Assignment 1
Differential Power Analysis (DPA) Attack on AES Algorithm

Description:

In this assignment, you will be launching a differential power analysis attack on the AES algorithm. You will focus on the first round of AES. Your objective is to extract the key for AES by analyzing the power traces provided to you. You will use python for this assignment.

For assignment 1, you are expected to:

i. Launch the DPA attack on a single byte of the AES algorithm. You will plot the "difference of means" trace for each key guess and inspect these plots to find which one corresponds to the secret key.

ii. Extend the DPA attack to extract all 16 bytes of the key. Instead of using visual inspection, you will update your code to extract the key from the "difference of means" trace.

iii. Launch the DPA attack for different number of power traces (20, 50, 100, and 200).

iv. Report the key recovered in each case along with the "difference of means" plot and the percentage accuracy.

Resources required:

1. **Understanding the algorithm:**
   a. Class Notes.
   b. AES documentation: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf (Uploaded in Canvas along with the manual)
   c. Additional materials:
      i. https://perso.uclouvain.be/fstandae/PUBLIS/34.pdf
2. **Power traces database:** (Provided as 'aes_power_data.mat' on Canvas).
   a. The database consists of 200 plaintext messages, the corresponding ciphertexts, and the power traces. Note that each plaintext as well ciphertext and consist of 16 bytes. Thus, the size of the plaintext matrix is 200x16. This database is for the 16 bytes key: "00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF"
   b. Each power trace has 40000 samples. Thus, the size of the traces is 200x40000. We also provide an sbox along with the database.
3. **Accessing Python:** It is expected that the students already understand how to install python and run

4. Python: You will use **Python** to implement the DPA attack.

b. Some useful python functions:
    a. **plot** plots a 2-D line of given data.
    b. **np.bitwise_xor**(A,B) returns the bit-wise XOR of A and B.
    c. **bitget (A,bit_index)** returns the bit value (0 or 1) at position bit_integer in array A.
    d. **sbox[**A**]** returns the sbox lookup of A, Hint: if A is an array then the result will also be an array.

Task 1: DPA attack on one byte of AES key

In this task, you will launch the DPA attack on the 16th byte of the AES algorithm to recover the secret key. **Hint:** In python indexing begins at 0 so to attack the 16th bit you need index 15...
You will use plaintext, ciphertext, and traces provided in the database. We assume that the power is correlated to the LSB of the output of Sbox in the first round. The power traces in the database represent the power consumption for the first round of AES.

The DPA attack steps are specified as follows:

a. Specify the number of power traces to use, which is 200.
b. Specify the byte to attack, which is 16 for this task.
c. For each key guess:
    a. Predict the output of AES Sbox for each guess. Note that there are 256 (0-255) possible key guess values. You need to XOR plaintext with key guess before feeding to the Sbox.
    b. For the DPA attack, we can focus on one-bit of the predicted output. We suggest focusing on the LSB (index=1) of the predicted Sbox output. Hint: you can use bitget function in Matlab.
    c. Based on the value of predicted Sbox output, classify the power traces into two groups/bins. If the predicted Sbox output is 1, the power traces are added to group-0, else it is added to group-1.
    d. Find the "difference of means (DoM)" of the two groups, i.e., sum the power traces in each bin and divide by the number of the elements in the bin. Note that the sum will be a vector of size 40000x1. Then subtract the two vectors to obtain the DoM trace.
    e. Plot the traces to observe distinct peaks. Recall that the trace for the correct key shows distinct peaks. The other traces appear to be almost random.
d. Use the script provided in the python file to plot the DoM traces for different keys in either 4x4 plots or 8x8 plots. Mark the plot for the correct key and include in the write-up. Note that for 256 key guess and 40,000 samples in each trace, the size of DoM trace matrix will be 256x40000.

Task 2: DPA attack to recover all 16 bytes of the AES key

a. In this task, you will extend the DPA attack to recover all 16 bytes (128 bits) of AES key. You will not perform visual inspection anymore. Instead, you can assume that the absolute of the maximum value of DoM trace indicates the correct key.
b. You will write a python code that detects the peak in the DoM trace and finds the key that generated this peak. Start from the first byte and launch the attack on all 16 bytes. Report all 16 bytes of the key recovered by the DPA attack as a 16x1 vector (in the hexadecimal format, e.g., [01 02, …, 0F]).
c. Calculate the percentage accuracy between the original key and the key reported by the attack: $\frac{Number\ of\ bits\ recovered\ correctly}{128} \times 100$

Task 3: DPA attack with different number of traces

a. So far, you have used all 200 traces provided in the database. In this task, you will launch the DPA attack for different number of power traces and report the accuracy.
b. Repeat the attack in Task 2 for the following number of power traces: 20, 50, 100, and 200. Report the 16 bytes of key recovered in each case in a Table.
c. Inspect the percentage accuracy and observe the trend with the increasing number of traces and report your observations in the write-up.

Due date and Deliverables:

The due date to submit the codes and report is listed on Canvas. The following deliverables have to be submitted in a zip file format with **assignment1_<FirstName_LastName>.zip** as the file name in the Canvas.

1. python code.
2. Report containing the description of your code and the results for Tasks 1 – 3.
3. For Task 1, report at least 16 DoM traces (in a 4x4 plot) and mark the correct key. Observe if there are any other traces with distinct peaks. Report those traces along and mark the key they represent.
4. For Task 2, report all 16-bytes of the key retrieved. For key byte 1 and 16, also report the absolute DoM trace showing the maximum value of trace for each key guess.
5. For Task 3, provide a table, and/or graph detailing the the accuracy vs number of traces.
6. **Please ensure that a timestamp and your NetID are included in all the tool generated log files.**

Assignment 1 Rubrics:

1. Task 1 – 5 points
2. Task 2 – 2.5 points
3. Task 3 – 2.5 points