Hardware Security: Side-Channel Analysis - Assignment 1

Niels van den Hork (s4572602)
Niels van Drueten (s4496604)

April 1, 2019

1 Introduction

This report discusses the first assignment of the Side-Channel Analysis of the Hardware Security course. We have implemented the Correlation Power Analysis (CPA) against the cryptographic implementation of the PRESENT cipher.

2 Code explanation

Our implementation is written in Python. The first thing our code does is to load the input variables file in Python. To load the input variables file, which is a Matlab file (.mat), we have used the `scipy.io` Python library. This library contains a `loadmat()` function that handles this.

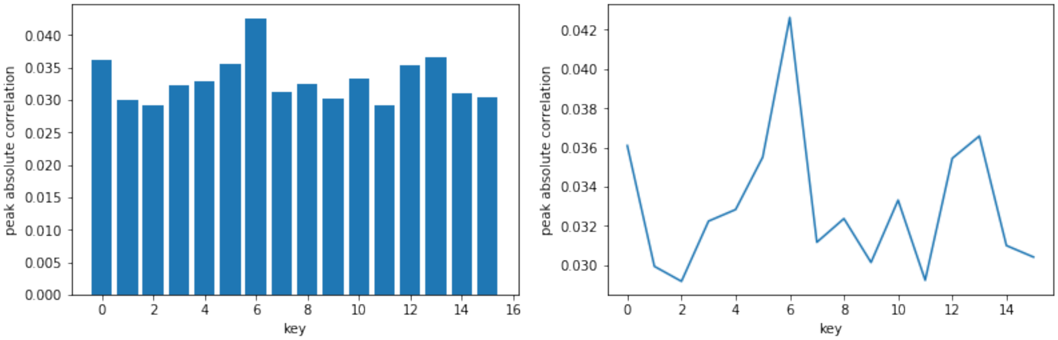
Then, we compute the value-prediction matrix. This matrix has a size of [no_inputs x no_keys]. This matrix predicts the value of the PRESENT cipher for each input variable and key combination. The input variable file contains 14900 4-bit inputs. The key bit length is 4, which results in $16(2^4)$ possible keys. This makes the value-prediction matrix a 14900 x 16 matrix. The PRESENT cipher uses a bitwise XOR of an input variable and a key part. The output of the XOR is directed to an SBox. The output of the Sbox depends on a known non-constant data value (the input variable) and a small part of the key (the key part).

The second matrix we compute is the power-prediction matrix with size [no_inputs x no_keys]. This matrix stores the hamming weight of every combination of every input variable and every possible key. The hamming weight counts the number of 1 bits in a bitstring. As a 1 bit has a different power consumption than a 0 bit, the power consumption of running the computation reveals some information about the key.

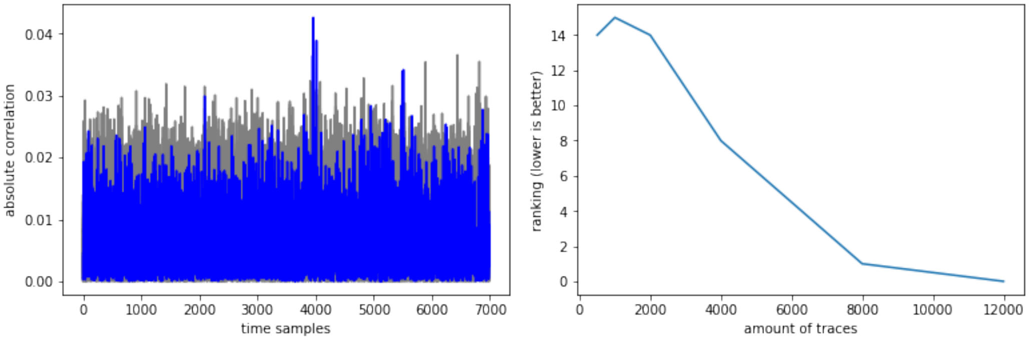
The next thing we compute is the correlation between our power-prediction matrix and a real world power trace. The size of the correlation matrix is [nr_keys x nr_samples]. The power trace is stored in a Matlab file. We load the trace file the same way as the input file. Our power-prediction matrix contains theoretical values of the power consumption and the traces are practical values of the power consumption. The key with the highest absolute correlation value, could be the secret key.

3 Results

We took the peak of the absolute Pearson correlation values per key and used this to determine which key had the largest correlation. In the following figure, it can be seen that key 6 (0b1010) has the highest correlation value.



On the left, we have plotted the absolute Pearson correlation values for every k candidate. The top candidate (key 6) is highlighted in blue.
The figure on the right shows the ranking that key 6 gets when only [500,1000,2000,4000,8000,12000] are considered


absolute correlation
time samples
ranking (lower is better)
amount of traces