6 Side Channel Analysis Attacks (Part 3)

Extracting Secrets from Power Consumption

A Introduction

In Lab 4 and 5, you began experimenting with SCA attacks, investigating statistical analysis of side channel power consumption and looking at patterns in the timing of power spikes to infer properties of the system under attack. In this lab, we will use different techniques to extract information from the power side channel itself by mounting a differential power analysis attack on AES.

B Background

The power consumed by a device at a particular instant in time will depend on what the device is doing at that time. In a processor, different instructions will require more or less power to run depending on what parts of the hardware are active. This can be easily viewed in the power side channel and leveraged in a simple power analysis (SPA) style attack. Similarly, different data being processed with the same instructions will require different amounts of power because processing a "1" bit will take different power than processing a "0" bit. More generally, this phenomenon can be observed in any unprotected application specific IC (ASIC), not just in a processor. Averaging repeated measurements and plotting a differential trace can improve the SNR and highlight regions of data-dependent information leakage (as in Lab 4). While SPA attacks can shed light on the particular algorithm being used (e.g. DES vs AES), more complex attacks like DPA require that the attacker know the algorithm in detail so it can be replicated offline. By running all possible inputs to the circuit through the relevant algorithmic processes offline, one can predict the most likely input circuit using a differential power analysis (DPA) attack. Note that DPA only aims to sort traces into the correct "bin" where a particular bit of an internal value is 0 or 1 (for a given key, input pair) according to the algorithm; you are not computing what the trace should be, just which trace represents a particular situation. Averaging the traces (sample-wise, not time-wise) helps to improve the SNR because outliers, values that are uncharacteristically high or low, will be effectively removed and the "true" value at that point in time will emerge. One way to break this assumption and make it more difficult to execute this attack is to decorrelate the information in the trace in subsequent iterations of encryption. Simply put, in a processor, one can add a random delay to some point of the operation such that when subsequent traces are captured and averaged together, the side channel traces will not "line up" and averaging will actually make the SNR *worse*, not better. This solution does not work for all attack types, especially higher order attacks, and can be defeated using more advanced signal processing techniques which attempt to stretch/re-align signals dynamically.

The other common, and often more powerful attack (in that it will generally require fewer traces than DPA to extract the encryption key) is a Correlation Power Analysis (CPA) attack. In the case of CPA, you are in a sense computing what *shape* you would expect the power (or other side channel) trace to take. You are leveraging the relationship between the Hamming weight (HW), or the number of 1s currently being processed in an internal variable, and the magnitude of the side channel trace at a given instant in time. In such an attack, numerous plaintexts will once again be encrypted, and the corresponding power traces will be recorded. In the case of the power side channel, the general shape of the power consumption will be estimated as the HW output from the

SBOX at a particular time, for a given set of inputs (plaintext, key). Just like with DPA, we do not attack the entire key at once, but rather a subkey, and the corresponding substring from the plaintext. We will compute the correlation coefficient between the predicted shape of the power consumption and the actual power consumption at the same point over a number of traces. Here, the correlation coefficient is defined as $r = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$, where $cov(X,Y)$ is the covariance of 'X' and 'Y', computed as $cov(X,Y) = \sum_{n=1}^{N}[(Y_n - \bar{Y})(X_n - \bar{X})]$, and σ_X and σ_Y are the standard deviation of the two datasets, calculated as $\sigma_X = \sqrt{\sum_{n=1}^{N}(X_n - \bar{X})^2}$. Note that, if you manually compute the correlation coefficient, if one of the signals is constant, e.g. $X = [1,1,1,1]$, the calculation of σ_X will result in a divide-by-zero. Python and Numpy handle this by setting the resulting correlation coefficient to NaN.

Reading Check

1. What are the primary differences between SPA, DPA, and CPA?
2. In your own words, describe the process for the DPA attack.
3. What is one technique for mitigating a basic DPA attack? What are some considerations / limitations to this solution?

C Hardware and Software Tools

For this lab, you will need the ChipWhisperer Nano or equivalent board, as well as a ChipWhisperer VM set up as in the previous lab.

D Getting Started

Unzip the PHS-Lab-06 folder. Navigate to localhost:8888 as usual and create a new directory under jupyter called lab-6. Upload the provided notebook files to this directory. Open Lab 6A (MAIN). Read through the walkthrough. You will need to also open Lab 6A (HARDWARE) and copy over the set-up / acquisition code into 6A (MAIN). The set-up code must be modified for the CWNANO by changing the scope type and platform accordingly.

Download and open the PHS-Lab-06 directory, which contains 3 folders" Firmware, Python, and Figures.

```
PHS-Lab-06
├─ Firmware
│  ├─ simpleserial-aes
│  ├─ Makefile.inc
│  └─ ...
├─ Python
│  ├─ Lab-06-SCA-A (MAIN).ipynb
│  ├─ Lab-06-SCA-A (HARDWARE).ipynb
│  ├─ Lab-06-SCA-B (MAIN).ipynb
│  ├─ Lab-06-SCA-C (MAIN).ipynb
│  └─ Lab-06-SCA-C (HARDWARE).ipynb
└─ Figures
```

The **Python** directory contains a Jupyter Notebook which walks you through the attack.

The **Firmware** directory contains the scripts needed to program your CW's target. In this experiment we are using the firmware in the "simpleserial-aes" folder. The folder includes a "makefile" to compile "simpleserial-aes.c" and "simpleserial-aes-CWNANO.hex" with pre-compiled firmware. Compilation is all done through the "Lab6x (HARDWARE)" notebooks.

The **Figures** directory is where you should save all your waveforms (i.e., plots) generated from the notebook.

E Lab Assignment

Work through Lab 6A (MAIN/HARDWARE), Lab 6B, and Lab 6C (MAIN/HARDWARE). Be sure to properly disconnect the scope and target in your current notebook before going to another notebook and running cells (scope.dis(), target.dis()). Failure to do so may result in undefined behavior from the ChipWhisperer. To fix it, restart the notebook kernel and disconnect/reconnect the ChipWhisperer.

For 6A, at the end, repeat the experiment using smaller Hamming weight differences (4 bit HW). Be sure to copy any plots/results before changing the acquisition code and re-running the notebook so you won't lose any results you need. Discuss in your report how the resulting peaks differ (i.e. is the spike still distinct when the difference in Hamming weight is smaller?).

For 6B, complete all sections.

For 6C, undergraduates should complete up to (and including) 1.7. Graduate students should also complete 1.8.1 and 1.8.2.

Finally, modify the firmware in Firmware/simpleserial-aes to include a random delay (like in lab 5) and re-run the notebook for 6C. Simply modify the jitter loop (line 44) to include a random component and recompile. To what extent did this impact the success of step 1.7?

F Badges and Achievements

This experiment involves the use of the **Chipwhisperer**, **firmware**, and **data visualization**. By successfully completing the lab you can unlock the following achievements:

- ML2: Machine Learning
- CRYPTO2: Side Channel Analysis
- C3: Firmware Programming in C
- PLOT5: Data Visualization

Depending on the demonstrated skill, each achievement will be further divided into bronze, silver, and gold levels.

G Deliverables

Submit your report along with all code. Follow all report guidelines as detailed in the Appendix. The following result(s) must be included in the report:

- Answers to the reading check questions.
- Two plots from 6A, one showing the differential trace for a HW difference of 8 (0x00 vs 0xFF), and one showing the differential trace for a HW difference of 4 (e.g. 0x00 to 0x0F) and discussion of the difference / reason for this difference.
- One plot from 6B from step 1.3.2
- One plot from 6C from step 1.6
- Discussion of the impact of a random delay on the DPA process (6C, 1.7)