Lab 2: Correlation Power Analysis Attack Report

Submitted by Germain Mucyo
Email: mucyo.g@northeastern.edu
NUID: 002301781
Course: EECE5699 Comp Hardware and Sys Security
Date: October 11, 2024

Introduction

This lab focused on implementing a **Correlation Power Analysis (CPA)** attack to recover a full 16-byte round key used in the AES encryption process. I utilized power traces captured from an FPGA board running AES encryption to perform the attack. The provided power traces and ciphertexts were analyzed using the Hamming Distance (HD) power model, and the attack aimed to recover all 16 key bytes.

The extra credit task involved attacking without prior knowledge of the leakage point and using the Hamming Weight (HW) power model for a potential secondary attack.

(a) Figure 1 shows a power trace representing power consumption during AES encryption. The x-axis denotes sample points, while the y-axis represents the measured power consumption at each sample point in my case I used the traces. This specific trace shows 11 significant dips, which correspond to the encryption operations during the AES rounds.

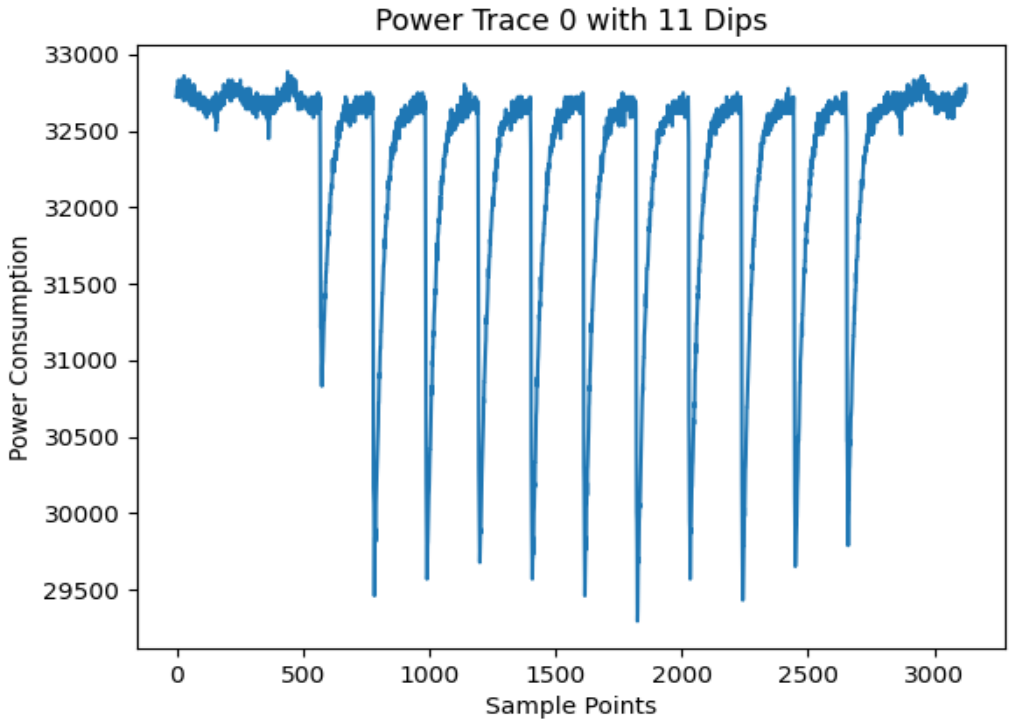


Figure 1 Power Trace plot

Q1: Explain why there are 11 dips instead of 10 dips.

Ans: In my case there are 11 peaks in the power trace. The first peak is the loading of plaintext into the register and the following 10 peaks are 10 rounds of the AES.

The **initial AddRoundKey operation** happens before any of the 10 main AES rounds begin. In this step, the plaintext is XORed with the first-round key, causing a noticeable change in power consumption, as Figure 1 shows the first peak is slightly differ from the rest.

After the initial AddRoundKey, AES enters its main rounds. Each of these rounds consists of several transformations (SubBytes, ShiftRows, MixColumns, and AddRoundKey). For each of these 10 rounds, the operations lead to a spike in power consumption, followed by a dip as the operations conclude. Overall, looking at my power trace I plotted, the 10 rounds are likely spaced relatively evenly after the first dip.

(b) A plot showing all key bytes recovery by CPA with 16 subplots, each representing a key byte. Figure 2.

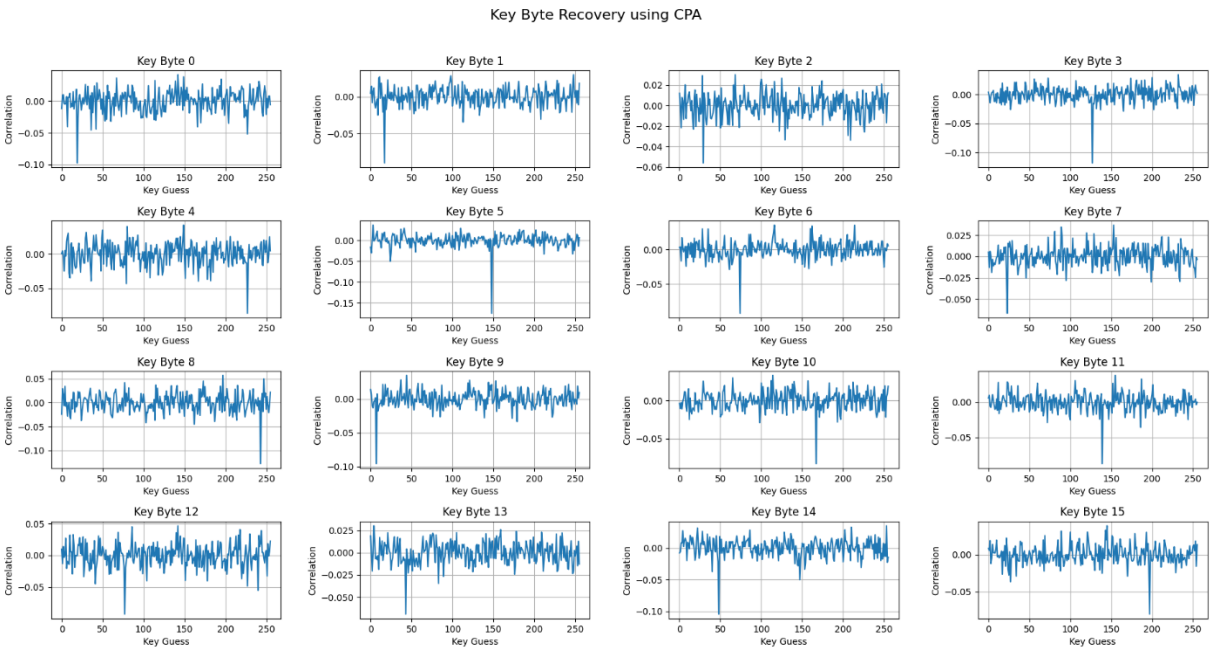


Figure 2 Key byte recovery using CPA (16 subplots).

The codes have comments to explain each step but let me explain how I came up with this plot; To recover the 16-byte AES key using Correlation Power Analysis (CPA).

Step by step explanation: I first loaded the power traces and corresponding ciphertexts. Then, I used the AES inverse S-box to calculate hypothetical intermediate values for each possible key guess (0–255) for each byte. By applying a Hamming distance model, I computed the power consumption estimates based on the XOR of the ciphertext byte and key guess, followed by the inverse S-box transformation.

Next I used the sample index 2663 as point to calculate the Pearson correlation between the predicted power values (based on the Hamming distance model) and the actual trace data. By examining how well these values correlate, I identified the most likely key byte for each of the 16 key positions in AES. Finally, I plotted the correlation values for all 16 key bytes, where the peaks in each subplot represent the most probable key byte values. Another important factor I should mention is that while doing this process of the key recovery I

(c). Results: Recovered AES Key

The following is the 16-byte AES last-round key recovered through the Correlation Power Analysis (CPA) attack is presented below in hexadecimal format **[0x13, 0x11, 0x1d, 0x7f, 0xe3, 0x94, 0x4a, 0x17, 0xf3, 0x07, 0xa7, 0x8b, 0x4d, 0x2b, 0x30, 0xc5].**

EXTRA CREADIT PART

(d)Attack Without Knowing the Leakage Point

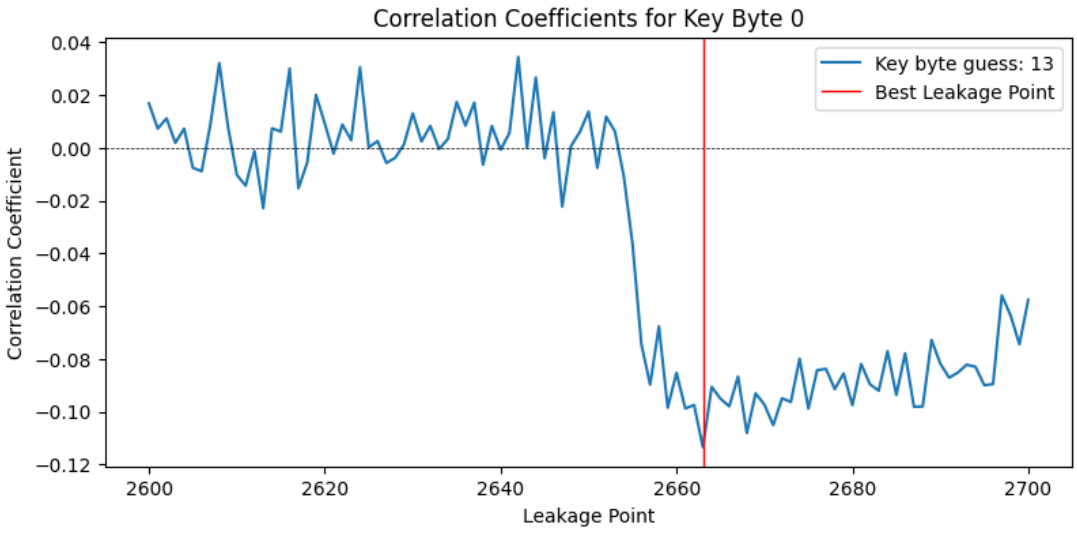
For the extra credit task, I performed the CPA attack without knowing the exact leakage point. Instead, I scanned across the time points in the range **[2600, 2700]**, calculating the correlation for each key guess across all time points.

I focused on key byte 0 and used a triple nested loop to iterate over the leakage points (2600 to 2700), key byte guesses (0 to 255), and power traces (0 to 7000). For each key byte guess, the script calculates the Hamming distance between the output of the inverse S-box and the corresponding ciphertext byte. It then computes the Pearson correlation coefficient between the calculated Hamming distances and the power traces to identify the leakage point that yields the highest correlation. Finally, the prints the best key byte guess and its corresponding leakage point, along with an optional visualization of the correlation coefficients (See image below)

The output is as follow:

- **Best Leakage Point for Key Byte 0**: **2663**
- **Best Key Byte Guess for Key Byte 0**: **13**
- **Max Correlation**: **0.1135**

Conclusion: The leakage point at **2663** was confirmed as the leakiest point for key byte 0, matching the known value.

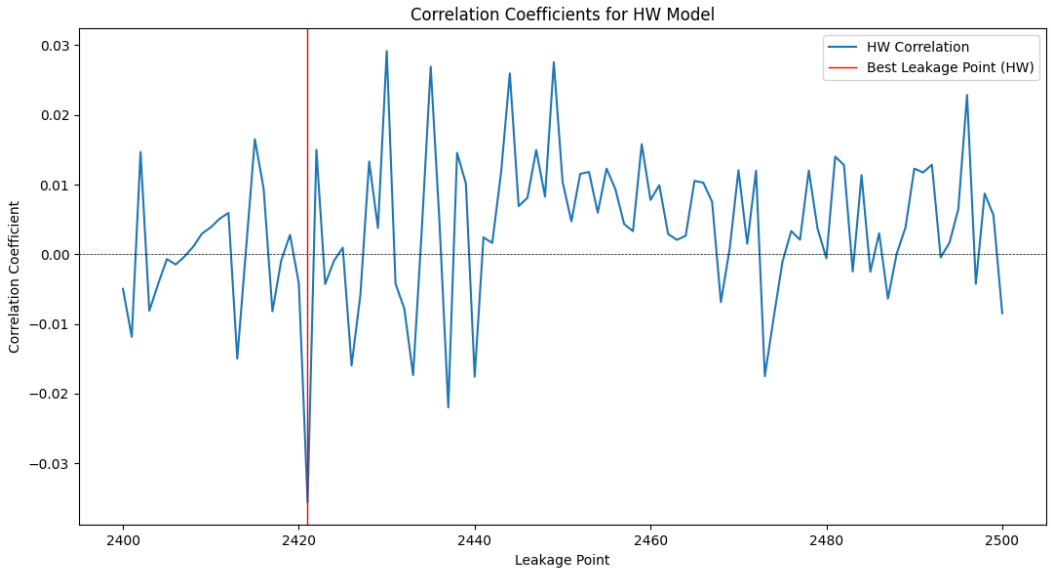


(e) I simulates the key recovery process for AES encryption by analyzing power traces and ciphertexts, focusing on the first key byte. It calculates the Hamming Weight of the inverse S-box output based on the ciphertext and a predefined correct key byte, and computes Pearson correlation coefficients between the calculated Hamming Weights and power traces for a specified range of leakage points (2400 to 2500). After identifying the leakage point with the highest correlation.

The script outputs the best leakage point and maximum correlation value, and optionally visualizes the results with a plot to illustrate the correlation coefficients, aiding in the analysis of the key recovery process. See the image below. Below are the output I have.

Best leakage point (HW model): 2421

Max correlation (HW model): 0.03561016532860794


Correlation Coefficients for HW Model
HW Correlation
Best Leakage Point (HW)
Correlation Coefficient
Leakage Point
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
2400
2420
2440
2460
2480
2500