Side Channel Analysis – CPA

Jordan Haws, Cameron Frandsen
Utah State University – Spring 2017

Abstract - This paper describes the process behind CPA attacks. This paper will describe in detail how to extract the round key, and the actual key from that.

I. Introduction

All electronic devices use electrical power to operate. Power consumed by a device is generally considered to be relatively constant and uniform but this is not the case. Any given device consumes as much power as it needs in a specific instant. For this paper we will be discussing how we analyzed power data from a device running a Data Encryption Standard (DES) encryption algorithm.

The electrical power being used by this device was monitored as it encrypted thousands of messages using DES. We will show in this paper that it is possible to decipher what specific computations the machine made as well as the encryption key used with enough data points for the electrical power. We used a Correlation Power Analysis (CPA) to correctly gather this information from the given power traces (data).

II. Project

We used the data for this project provided by the 2008/2009 DPA Contest. This contest is hosted annually, and promotes hardware side channel attacks such as the one performed in this paper. We were given some Matlab code that runs the DES algorithm and the data.

The each message was accompanied by a power trace, key, and encryption data in the header file. This header file data was used to verify that the analysis was correct and that the attack was successful in obtaining the encryption key.

After this attack was performed successfully, we modified our attack in two different ways to optimize how many power traces were needed to find the round keys. After the round keys were found, the last eight bits in the key were guessed by brute force until the entire encryption key was found.

III. Software

This attack was programmed in Matlab since most of the calculations were in matrices. To start this attack, the power traces were converted into csv format using code that was posted on the competition's site. Once the files were converted, a matrix was created to hold all of the file names so that the next file could be read in when needed.

The next step was to find all of the hamming distance (the number of corresponding bits that are different) for all 64 guesses for each Substitution Box (SBox). These hamming distances were taken from register 1 to register 2. Between register 1 and register 2, the bits start to get encrypted. The first four bits of register 1 do not correspond to SBox 1, but the first four indexes of the index matrix correspond to SBox 1.

After all 512 hamming distances were found, (64 guesses for 8 SBoxes) a Pearson's correlation was calculated between the the hamming distance and the voltage. However when there is only one power trace and one set of hamming distances in the Matlab Correlation function, the correlation coefficients were all NaN. The next power trace was read in and added to the matrix with it's corresponding hamming distance. When the correlation coefficient is calculated this second time they were all 1 or -1. Only after a third trace was added did the coefficients start to change and differ from one another.

The correlation function returns a correlation for every guess to every point in the trace. Since we were only looking at the first register, we only cared about the first spike in the trace, which corresponded to index number 5700 to 5800. The only point that mattered in this set is the peak power that corresponded to the first round of DES performed. After finding which of the 100 points is the highest that was the only column that was important to our correlation. The highest value in that column is the guess with the highest correlation. This was done for all 8 SBoxes and after getting the same guess 100 times in a row we were sure we had the correct round key.

After finding the round key, the real key was calculated by reversing the round key scheduler. However, that only gave 48 bits of the total 64 bit key. The next 8 bits were found by brute force. The DES algorithm was ran up to 64 times, and each time we checked to see if the output message was correct. Once the message came out correctly, the parity bits were calculated and added into the key. This gave the 64 bit key.

A. Optimization 1

The first optimization is sorting the traces by the highest peak values. We sorted the traces in Matlab by the peak values. By alternating between highest values and and lowest values, the correlation was able to measure greater differences, and able to find the round key faster.

B. Optimization 2

The second optimization was to try and optimize the hamming distances. If the hamming distances have a greater chance of being different, the correlation would be able to measure the difference better and be able to cancel out the noise more efficiently. It would be better to use the hamming distances that are more different from each other. However, it would be very impractical to read in all of the messages, and calculate all of the hamming distances in order to sort by hamming distance. So we parsed the original messages, looking for the messages that had the most 0s or 1s in the message.

V. Conclusion

The baseline and optimizations of this power analysis side channel attack are given below. The baseline is listed first, and then three variations of two optimizations.

Table 1: Results

Optimization	**Traces Needed**
Baseline	334
Max Message Sorted	474
Min Message Sorted	298
Max and Min Message	387
Max Round 1 Power	712
Min Round 1 Power	594
Max and Min Power	163

Our baseline code did fairly well and used relatively few power traces to find the correct round keys.

As a whole the message sorting section was not impressive. Running the messages with the most zeros seemed to do better, but we suspect that it might not necessarily be a better optimization, but just ran better with the given traces we had. We think this because the combined max and min runs should have been the best of the three and better than the original baseline, but it wasn't.

Both the max and min power options gave us worse results than our baseline. This makes sense because when the traces are organized by power so close together the messages themselves were probably very close. When close messages are correlated the noise is not cancelled as much as a random assortment. This is supported by results of the combination of the max and min power optimization. It performed far better than the baseline and we attribute this to the alternating variations in the traces.

This method of side channel analysis is very feasible and possible to gather the encryption data for a DES algorithm. These optimizations show that only a small number of power traces are needed to crack this encryption.