Report on implementation of Correlation Power Attack

Group No: 8

Pranjal Kumar Srivastava Roll No: 22111046

Gunj Hundiwala Roll No: 22111024

Kapilkumar Kathiriya Roll No: 22111028

Amit Kumar Roll No: 22111008

Instructor : Prof. Urbi Chatterjee

Indian Institute of Technology ,Kanpur

Abstract

This report demonstrates the fundamental technique of Correlation Power Attack (CPA). The Correlation Power Attack (CPA) relies on targeting an intermediate computation, typically the input or output of an S-Box. The power model is subsequently

used to develop a hypothetical power trace of the device for a given input to the cipher. This hypothetical power values are then stored in a matrix for several inputs and can be indexed by the known value of the ciphertext or the guessed key byte. This matrix is denoted as H, the hypothetical power matrix. Along with this, the attacker also observes the actual power traces, and stores them in a matrix for several inputs. The actual power values can be indexed by the known value of the ciphertext and the time instance when the power value was observed. This matrix is denoted as T, the real power matrix. It may be observed that one of the columns of the matrix H corresponds to the actual key. In order to distinguish the key from the others, the attacker looks for similarity between the columns of the matrix H and those of the matrix T. The similarity is typically computed using the Pearson's Correlation coefficient.

Our Performance in Assignment:

Our Work in Assignment:

I. **Code Analysis**

As we are group 8, the target byte for our group was 10^{th} byte. The given code file was made to extract 0^{th} byte. We found 2 issues in the given code file:

1. The file was throwing an error on running as the size of 'traces' and 'hypothetical_model' were different. So while computing pearson's correlation coefficient, it was throwing an error. While debugging this issue, we realised that the size mismatch is coming because while reading 'traces' from input file 'AES_Power_Trace.csv', the first line is not being read as pandas considers the first line to be header. Hence, to correct this we changed the reading file code from:

   df = pd.read_csv (r'AES_Power_Trace.csv')

   To

   df = pd.read_csv (r'AES_Power_Trace.csv', header=None)

2. The 'number_of_ones' function given didnt't seem to be giving the correct value. The purpose of this function is to calculate the number of 1s in the binary representation of a number. While going through the code, when for testing I gave it integer 1 as input, we were expecting it to output 1. But it instead output 2.

```
[7]:
def number_of_ones(n):
    c = 0
    while n:
        c += n%2
        n /= 2
    return c
```

```
[8]:
number_of_ones(1)
```

```
[8]: 2.0
```

Figure: Function and Output

The reason of this behaviour is that python considers the variable ‘n’ as float. Hence, even the output returned is 2.0 and not simply 2. Therefore, if we used this function, it would have output the number of 1s as wrong and affected the CPA computation. Hence, we changed the logic in ‘number_of_ones’ function to:

```
[9]:
def number_of_ones(n):
    return bin(n).count("1")
```

```
[10]:
number_of_ones(1)
```

```
[10… 1
```

Figure: Function Logic Updated with Correct Output

II. Logic

As we are group 8, we were expected to fetch the 10th key byte.

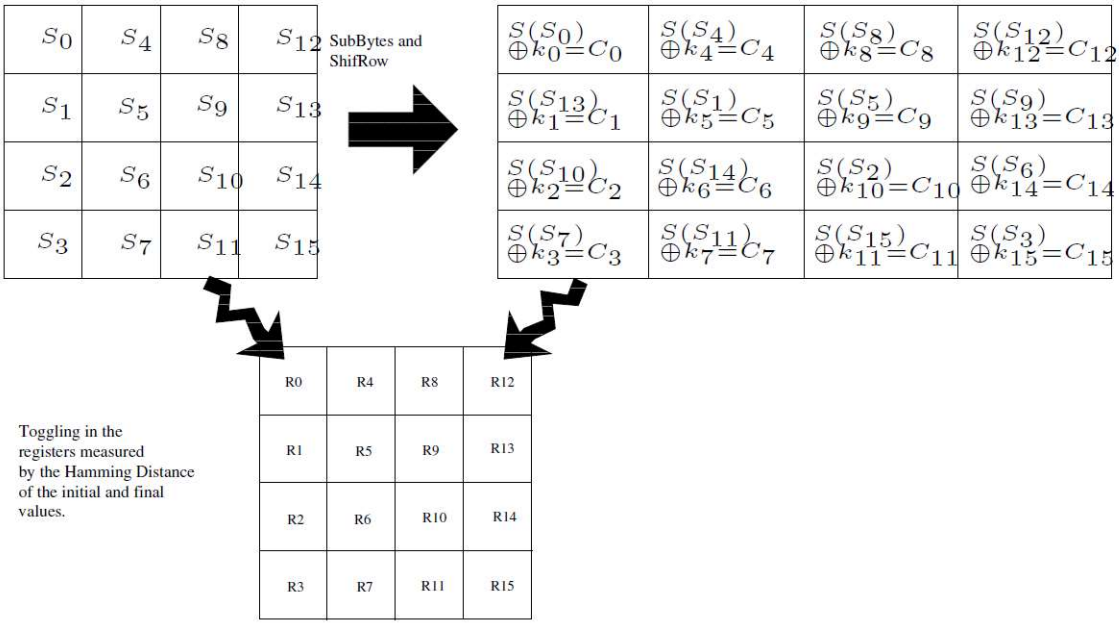

Figure: Computing Hamming Distance in the Registers due to Last Round Transformation of AES

Due to the Shift Row operations, the Hamming Distances need to be properly expressed.

For the above diagram, for a given Matrix S of dimensions 4 by 4, the Shift Row operations can be defined as:

S[0][1], S[1][1], S[2][1], S[3][1] = S[1][1], S[2][1], S[3][1], S[0][1]
S[0][2], S[1][2], **S[2][2]**, S[3][2] = **S[2][2]**, S[3][2], **S[0][2]**, S[1][2]
S[0][3], S[1][3], S[2][3], S[3][3] = S[3][3], S[0][3], S[1][3], S[2][3]

Similarly, the Inverse Shift Row transformation as given in the diagram can be defined for a given matrix S as:

S[0][1], S[1][1], S[2][1], S[3][1] = S[3][1], S[0][1], S[1][1], S[2][1]
S[0][2], S[1][2], **S[2][2]**, S[3][2] = **S[2][2]**, S[3][2], **S[0][2]**, S[1][2]
S[0][3], S[1][3], S[2][3], S[3][3] = S[1][3], S[2][3], S[3][3], S[0][3]

Hence, the 10th byte (S[2][2]), after Shift Rows operation, is getting assigned to the 2nd byte (S[0][2]). After Inverse Shift Rows, the Shift Rows operation will be reversed.

Therefore, computing Hamming Distance for the 10th Byte to get the Hypothetical Power Model can be expressed as:

Hamming_Distance(C2, InverseSBox(Key XOR C10))

where C2: 2nd Byte of the Ciphertext,
C10: 10th Byte of the Ciphertext
Key: Guessed Key

On running this logic, for given byte 10, we got the Key value **170** as answer. This was also evident from the output graph plot:

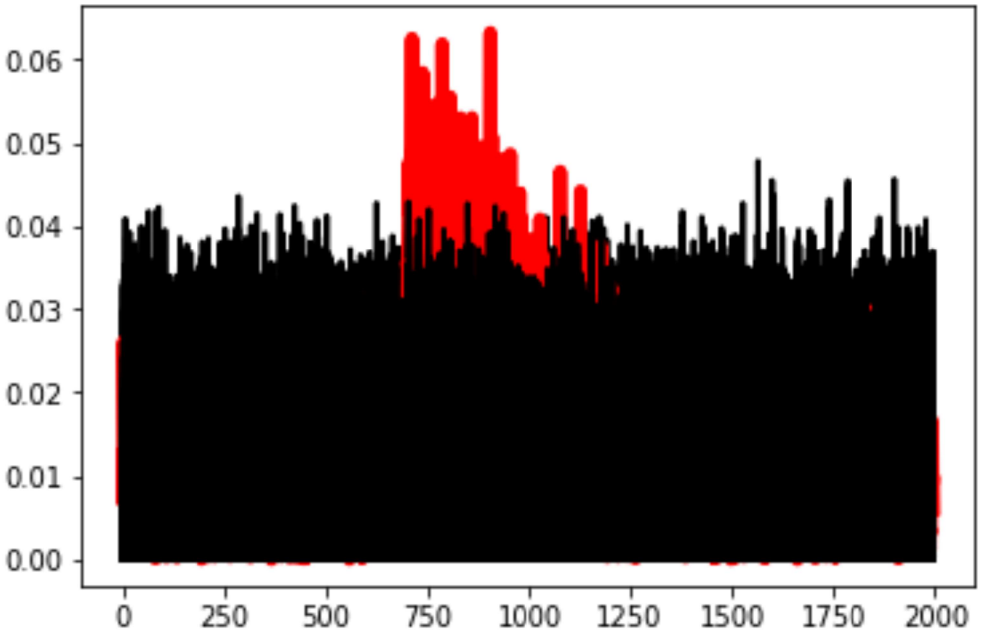

Figure: Graph Plot for CPA Output