Assignment Homework 2

Due date: Feb 10, 2023 (AoE) = Anywhere on Earth

Please complete this assignment (200 pts total) and submit your report/program code on Canvas (all files compressed in one `.zip` without the `.bin` files)

For side-channel analysis, it is often necessary to deal with substantial amounts of data that typically exceed the memory available in a computer. Consequently, it is not possible to load all data into memory and compute the result directly. Instead, the result must be computed in incremental steps. In addition, working with large data sets can introduce numerical instabilities. Of course, this can lead to significant problems and make a successful analysis impossible. Therefore, you are asked to investigate the numerical stability of your preferred programming environment to avoid follow-up problems in subsequent homework assignments.

1. The file `measurement_data_uint8.bin` contains 1 billion samples of type uint8. This is representative of actual measurement data, as many oscilloscopes have 8 bit of resolution due to the Analog-to-Digital Converter (ADC) being used. Your task is to compute the mean and variance of this data by applying different methods. Note: this is about incrementally processing the data and updating intermediate variables, to then compute the final result. Do not use any readily available functions for mean/variance (90 pts)

   a) Naïve approach [*] (5 pts)

   b) Welford's algorithm (cf. Wikipedia or other online reference) (10 pts)

   c) One-pass arbitrary order method [†] (35 pts)

   d) Histogram method [‡] (35 pts)

   e) Compare the runtime of the different methods (5 pt)

   Note: you can confirm your result by comparing the mean with the standard deviation. Aside from the shift of the decimal point, the initial four digits are the same and will be easily recognized as part of the hacker culture ("1337").

For the next two tasks, please consider the following files and the notation in the appendix:

- `traces_10000x50_int8.bin`, containing $\boldsymbol{I} = I_0 || \ldots || I_N$, whereas $N = 10000$ and $T = 50$
- `plaintext_10000x16_uint8.bin`, containing the corresponding plaintexts $\boldsymbol{P} = P_0 || \ldots || P_N$

[*] `https://en.wikipedia.org/wiki/Algorithms_for_calculating_variance`
[†] `https://eprint.iacr.org/2015/207.pdf`, only Section 4 and Appendix A
[‡] `https://eprint.iacr.org/2017/624.pdf`, only Section 3 and 4

2. In the lecture, we learned that not only key extraction may be a viable goal of a side-channel analysis but also leakage detection. In particular, leakage detection may help to choose points for more computationally demanding attacks or as part of a security certification process to confirm the lack of points of interest. The most basic leakage detection is the SNR (cf. Appendix). To develop a natural understanding of what is happening, first compute the numerator and denominator separately. (20 pts)

   a) Compute the numerator ('signal') of the traces and plot the result. (5 pts)

   b) Compute the denominator ('noise') of the traces and plot the result. (5 pts)

   c) Compute the whole SNR and plot the result. (10 pts)

   Caveat: many software libraries treat `0` differently when computing, e.g., variance. In numpy, you should use, e.g., `where=means != 0` to exclude zeros in your computation.

3. Perform a Correlation Power Analysis (CPA) on the traces to extract they key. Do *not* use any external libraries that provide readily available functions for side-channel analysis. Please report the execution time of your attack. (90 pts)

Please write all your programs in one of the following languages/environments: Python/Jupyter (*strongly recommended*), C/C++, Rust, Java, Matlab/Octave. *Thoroughly document your code using comments*. Your `.zip` file should contain your code, instructions how to make it run (if needed), the figures, etc.

A. Appendix

A.1. Notations

For a specific side channel experiment, we collect the set $\boldsymbol{I}$ of traces with N being the number of collected traces. One trace I of length T is represented by its samples $I = (i_0, ..., i_T)$ which have been acquired over time. The plaintext is denoted as $P = p_0||...||p_{15}$ and the key as $K = k_0||k_1||...||k_{15}$. The target intermediate value of the AES S-box is defined by $v_{i,n} = \text{SBOX}(k_{i,n} \oplus p_{i,n})$ for the subkey and plaintext of target byte $i \in [0, 15]$ and trace number $n \in [1, ..., N]$.

We denote $\oplus$ as the bitwise XOR-operator, $\boldsymbol{V}$ as the set of all possible intermediate values v, and $|\cdot|$ as the number of elements in a set. Whenever accessing a single value of a trace with number n, point in time t, and intermediate value v we denote this as $i^v_{n,t}$. $\boldsymbol{I}_v$ denotes the set of traces for the intermediate value v.

A.2. Signal-to-Noise Ratio (SNR)

When investigating the properties of a measurement campaign from a security point of view, we are mostly interested in the *effectiveness* of distinguishing the targeted values. For this purpose, we recall the SNR definition by Mangard et al., denoted as:

$$\text{SNR}_\text{M} = \frac{\text{Var}(\text{E}[\boldsymbol{I}_{X0}], ..., \text{E}[\boldsymbol{I}_{X|\boldsymbol{V}|}])}{\text{E}[\text{Var}(\boldsymbol{I}_{X0}), ..., \text{Var}(\boldsymbol{I}_{X|\boldsymbol{V}|})]} \quad \forall\, X_j \in \boldsymbol{V} \tag{1}$$

It is a useful tool to identify the points in time that have leakage in their first statistical moment.