exploit this property to develop an attack
    - algorithm is to go from backward in SPN network, we can observe that $j = c \oplus k_4$ so we can iterate through each of the possible key value (it is 8 because we are finding last 3 bit of key k_4), so iterate over all the key values and find corresponding $j's$ for both (pt_1, ct_1) and (pt_2, ct_2) - then pass these $j's$ last 3 bits to s-box in backward fashion and find h and h' and xor them resulting to δh
    - now utilizing property 2 that δh is invarient of keys so we can check that for which key value we get a valid δh value (valid δh value is whose last 3 bits are matching with the last 3 bits of possible values of δh as found in part-c)
    - iterate the above process for all the plain text and ciphertexts and take the intersection of the valid keys
    - output is shown at figure
    - last 3 bits of k_4 are 011

part-e

- repeat the algorithm in part-d with first 3 bits and find out all possible values of keys by intersection from all the plain text and ciphertext, we can see the output at figure, 4 potential keyvalues are possible for first 3 bits and notice that first bit is same for all the keys i.e 0 and hence we can find another bit of k_4 which is first bit, i.e. 0
- k_4 = 0__011



```
➜  assn4 python3 b19cse114-code-2.py
que1-----------------------------------------------
8 0 0 0 0 0 0 0
0 4 0 4 0 0 0 0
0 0 0 0 2 2 2 2
0 0 0 0 2 2 2 2
0 0 0 0 2 2 2 2
0 0 0 0 2 2 2 2
0 4 4 0 0 0 0 0
0 0 4 4 0 0 0 0
que2-----------------------------------------------
A:  001111
B:  101100
D:  110010
E:  111001
F:  100101
G:  110001
H:  001001
J:  101101
cipher_text:  010011
que3-----------------------------------------------
total delta-H possible are : 6
{'001011', '000111', '000011', '000001', '001001', '001111'}
que4-----------------------------------------------
possible last 3 bits:  {'011'}
que5-----------------------------------------------
possible first 3 bits:  {'011', '010', '001', '000'}
```