Bit-by-Bit: The Side Channel Hackathon

Challenge 3

21st September 2025

Challenge Task: Side-Channel Key Recovery with Neural Networks

In some applications, the number of traces that can be collected from a target device is very limited. Consequently, standard attacks such as CPA (Correlation Power Analysis) performed using those traces may fail to recover the correct key. In such scenarios, a more powerful adversary can still succeed by using a clone of the target. Concretely, suppose the adversary aims to recover the secret key of device A but can collect only a small number N_a of traces from A. For each of these N_a traces the adversary knows the corresponding plaintext and ciphertext, but not the secret key. Because N_a is small, CPA on device A alone is ineffective.

When CPA and similar direct attacks fail, the adversary may obtain a second device B that is identical (or a close clone) of device A. Device B can be bought on the market or acquired by other means and is under the adversary's full control. The adversary sets B's secret key to a known value (typically different from A's unknown key) and collects a large number N_p of traces by repeatedly executing the encryption operation on B. For each trace from B the adversary knows the plaintext, ciphertext, and the secret key. Using this dataset (trace, plaintext, key) from B, the adversary trains a machine-learning model to predict an intermediate sensitive variable (this is typically a variable dependent on the secret key but different than it) from a trace. The model takes a trace as input and outputs a probability distribution over the possible values of that intermediate variable.

Once trained on B, the model is applied to the N_a traces collected from device A to produce, for each trace, a distribution over the intermediate variable's values. The adversary aggregates these per-trace distributions across the N_a traces (e.g., using likelihood aggregation) to compute a ranking over candidate secret keys for device A. In this way, a clone device with plentiful labeled traces enables a successful side-channel attack even when only a few traces are available from the target.

Dataset

You are given two datasets collected from AES encryption devices.

Dataset A (target). Traces were recorded from device A while it performed AES encryption. Each trace is paired with the first plaintext byte only. Your task is to recover the secret key used by device A; because the dataset contains only the first plaintext byte, it is sufficient to recover the first byte of the key.

Dataset B (profiling / clone). Traces were recorded from device B (a clone or identical device). Each trace in this dataset is paired with the first plaintext byte and the corresponding key byte (which is known for device B).

Method

As stated earlier, the attack needs to be performed using two steps.

Training using Dataset from Device B

- Use the dataset from device B (plaintext, key byte, and trace) to train a neural network F that takes a trace as input and predict a probability distribution $\mathbf{p}$ over a intermediate sensitive variable Z. Z may the $Sbox(P \oplus K)$ or $HW(Sbox(P \oplus K))$ where P and K be the random variables representing plaintext and key byte respetively. Therefore, $\mathbf{p} = F(\mathbf{t})$ where $\mathbf{t}$ is a trace. Note that since $\mathbf{p}$ is a probability distribution over the values of Z, $\mathbf{p}$ will be a vector of dimension $|Z|$ where Z can take any value between 0 to $|Z| - 1$. $\mathbf{p}[z]$ represents the probability of $Z = z$ for trace $\mathbf{t}$.

- To train the neural network, we can generate the label for each trace $\mathbf{t}_i$ as $z_i = Sbox(p_i \oplus k_i)$ or $z_i = HW(Sbox(p_i \oplus k_i))$ where t_i, p_i, and k_i be the i-th trace, plaintext, key respectively. Then $\{(\mathbf{t}_i, z_i\})_{i=1}^{N_p}$ will be your labelled dataset used to train the neural network.

Key Recovery of Device A

In this step, use the neural network model F trained in the previous step and the dataset collected from device A to recover the secret key of A. The procedure is as follows

- For each possible key $k \in K$, compute the log-likelihood score:

$$score(k) = \sum_{i=1}^{N_a} \log\left(\tilde{\mathbf{p}}_i[\tilde{z}_{i,k}]\right)$$

  where:

  - N_a is the number of traces from device A,

  - $\tilde{\mathbf{p}}_i$ is the probability distribution for trace i.
  - $\tilde{z}_{i,k} = Sbox(\tilde{p}_i \oplus k)$ or $HW(Sbox(\tilde{p}_i \oplus k))$

- Collect these scores into a **guessing vector** $G = [G_0, G_1, \ldots, G_{|K|-1}]$, sorted so that G_0 is the most likely key.

- The **rank** of the true key is its index in G.

Deliverables

- Your trained neural network model and a description of its architecture.

- A sorted list of possible keys sorted by their likelihood. The rank of the correct key should be higher (i.e., it should appear earlier in the list).

- A short report including:
  - Your methodology and justification.
  - A sorted list of possible keys.