Project # 4

This project is due Thursday, March 19.

1. *Information from Side Channels*
   Let X be a random variable on the discrete space $\mathcal{X} = \{0, 1, \ldots, 255\}$ with a uniformly distributed probability distribution.

   (a) Calculate the probability $p_i = P(X = x)$ and the entropy $\mathsf{H}(X)$.

   The Hamming Weight function $\mathrm{HW}(x)$ of an 8-bit value $x = (x_7 x_6 ... x_0)_2$ is defined as $\mathrm{HW}(x) = \sum_{i=0}^{7} x_i$ which implies $\mathrm{HW}(x) \in \{0, ..., 8\}$. Let $\mathrm{HW}(X)$ be the random variable denoting the Hamming Weight of the values x of X. If x is an 8-bit string, $\mathrm{HW}(X)$ takes values in $\mathcal{W} = \{0, 1, \ldots, 8\}$.

   (b) Calculate the probability $p_w = P(\mathrm{HW}(X) = w)$ for all possible $w \in \mathcal{W}$. Find the entropy $\mathsf{H}(\mathrm{HW}(X))$ of the Hamming Weight of X.

2. *Differential Fault Analysis of AES*
For this attack we assume an error is induced into an implementation of the AES-128 to produce a faulty ciphertext. The error is a one-byte error and is always induced on the last state byte, during round 9, before the `MixColumns` operation. Two pairs of ciphertext and corresponding faultytext are provided. The induced errors are unknown and can be different. However, the key is the same in both cases. Your goal is to recover 4 bytes of the last round key.

   (a) Which bytes of that round key can you recover? What is their value?

   Hint: Recall that for a given ciphertext byte c and corresponding faultytext byte c' and a key candidate k, the resulting input difference Δ (the difference between the correct and the faulty intermediate state) is computed as

$$\Delta = S^{-1}(c \oplus k) \oplus S^{-1}(c' \oplus k)$$

   Here $S^{-1}(\cdot)$ is the inverse of the `SubBytes`.

   Hint: Recall that a one-byte fault before the `MixColumns` will result in dependent faults in four of the output bytes. Furthermore, a one-byte difference Δ before `MixColumns` will result in dependent differences in four s-box computations in the following round.

   These are the recorded ciphertext/faultytext pairs in hex notation (remember that only 4 bytes of the last round key are to be recovered):

```
ciphertext1 = 0xe719f8ab9e0b846f0cf2e5c32a0e5b45
faultytext1 = 0xe719f86f9e0beb6f0c97e5c38b0e5b45

ciphertext2 = 0x78f272c7cf5383085fa240236d97130f
faultytext2 = 0x78f27277cf53e7085f944023fa97130f
```

Differential Power Analysis:

Preparation: Assume an adversary that is able to send a number of challenges to an implementation of which he is able to predict a data dependent and key dependent intermediate value f(d, k), where d is known data and k is a part of the key. The adversary is additionally able to sample the power consumption while the intermediate value is processed.

Acquiring data: The adversary sends D challenges with the known data d_l, $l \in \{1, ..., D\}$. He stores all data values in the data vector $\mathbf{d} = (d_1, ..., d_l, ..., d_D)^T$. For each challenge d_l, the adversary records one power trace $\mathbf{t}_l = (t_{l,1}, ..., t_{l,j}, ..., t_{l,T})$, where T is the number of samples $t_{l,j}$, $j \in \{1, ..., T\}$ for each trace. All power traces of all challenges are stacked row-wise to form the $(D \times T)$ matrix $\mathbf{T}$.

$$\mathbf{T} = \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_l \\ \mathbf{t}_D \end{pmatrix} = \begin{pmatrix} t_{1,1} & \cdots & t_{1,T} \\ \vdots & t_{l,j} & \vdots \\ t_{D,1} & \cdots & t_{D,T} \end{pmatrix}$$

Calculating hypothetical intermediate results: The partial key k can take K possible values. All key hypotheses k_i, $i \in \{1, ..., K\}$, are stored in the vector $\mathbf{k} = (k_1, ..., k_i, ..., k_K)$. The adversary is now able to compute the hypothetical intermediate values $v_{l,i}$ =f(d_l, k_i), which again form a matrix $\mathbf{V}$ of size $(D \times K)$.

$$\mathbf{V} = \begin{pmatrix} v_{1,1} & \cdots & v_{1,K} \\ \vdots & v_{l,i} & \vdots \\ v_{D,1} & \cdots & v_{D,K} \end{pmatrix}$$

Recall that the goal of the adversary is to find the correct hypothetical key, hence to choose the correct column of this matrix.

Mapping intermediate values to hypothetical power consumption values: The adversary will now map $\mathbf{V}$ to $\mathbf{H}$, where $\mathbf{H}$ is the hypothetical power consumption of the device in the moment the the intermediate value v =f(d, k) is processed. The adversary needs to know how the target device leaks information. The chosen power model in this case is the Hamming Weight HW(v), leading to $\mathbf{H} = \text{HW}(\mathbf{V})$.

Finding the 'best' key hypothesis: The adversary compares each column of the the hypothetical power consumption $\mathbf{H}$ (each column corresponds to one key hypothesis k_i) to each column of the actual power consumption $\mathbf{T}$ (each column of $\mathbf{T}$ corresponds to one point in time, j). Hence the adversary browses through all keys *and* through all points in time to find (i) when the data leakage occurs and (ii) for which key it occurs. In other words he checks when and for which key he was able to 'best' predict the power consumption of the device. He stores the results of this comparison in a matrix $\mathbf{R}$ of size $(K \times T)$, where each element $r_{i,j}$ is a metric to measure a relation between the predicted and the actual power consumption.

DPA using correlation: A strong metric for the relation of two variables is the correlation. In DPA, the power consumption at time instance j (the jth column of **T**) is correlated to the hypothetical power consumption of key guess i (the ith column of **H**). Both vectors have the same length D. This correlation yields the correlation value $r_{i,j}(\mathbf{h}_i, \mathbf{t}_j)$. It is typically calculated using the Pearson correlation coefficient

$$r_{xy} = \frac{\sum x_i y_i - n\overline{x}\,\overline{y}}{(n-1)s_x s_y}$$

Determining all $r_{i,j}(\mathbf{h}_i, \mathbf{t}_j)$ finally results in the correlation matrix **R**.

The expression for $r_{i,j}(\mathbf{h}_i, \mathbf{t}_j)$ can be used to calculate all values of the correlation matrix **R**.

3. Implementation Assignment: **Differential Power Analysis:**

   The power consumption **t** of an AES engine was recorded for all possible inputs of the first byte x. The power consumption has been measured while the output $y = f(x, k)$ of the first s-box was processed. Use the provided 500 power traces to recover the secret key byte k.

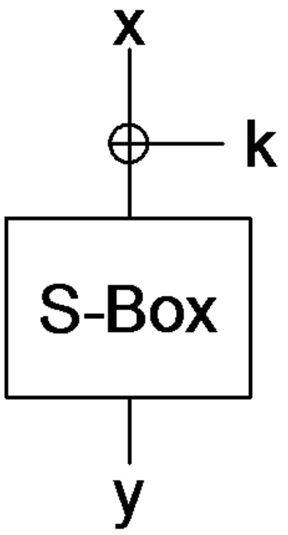

Figure 1: AES XorKey() and SubBytes() for one byte.

   The format of the input files is organized as follows: Power traces are stored in `PowerTrace.dat`. There are 500 traces in this file; each trace contains has 30,000 points stored in 8 bit unsigned integer format. The corresponding 500 plaintexts are stored in `plaintext.txt` file Each plaintext consists of a 128-bit number in hexadecimal format, one plaintext per line. Each byte of the plaintext can serve as the input x to the AES shown in Figure 1.

   (a) Implement the routine to read the data from the sample files. Compute the sample mean trace $\overline{t}$ and the sample variance trace S_{tt} for each point in time (again using the provided routine). Plot the results over time with a tool of your choice (gnuplot, matlab, excel, ...).

(b) Fill the routine that generates the s-box output y for a given value x and a given key byte k. Perform this attack to recover the first key byte k_0 using the first byte of the plaintext Additionally you should fill the routines that calculate the hamming weight $\mathrm{HW}(x)$ of a given byte x (or return the ith bit of a given byte x). Use these functions to calculate all hypothetical intermediate values y for the given x and all k. Also calculate the sample mean $\overline{y}$ and sample variance S_{yy} for all possible k.

(c) Build a DPA using the correlation coefficient method. The routine should return the most probable key hypothesis and the correlation value between each $\mathrm{HW}(y)$ and the power consumption P_t for each point in time. Plot the correlation traces for all 256 key hypotheses into one plot. Mark the correlation trace to make it distinguishable from the other traces. What is the best key hypothesis for the given input files?

(d) Instead of the Hamming weight of $f(x, k)$ compute the correlation coefficient (i) between $\mathbf{t}$ and the most significant bit of $f(x, k)$ and (ii) between $\mathbf{t}$ and the least significant bit of $f(x, k)$ for each key hypothesis and identify the 'best' hypothesis as result of DPA. Again, plot the correlation traces of all possible key values k into one plot. What is your observation regarding the quality of the DPA results?

4. Bonus Problem: **Full Key Recovery**
Recover the full 128 bit key of the implementation for an additional 2 bonus points.

Good luck and have fun!