A Side-Channel Attack on an FPGA Implementation of an Arbiter PUF

Project for IL1333 2021

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 success probability which your attack achieves, i.e. the probability that the model correctly recovers the output bit response of the arbiter PUFs. Include your training and testing scripts in the Appendix.

All files 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 success probability similar to the one of provided examples of trained models. You can do any changes you like to increase the success probability. adding reduntant FF's coz thats what made [1] yield a succesful attack using only 1 trace ?

The project submission deadline is June 1st, 2021. 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 given 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, "Profiled deep learning side-channel attack on a protected arbiter PUF combined with bitstream modification," 2020. https://eprint.iacr.org/2020/1031.pdf.