Physical Attacks on Secure Systems (NWI-IMC068)

Tutorial 2: DPA attack on an unprotected software implementation of AES

Goals: After successfully completing these exercises, you should be able to perform a Differential Power Analysis (DPA) attack and extract the key from an unprotected software implementation of a cryptographic algorithm. You will have the option to choose a suitable leakage model and a side-channel distinguisher.

Before you start: To complete this tutorial, your teacher will provide you with a package that contains the ChipWhisperer-lite board.

Instructions: Download the Jupyter Notebook for this tutorial from Brightspace. Follow the instructions in the notebook to complete the code and answer the questions in this document. The section names in the notebook correspond to the section names in this document.

1 Collecting Traces with ChipWhisperer

For this tutorial, we will collect 2 500 traces of TinyAES with ChipWhisperer. Make sure that each trace has 4 000 samples. Save the collected traces, the key used, and the plaintexts in a file. Refer to Tutorial 1 for instructions. Now disconnect from the ChipWhisperer, restart your Jupyter Notebook, and follow the instructions.

If you cannot collect traces with ChipWhisperer for some reason, or you forgot to save the collected traces and you want to finish the tutorial at home, you can download some sample traces from Brightspace.

2 DPA attack with 1-bit leakage model and DoM as distinguisher

The sensitive variable we use to attack the software implementation of AES is the S-Box output; see Fig 1. We attack the TinyAES-128 implementation.

2.1 Questions before starting the attack

The AES S-box is defined in the provided Jupyter notebook file.

Q1. How many bytes does our AES-128 key have?

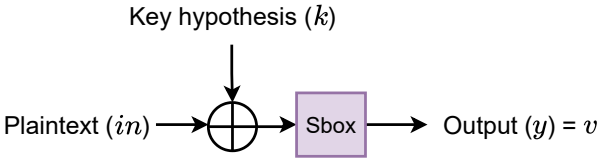

Figure 1: TinyAES Sbox

Q2. Can we use the divide-and-conquer strategy to attack the AES cipher? Explain your answer.

Q3. How many bits/bytes can we attack at once? Explain your answer.

Q4. How many possible `keybyte_hypotheses` are there?

Now that you have answered all the above questions, implement the components in the following order to perform the first attack.

1. Define a function called `intermediate_variable` that gets a `plaintext` and a `key_hypothesis` and returns the output of Sbox.
2. Set the variable `nr_keybyte_guesses` to its correct value.
3. Each group will attack the implementation using different bits. Select the target bit by computing the sum of your two student numbers modulo 8. Name the bit `specific_bit`.
4. Let's attack the first byte of the key (`target_byte=0`). Complete the code snippet in the notebook to recover the first byte of the key. Follow the instructions in the notebook to visualize your results.
5. Repeat the attack for all key bytes.

Answer the following questions.

Q5. Save the recovered bytes in an array. Print the correct key and the recovered key and see if they match the key chosen during trace collection.

Q6. For this attack we have used `single bit value` leakage model and the distinguisher was `Difference of Means (DoM)`. Would this attack be sufficient if we had more noise in the system? Why?

3 DPA attack with two different leakage models and CPA as distinguisher

In this assignment, we are going to do a DPA attack with an 1-bit leakage model and a 2-bit leakage model, using correlation as distinguisher. Before starting the attack in the notebook, answer the following question.

Q1. What is the value-prediction matrix? How many rows and columns does it have?

Now, we are ready to complete the following steps.

1. We want to compute the value-prediction matrix. Define a matrix and name it `prediction_matrix`. Initiate it with zeros (use `dtype=int` while setting the matrix elements to zero). Then, fill the elements with what your `intermediate_value` function returns for the corresponding trace and `key_hypothesis` pairs.

2. Define another matrix with the same shape as the value-prediction matrix. This matrix is called `labels`. Fill the matrix with the corresponding labels. Labels are defined by the chosen leakage model. First, we use the least-significant bit as the leakage model.

3. To complete the attack, correlate the labels from the leakage model with the power traces for `target_byte = 2`. Use the instructions in the notebook.

4. Plot the correlation for each key guess. Can you see from the plots what the correct key would have been? At what timesample does our sensitive variable occur?

5. Do the attack again, but now with a leakage model for 2 bits: the second least significant bit and the least significant bit. Follow the instructions in the notebook.

6. Plot the correlation for the correct key guess for both the 1-bit and the 2-bit models.

Answer the following questions.

Q2. What is the correct key guess for our target byte?

Q3. Which of the two leakage models has stronger correlation? Why do you think this is the case?