A white background with black text

Description automatically generated

A graph of a graph

Description automatically generated

The peaks correspond to a heavy, high-power operation. Each power peak probably corresponds to a round key addition operation, since 11 round key addition operations happen each time AES-128 encrypts/decrypts.

A close-up of a paper

Description automatically generated

An easy target operation for DPA is the first round key since it is simply an XOR of the plaintext against the key.

A graph of a sound wave

Description automatically generated

Key guesses 0 and 255 have the highest correlation.

A graph of a sound wave

Description automatically generated



0x00 is the correct key guess. 0xFF has a high negative correlation because it is the exact opposite of the correct key, so its hamming distance guesses are negatively correlated with the peak power.

A close up of a text

Description automatically generated

The point that has the maximum power leak is the first peak, around index ~48.

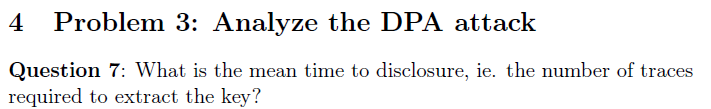
A graph of colored lines

Description automatically generated

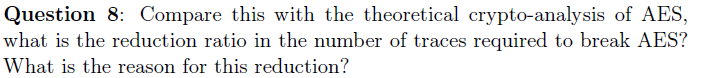
The correct key guess always showed the highest correlation from the very beginning, but it stabilizes around 3000.



0x000102030405060708090a0b0c0d0e0f



16 bytes, 256 traces per byte -> 16 \* 256 = 4096 traces total



AES bruteforce: 2^128  
DPA: 4096 = 2^12  
Reduction ratio: 2^16

The reason for this reduction is we can bruteforce just one byte at a time, avoiding the exponential increase.

A close-up of a white background

Description automatically generated

The first peak typically varies between index 51 to index 52 for the 10,000 traces. Instead of just always sampling at index 51, you could instead perform cross correlation first to align the traces, and then sample that point. Or you could use a sliding window to find the high peaks and find the value of the highest point there. I implemented the former method (see function track\_peaks() in my code).

A close-up of a white background

Description automatically generated

First, you can use DPA to calculate the last round key, since the last step of AES is find the ciphertext by XORing the output of ShiftRows and AddRoundKey. We can then invert ShiftRows and SubBytes trivially.

At this point, we have the output of the second-to-last AddRoundKey. We can then use DPA again with the same concept as before to find the second-to-last round key. With this key, we can find the output of MixColumns. Using InvMixColumns, we can inverse MixColumns and find the output of ShiftRows.

At this point, we have inverted every type of operation in AES128 (other than round key generation). We can then repeat these operations for every single step in AES128 to completely reverse the encryption and find the original key.