ASSIGNMENT – 3

Name: Kakumanu Vamsee Krishna
Roll no: 22111065

Group Details: Group No. 3
Members:
Kakumanu Vamsee Krishna (22111065)
Rumit Gore (22111409)
Sandula Lavanya (22111078)
Aditya Kankriya (22111072)
Pratik Patil (22111047)

Python Tool Version used: 3.9.12(Spyder-Anaconda)

Question 1:

DOM Attack: In this assignment, you will be provided with the power traces of AES. The power traces are stored in a CSV file, where each row indicates the power consumption of one AES execution. For every row, the first entry is plaintext, the second entry is ciphertext, and all the subsequent entries are power consumption values. Your task is to write a code for Difference of Mean Attack, and use that code on the given power traces to recover the 4th and 5th bytes of the secret key used in the AES execution. The demo code is for finding out the 0th byte of the secret key.

Note: You will be provided with 10 CSV files, you have to use one according to your
group number.
(e.g., if your group number is 3, then use HW power trace 3.csv)

Logic:

Here, we had to implement DOM attack, and use the power traces provided to recover 4th and 5th bytes of secret key used in the AES execution. We have example code for recovering 0th byte. So, we need to run the same code twice to get the 4th and 5th bytes.

Now, for the 0th byte recovery we shifted the 128-bit cipher text to 120 bits towards right in order to get the 0th byte in last 8 bits and hence, got the 0th byte recovered.

Now, for extending the same concept for 4th and 5th byte, secret key we have to get the first 4 and 5 bytes of data of cipher text in order to recover the intended byte.

So, to recover 4th byte first, we shift the 128-bit cipher text, (120-32) i.e., 88 bits towards right to get the first 4 bytes at the end. Now, we do 'logical and' operation on the last 8 bits of the shifted cipher text with all 8-bit 1's to eventually get the 4th byte recovered.

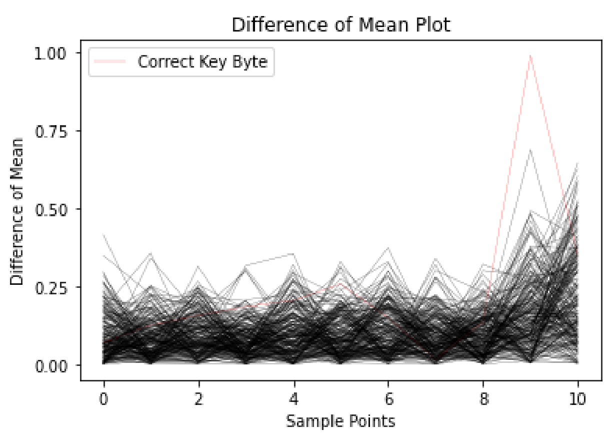
Similarly, to recover 5th byte of secret key, we shift the 128-bit cipher text, (120-40) i.e., 80 bits towards right to get the first 5 bytes at the end. Now, we do 'logical and' operation on the last 8 bits of the shifted cipher text with all 8-bit 1's to eventually get the 5th byte recovered.

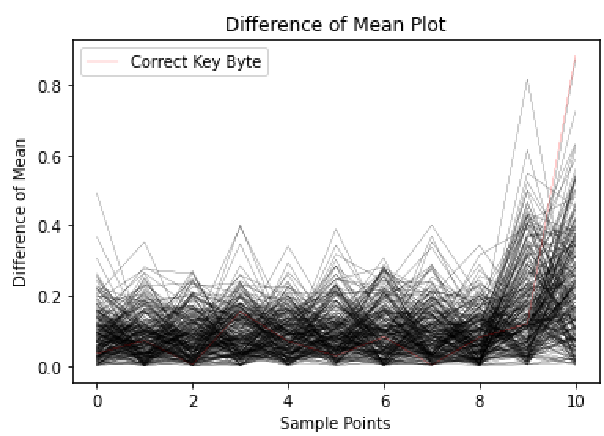
<u>Code implementation of same</u>:

```
rs = 120-40+(v-1)*8
ct_temp = ct >> (rs)& 0xFF

state_9th = InvSbox[ct_temp ^ kb]
binexp = '{0:08b}'.format(state_9th)
```

Where, '**v**' is the loop variable taking values **0 and 1**.
0 for the **4th** byte recovery and **1** for the **5th** byte recovery.
And '**rs**' is the right shift value.
88 for 4th byte and 80 for 5th byte





Key Bytes to recover: **4th and 5th**
CSV file used: **HW_power_trace_3.csv (Group 3)**
Recovered 4th byte secret key: **147**
Recovered 5th byte secret key: **227(this is correct)**

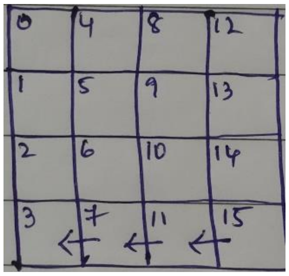
Question 2:

CPA: In this assignment, you will be provided with the power consumption of the last round of AES. The power traces are obtained from SAKURA-G platform that runs an implementation of AES-128 on a Spartan-6 FPGA. The power trace is stored in a CSV file, where each row indicates the power consumption of one AES execution.
For every row, the first entry is plaintext, the second entry is ciphertext, and all the subsequent entries are power consumption values. Your task is to write a code for Correlation Power Attack, and use that code on the given power trace to recover the target byte assigned to your group.

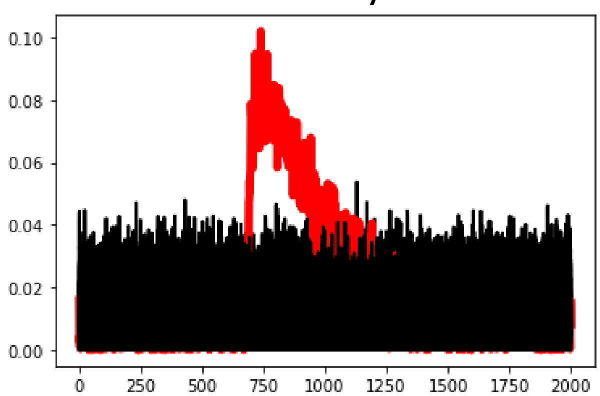
Logic:

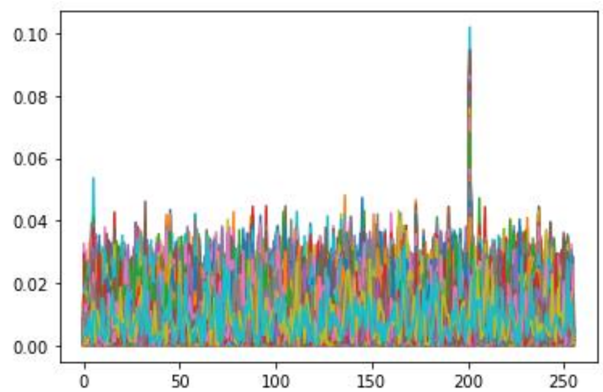
We have to attack the 3rd byte according to the diagram above. The matrix has all the 128 bits arranged byte wise in a 4x4 format.
The last round has the following stages: SubByte, ShiftRow and AddRoundKey. We are given the cipher text so first we take the 3rd byte from the ciphertexts[i][j] matrix and
XOR it with the corresponding key (k).

```
for j in range(0,8937):
    for k in range(0,256):
            x=ciphertexts[j][slice(6,8)] #slice of 3rd byte in CT10
            x8=int(x,16)^k
            cipher9[j][k]=InvSbox[x8] # 3rd byte of CT10 which is C15 in CT9,simulating shift row
            x1=ciphertexts[j][slice(30,32)] #slice of 15th byte in CT10 (which is 3rd byte in CT9)
            x2=int(x1,16) #C15 in CT10
            xor1=x2^int(cipher9[j][k])
            hamming_dist=number_of_ones(xor1)
            hypothetical_model[j][k]= hamming_dist
```



Now shifting this element to the right 3 times, that is to the 15th element of the ciphertexts matrix because of the 3rd byte being in the 3rd row within the matrix.
Then apply the InvSbox substitution corresponding to the 15th element to find the hamming distance by
counting the number of ones. The key which gives the highest peak(@201) for most of the test cases with respect to this value will be considered our key.





Target Byte assigned: **3rd Byte (Group 3)**
Value of recovered 3rd byte: **201**