ECEN 759 Lab 7: Differential Fault Analysis Attack on AES

S M Farabi Mahmud

May 6, 2021

1 Preliminaries on AES

Since we are using a AES with 128 bit key length, there are 10 rounds of operation in total. First 9 rounds consists of the following operations -

- Shift Rows **SR**
- Sub Byte **SB**
- Mix Column **MC**
- Add Round Key **RK**

However, in the last round, we do not have the **MC** operation. We express the key of each i-th round as K^i The plaintext and the ciphertext is denoted by P and C respectively. M^i_j denotes the j-th byte at i-th round.

2 Reverse Key Scheduling

We know the key scheduling algorithm uses specific algorithm to generate all the round keys from the master key. We use the exact inverse order to generate the round keys from the later round keys. For this purpose, we use the following code -

```
def prev_roundkey(arr, round_constant):
  prev_arr = []
  w4 = [arr[0],arr[1],arr[2],arr[3]]
  w5 = [arr[4],arr[5],arr[6],arr[7]]
  w6 = [arr[8],arr[9],arr[10],arr[11]]
  w7 = [arr[12],arr[13],arr[14],arr[15]]

  w3 = [x^y for x,y in zip(w6,w7)]
  gw3 = gfunction(w3, round_constant)
  w2 = [x^y for x,y in zip(w5,w6)]
  w1 = [x^y for x,y in zip(w4,w5)]
  w0 = [x^y for x,y in zip(w4,gw3)]
  prev_arr.extend(w0)
  prev_arr.extend(w1)
  prev_arr.extend(w2)
  prev_arr.extend(w3)

  return prev_arr
```

3 Single Bit Fault Attack

In this section, we will describe how we have implemented the single bit fault attack on AES. We used the concept from Giraud's work [1]. In the first attack, we target the one bit of the input of the Round 10 that is denoted by M^9.

```
def dfa_bit_fault(c, ds):
  locations = [0x1, 0x2, 0x4, 0x8, 0x10, 0x20, 0x40, 0x80]
  recovered = []
  for j in range(16):
    count = [0]*255
    for x in range(255):
        for e in locations:
            for d in ds:
                lhs = c[0][shift_row_index(j)] ^ d[shift_row_index(j)]
                rhs = subbyte(x) ^ subbyte(x ^ e)
                if (lhs == rhs):
                    count[x] = count[x]+1
    recovered.append(np.argmax(count))
  return recovered
```

Listing 1: Main loop for the DFA Single Bit Fault Attack

For this reason, we can directly determine that the output of Round 10, the ciphertext C that follows the equation -

$$C_{SR(j)} = SB(M_j^9) \oplus K_{SR(j)}^{10} \tag{1}$$

Here, in this equation 1, we can get the output of the round 10, the ciphertext C from the input of the Round 10 M^9 and the XORing that with the round key K^{10}. If we apply a fault e_j on one bit of the j-th byte of the M^9 we can get the faulty ciphertext D following the equation 2

$$D_{SR(j)} = SB(M_j^9 \oplus e_j) \oplus K_{SR(j)}^{10} \tag{2}$$

XORing equation 1 and equation 2 we can get -

$$C_{SR(j)} \oplus D_{SR(j)} = SB(M_j^9) \oplus SB(M_j^9 \oplus e_j) \tag{3}$$

We have only one bit fault, so possible location $e_j \in \{0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80\}$. Again, one bit fault at M^9 impacts only one byte of the faulty ciphertext D, we can generate all possible values for the M^9 in range $\{0, 1, \ldots, 255\}$ and all possible bit locations and see if the equation 3 holds for these values. Then we determine the possible value of the recovered key from the maximum of the count.

4 Byte Fault Attack

References

[1] Giraud, C. Dfa on aes. In *International Conference on Advanced Encryption Standard* (2004), Springer, pp. 27–41.