Systems and Network Security Laboratory

Homework 1: AES trace simulator

Deadline: 2019/5/13 13:59

jiunpeng@ntu.edu.tw

1. Problem Statement

Performing a side-channel analysis can be quite challenging. Setting up a proper measurement environment and capturing the leakage information could be even harder for beginners. What if we don't have the measurement equipment? How do we evaluate the leakages of a design if we only have the source code? A trace simulator could be beneficial in this situation to both the cryptographic designers and hackers. To a designer, a trace simulator can help gain more insight of the implementation, and test the effectiveness of different countermeasures. The trace simulator can also help hackers to verify their side-channel analysis program and test different attack methods. In this assignment, please implement an AES-128 encryption program that can:

1) Encrypt the plaintexts in the given plaintext file. (.csv format)
2) Generate the simulated leakage traces for each encryption.

The encryption **key is the MD5 digest of your student ID**. For example:

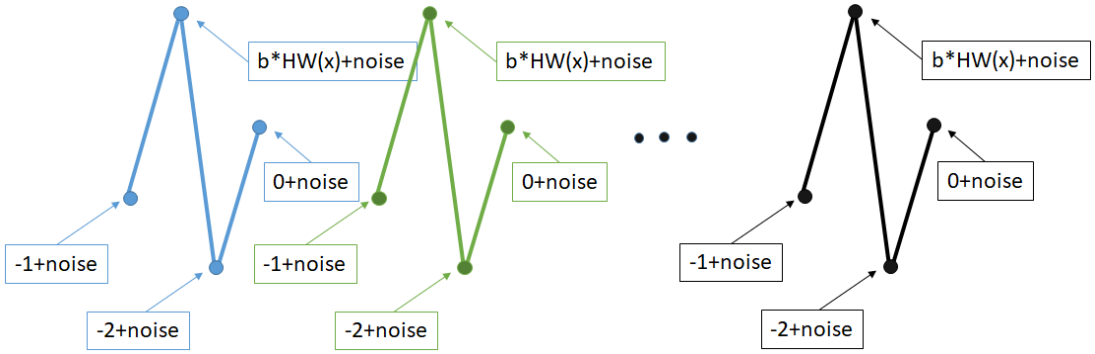
Student ID: r06921000

Key: `3220db6534d687f844c41b6de5a4c737` (128-bit)

```
$ echo -n r06921000 | md5sum
```

2. Trace simulation

Suppose we want to simulate the leakage of an 8-bit microprocessor running an AES-128 encryption program. It is reasonable to assume that we can capture the leakage information of the outputs of each subroutines in AES: SubBytes, ShiftRows, MixColumns, and AddRoundKey. To simulate a power trace, please generate 4 samples for each intermediate value, and only let the 2nd sample contain the leakage information, as shown in the figure below.



The leakage model is given as follows:

$$Leakage(x) = b * HammingWeight(x) + GaussianNoise(0, noise)$$

where *b* and *noise* are the user-defined parameters that can alter the signal-to-noise ratio (SNR) of the simulated traces. For example, let (b, noise) = (1, 0) we can get a perfect leakage trace where the SNR is infinite.

Please note that each trace should contain 2560 samples:

4 samples * 16 bytes * 40 operations.

For simplicity, after each of the 40 operations is done, append the 64 points to the trace in byte order (byte 0 to byte 15) regardless of the actual computation order of each byte.

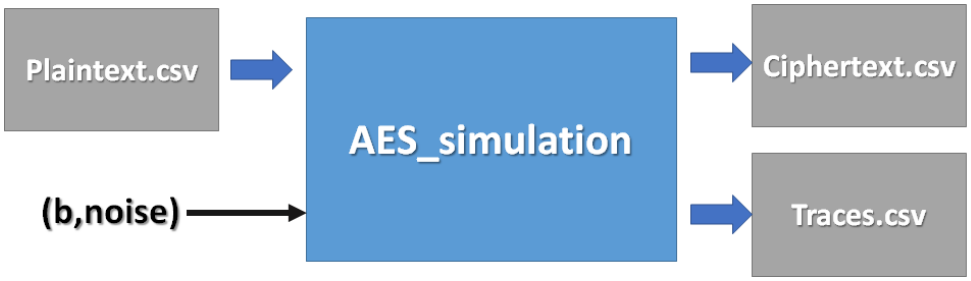
3. Input/Output Specification

The program should take in a Plaintext.csv file with the optional parameters *b* and *noise* for user to adjust the signal-to-noise ratio of the simulated trace. **Please set the default value of *b* to 1.0, and *noise* to 0.0.**

After the simulation, the program should generate 2 output files:

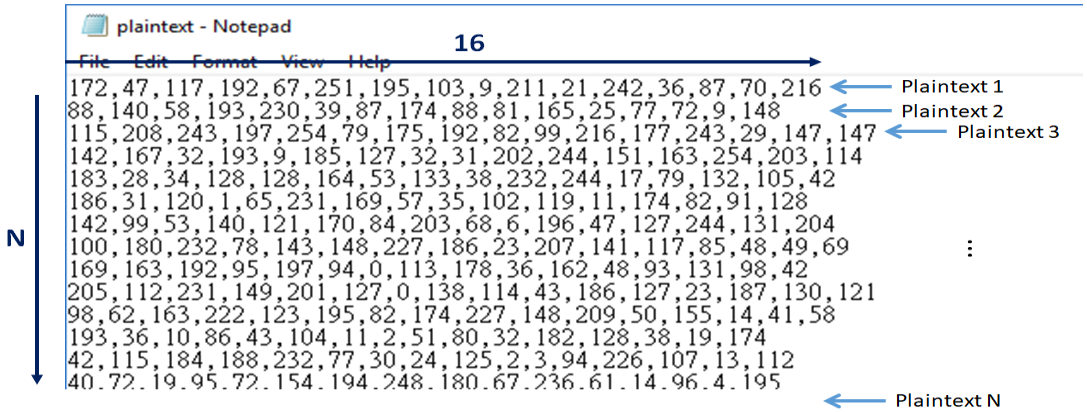
1) Ciphertext.csv: the encrypted data of the Plaintext.csv file.

2) Traces.csv: the simulated traces of the encryption process.



Plaintext/Ciphertext format:

The Plaintext and the Ciphertext are both CSV files. There are 16 numbers in each line, representing the 16 bytes (128 bits) of the input plaintext. Therefore, all the numbers are between 0 and 255. The total number of plaintext depends on the test case.



Trace file format:

The trace file is also a CSV file. As the description above, each trace consists of

2560 floating point numbers. The total number of the traces should be the same as the number of input plaintexts.

4. Command-line Parameters

In order to test your program, you are asked to add the following command-line parameters to your program:

[executable file name] [Plaintextfile.csv] [b] [noise]

Where b and noise are optional. For example:

```
./aes_sim Plaintext.csv 1.0 0.1
```

5. Language

Language: C/C++, or python

6. Submission

Please submit a .zip file named **HW1_${StudentID}.zip** (e.g. HW1_r06921000.zip) including the following materials:

1) source code
2) a text README file named **README.txt**
3) a report named **report.pdf**

```
HW1_r06921000
|-- README.txt
|-- report.pdf
|-- src/
```

README file:

Please provide the following information in your README file:

1. Student ID
2. Key
3. Compiler / Interpreter version
4. How to build the program
5. How to run the program

Report:

1. If you are using an open source AES implementation, please state your source.
2. Please plot the simulated traces.

Late submission penalty: 20% per day.