Lab 3 Handout

*Differential Fault Analysis Attack on AES*

# Introduction

Electronic devices are susceptible to faults, whether they have malicious intensions or occur due to environmental conditions. Their effects vary from alterations to the intended execution (soft errors) to catastrophic results.

Non-tolerance to soft errors can be advantageous for cryptanalysis. Since AES is based on the principle of *confusion* *and* *diffusion*, a small alteration in the secret key or the plaintext would result in a totally different ciphertext. By inducing faults in a cryptographic mechanism such as AES, one can examine the differences that occur between the normal and the faulty execution of the cryptography and derive information regarding the secret key.

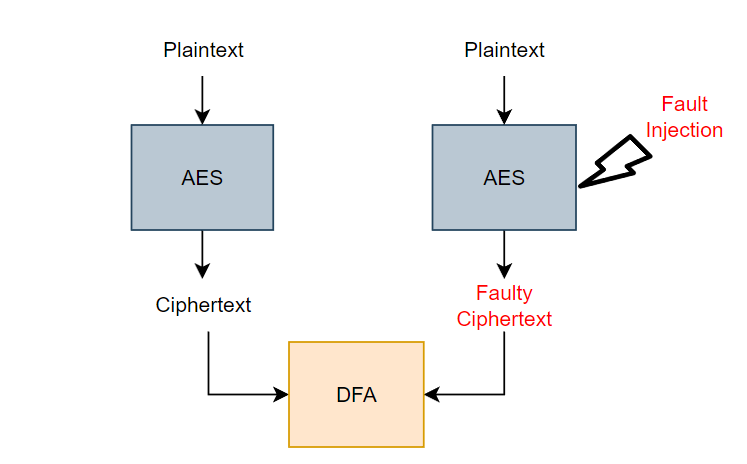


Figure 1. DFA illustration

Differential Fault Analysis (DFA) attacks examine the differences between the correct ciphertext and a faulty ciphertext. Since the attacker can hypothesize the effect of the attack (by recreating the values and the faults), it is easy to retrieve the original intermediate state and its corresponding round key.

Hence, it is important to know at which point during the encryption execution did the fault occurred. It should be noted that real fault injections today are of high prediction, so this is plausible. For our purpose, we will use a simulated fault injection – the code used generates a single-bit flip.

We opt to attack the input of the 10th round of AES encryption, so to derive one 10th round key byte. The whole 10th round key is enough to reveal the secret key used for AES-128. Fig.2 demonstrates the value substitutions and permutation performed at that stage. Note that, for the first byte, only SubBytes and AddRoundKey have an effect as ShiftRows does not alter the first row of the AES stat and MixColumns is skipped during the last AES round.

# DFA attack

Chart, box and whisker chart

Description automatically generated

Figure 2. Value propagation for 10th round of AES

Fill out the given code, given the following instructions:

## Part 1 : Attack Simulation

Task 1 : Create the fault injection functionality

Go to fault\_injection.m, inside AES folder (contains the AES operations) .

The function will take bytes\_in, which is the 16-byte AES state as an argument.

Additionally, it will take byte\_to\_attack – the position of the state byte under attack and

faulty\_bit, the position of the bit that will be flipped to simulate the fault injection.

Create mask\_bit, a value that, if XORed with another, causes a bit flip, using the faulty\_bit.

Apply mask\_bit over the input state’s byte under attack.

Return the resulting state (bytes\_out).

Task 2 : Add the fault injection functionality

Go to cipher.m, inside AES folder.

Locate the start of round 10 and add the following line of code:

state = fault\_injection(state,byte\_to\_attack,faulty\_bit);

Task 3 : Setup the analysis parameters

Go to DFA\_attack.m .

Set the byte\_to\_attack value.

Set the bit\_to\_attack vector with 3 values, from 1 to 7.

Optionally, Use randi to generate a random vector of 3 values, where each value is used once.

Task 4 : Perform the encryption

Perform four encryptions, one correct and three faulty, using the parameters you previously set.

Examine the effect of fault injection on the resulting ciphertexts.

## Part 2 : Differential Fault Analysis

Task 8 : Derive the 10th round key byte under attack

Verify that the solution corresponds to the input of the 10th round

Task 7 : Examine solution sets for recurring values

Create the conditions so that, if a value appearing in solution\_1 does appear in solution\_2 and solution\_3 (common value in all three) puts value in the final solution vector.

Task 6 : Create the fault hypotheses

Create a hypothesized attack by recreating the fault injection attack.

For each bit (r), create a mask\_bit (e) similar to Task 1.

Then for each possible byte value (x):

Produce the correct sbox value : SubBytes(x)

Produce the faulty sbox value : SubBytes(faulty\_x). Determine the faulty\_x by using e.

Extract the difference of SubBytes(x), SubBytes(faulty\_x)

If the hypotheses match any of the actual differences, append x to the corresponding solution vector (solution\_1, solution\_2 or solution\_3)

Task 5 : Calculate the differences of the correct ciphertext and each faulty ciphertext

Fill out the question marks so that every dif\_ contains the difference between the correct ciphertext and faulty ciphertext’s byte under attack.

# Deliverables

The deliverable of the this lab will be **a report**. Describe the methodology followed in the lab. In addition, answer the following questions regarding the attack:

1. Find bit\_to\_attack combinations that produce result in a successful DFA attack.
2. If the DFA code remains the same, what other 10th round bytes can you extract?
3. In order successfully to extract all 10th round key bytes, what changes should you perform?
4. Compare side-channel leakage analysis and differential fault analysis. What are the requirements of performing each attack, the complexity of each analysis and the results?
5. Describe a countermeasure to thwart a fault-injection based attack.