Assignment Homework 2

Due date: Feb 12 (end of day)

Please complete this assignment (100 pts total) and submit your report/program code on Canvas (all files compressed in one `.zip` without any data sets)

Part 1

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, ideally, in a 'one-pass' fashion (reading the data only once). 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 and I suggest to process each sample individually, to 'emulate' processing the incremental way of how data needs to be processed for Part 2 (40 pts)

   a) Naïve approach * (5 pts)

   Execution Time: 1.396045 seconds

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

   Execution Time: 7.194937 seconds

   c) One-pass arbitrary order method † (10 pts)

   Execution Time: 6.838472 seconds

   d) Histogram method ‡ (10 pts)

   Execution Time: 1.658899 seconds

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

   The Naïve approach outperforms other methods in terms of speed, which I think is because its calculation process is relatively simple. Although Welford's algorithm and One-pass arbitrary order method take longer to compute, they provide better numerical stability. In contrast, although the Histogram method has moderate execution time, it is not as numerically stable as Welford's

algorithm and One-pass arbitrary order method.

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").

* 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

Part 2

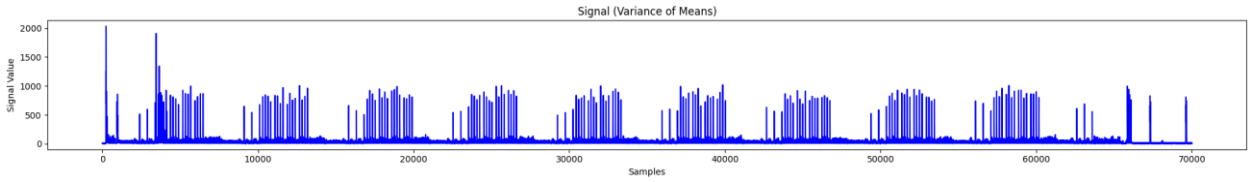
For the next two tasks, please download/install SCARR: https://github.com/decryptofy/scarr. In addition, please download the following data set from Box.com (see SCARR's README.md): `benchmark_medium_sw_aes128_compressed.zarr.tar`. Note that a Zarr data set is a directory and the above data set is archived – you need to untar it first; this does not remove the compression. You are allowed to use SCARR to verify your SNR and CPA computations, but you are not supposed to adopt code from SCARR for your own homework assignment. Based on the size of the data set, you may experience the following:

- You computer does not have enough memory: please use one of the 'online'-algorithms from Task 1
- Your computation is not very fast: focus on the first 5000 points only

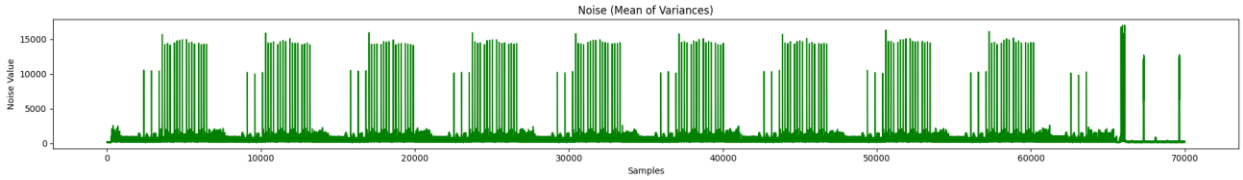
If submissions are sufficiently fast, we can do another scoreboard (vs. SCARR). I invite you again to compete against the best. For the benchmark scenario, all of the data set needs to be considered (all traces and all points).

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)

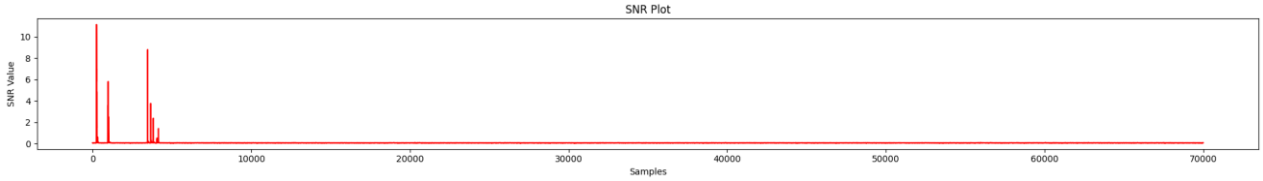


   b) Compute the denominator ('noise') of the traces and plot the result. (5 pts)

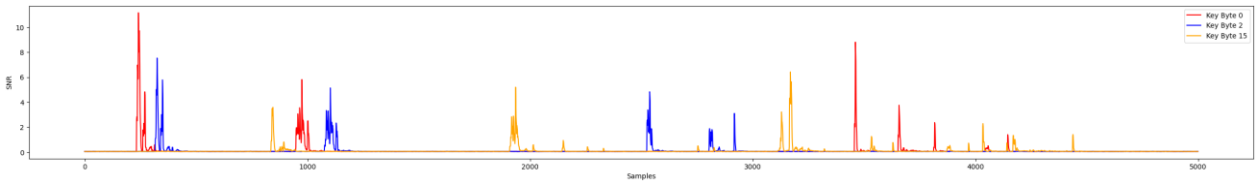


   c) Compute the whole SNR and plot the result. (10 pts)

   One Key Byte: 3.313729 seconds



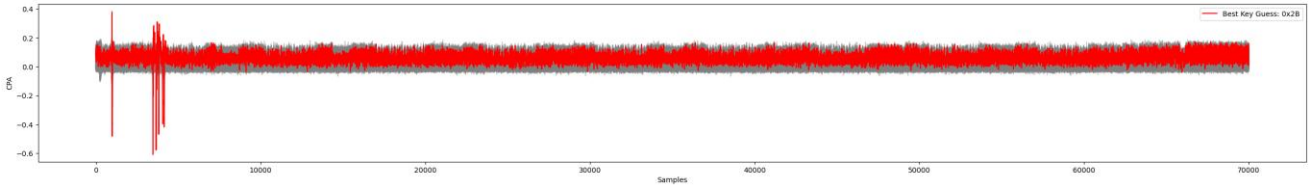
   All Key Byte: 57.283338 seconds



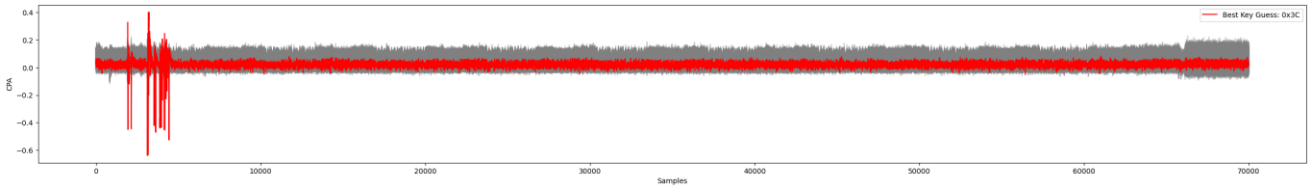
Caveat: many software libraries treat `0` differently when computing, e.g., variance. In numpy, you should use, e.g., `where=means != 0` to exclude zeros in your computation.

3. Perform a Correlation Power Analysis (CPA) on the traces to extract they key and plot the results. Do *not* use any external libraries that provide readily available functions for side-channel analysis. *Please report the execution time of your attack and the properties of your system, otherwise I will deduct points*. I am fully aware that the key is part of the data set, i.e., you can easily verify the results yourself. (40 pts)

   One Key Byte: Execution Time: 43.491906 seconds



   All key byte: Execution Time: 630.248197 seconds



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:

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

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