Lab 2: Differential Fault Analysis of AES

Due Date: Jan. 29th (T) in class

1 Overview

In this assignment, you are asked to implement differential fault analysis (DFA) attack of one-block AES encryption.

2 Details

Implement a differential fault analysis attack on AES encryption in software. Please use this standard S-box based AES software implementation, https://github.com/kokke/tiny-AES-c, rather than the OpenSSL implementation based on T-table. The reason is you need dictinct four operation steps (include MixColumns) for each round to inject fault, instead of combined T-table lookup operations (three steps combined). Study the code carefully and understand the AES algorithm and this implementation' data structure in great details. For example, the state is organized in two-dimensional array - row and collumn, but identify which array index is used for row and which is used for column.

You have to modify the AES files to simulate fault injection in a seleted intermediate state defined by the fault model. From the fault injection simulation, you obtain the faculty ciphertext (F) for the same plaintext input. Then you need to write your analysis code to take the correct ciphertex (C) and the F as input to retrieve the AES encryption key. Note that you need to copy the MixColumns function from aes.c to your analysis code to simulate the fault propagation through MixColumns to generate differential output.

Fault model: The fault type is random fault, and the location is a known byte position (i.e., the row and column indices are both known) of the input state of 9th round MixColumns operation. Your goal is to recover the corresponding 4-byte subkey. One {C,F} pair may not give you the unique result (subkey value and fault value), and you need to produce multiple {C,F} pairs by changing the plaintext input while keeping the same fault injection (the fault Δ is the same). Analyze how many encryption output pairs are needed to find the entire 4-byte subkey, and how the key candidate space is screened with more encryption pairs analyzed. Print this statistics as your analysis code runs.

3 What to Submit

For the lab submission, you will need to turn in all your code, which should be testable by us using Makefile commands. Include a Makefile and a README file for how to compile and run your programs. Use the following command to zip your submission. There is no restriction on the programming languages, but state in the README file which language you used for this lab.

```
$ zip [your last name]-lab2.zip codes
```

4 Extra Credit

Bonus point:

1. If you relax your fault model to only knowing the column the one-byte fault falls onto (without knowing the row) and successfully retrieve the 4-byte subkey, there is 10-point bonus.
2. If you recover the entire 16-byte key, there is 10-point bonus (you can still use the original strict fault model - known fault byte position).