A Side-Channel Attack on an Arbiter PUF

Project for IL1333 HT2021

1 Project rules

The project can be done in groups of max 2 people. The project is expected to deliver a 2-3 page report describing how you trained and tested deep learning models and the classification accuracy which your trained models achieve, i.e. the probability that the model correctly recovers the output bit response of the arbiter PUFs. Include in your report t-test results and plots showing the distribution of '0' and '1' PUF's responses at the trace point with strongest leakage. Include your training and testing scripts in the Appendix.

All files which you need for the project are contained in PUF_traces_and_scripts.zip which is available in Canvas. The included file README describes the installation process and content of directories. The basic training and testing scripts are included. If you use them without any modifications, you will get the classification accuracy similar to the one of provided examples of trained models. You can do any changes you like to increase the classification accuracy.

TIP: You can increase the classification accuracy of neural networks by using for testing traces captured for the same challenge with multiple repetitions and multiplying their score vectors (or, equivalently, adding logs). The directory puf0_avg0_blocks_of_1000 contains traces captured in blocks with 1000 repetitions. Include in your report results for different number of repetitions (e.g. 1, 10, 100, 1000).

The project submission deadline is February 1st, 2022. Upload pdf of your report to Canvas.

2 Problem formulation

The power traces contained in the directory traces are captured from an Xilinx Artix-7 FPGA implementing a 128-bit arbiter PUF.

Each trace contains 150 points corresponding to values of the total power consumed by the FPGA during the computation of the 1-bit PUF output response R to a random 128-bit input challenge C.

The traces in the directory $puf0_avg100$ are captured for the PUF implementation with no extra flip-flops (FFs). The traces in the directory $puf12_avg100$ are captured for the PUF implementation with 12 extra FFs added manually to the bitstream as described in [1] in order to make the side-channel analysis easier. In both cases, each trace is computed by repeating the power measurements 100 times for the same challenge C and taking an average (mean) of the results. The responses R are saved in files labels.npy.

In the third set of traces, $puf0_avg0_blocks_of_1000$, each trace represents a single measurement from the PUF implementation with no extra FFs. The traces are captured in blocks of size 1000. In each block, all traces is captured for the same challenge C.

References

[1] Y. Yu, M. Moraitis, and E. Dubrova, "Why deep learning makes it difficult to keep secrets in fpgas," in *DYMAMICS'2020*, 2020.