Side-Channel Analysis: Correlation Power Attack on a Data Encryption System Device

Sergio Gonzalez & Taylor Spencer, Utah State University

***Abstract*–The objective of this report is to explain how a power-based attack is possible on an embedded device. A Correlation Power Attack (CPA) algorithm was implemented on a simple embedded crypto-device. Using power traces cipher-keys were guessed and correlated with the known key. Several possible optimizations for obtaining knowledge about the key using a minimal number of traces will also be defined.**

***Index Terms*–Correlation Power Analysis (CPA), Side-Channel Analysis, Data Encryption Standard (DES), Embedded Systems Cipher.**

I. Introduction

Suppose you have access to some encryption device with a hardwired key being used. For any given message sent through this device, power traces of the encryption algorithm can be recorded while generating and returning a cipher-text. A correlation power attack (CPA) correlates a power trace taken from some known message and hidden key with a table of expected power produces given by manipulating the key provided for encryption.

This process of correlating a series of key guesses with the actual power trace for a given message is done repeatedly until accurate inferences can be made about the key. While this is done more effectively on a small portion of the key, this still proves to be extremely productive. If this process is done to gain knowledge about 12 bits of the 48 bit key, this reduces the number of possible keys from roughly $2.8 * 10^{14}$ to $6.8 * 10^{10}$. That reduction is massive and could equate to years of processor time.

A. Preliminary Research

The basis of work for the side-channel analysis came from the in-class lecture notes. Because of the simplicity of the DES encryption algorithm when compared to more modern AES and other algorithms, the DES algorithm was implemented. While this simpler algorithm was chosen to ease in demonstration of a possible attack, similar methods could be used to obtain the passkey of a more sophisticated algorithm. The group responsible for the annual DPA contest, (http://www.dpacontest.org/index.php), provided the tracefiles and other valuable information used for this demonstration.

II. CPA Base Implementation

A simplified explanation of DES algorithm begins with some message and some passkey. The message is split into two halves at the beginning of each round, the right half is introduced, through an XOR operation then to a modified version of the passkey, having also been through a series of permutations. The message is then permuted again and switches places with the left half of the message at the end of the round. A typical DES implementation includes 16 rounds. Note that this is an extremely simplified explanation of the DES algorithm. The key information to pull from the algorithm is the reveal of sensitive data with regards to the passkey.

The sensitive data revealed is simply an inference that can be made about the passkey due to the change in the message before and after the passkey is introduced. Consider two registers, say R0 and R1. The hamming distance between the two registers is equivalent to the XOR of the two registers. If R0 contains the message before the passkey is introduces and R1 contains the message post introduction, the hamming distance between the two will become the sensitive data in question. Below is the diagram of the DES implementation.

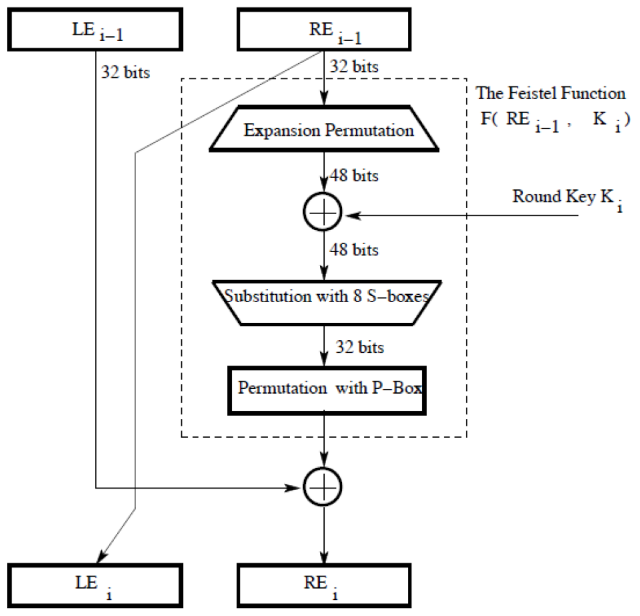

Figure 1. One round for a DES implementation

DPAcontest.org provided a simple DES implementation using matlab. This was used to ease in plotting matrices of data and the quick response to parse through large amounts of data. A modified script was created to use this DES file such that the following results were produced:

1) The hamming distance between the message before and after the passkey is introduced. This is the sensitive data that corresponds to an expected power usage for some given passkey. If solving for a single Sbox, this matrix will contain a column entry for each sub-key guess corresponding to each tracefile message.

2) The actual power usage for a given message and a given, but theoretically unknown, passkey.

3) Organization for the data from these matrices in a useful and productive manner such that the correlation coefficient can be easily produced.

With this information available, a plot can be produced to visually show correlation of the actual power being used corresponding to the expected power from a sub-key guess.

A. *Optimization I: Selective Input Messages*

For this optimization, consider the xor operation. Specifically, what must be done for the xor operation to return a maximum value corresponding to a maximum hamming weight? The answer is simple: either one register begins with all bits high and the other with all low or the first is all low and the second is high. Consider if the message were to be all high bits and the passkey were zero. The xor between the pre-message and the post-message with respect to the key being introduced would be zero, due to no change being introduced by the passkey. Hence, it is simple to see that the maximum effect returned from the hamming weight would be found if the message began at all zeros and the passkey were all ones. By sorting the messages by the least number of 1-bits would produce a faster result when guessing with keys of a high number of 1-bits.

B. *Optimization II: Area under the spike*

Another possible optimization is to calculate the area under each spike. The highest area under the spike would give the biggest power trace due to the peak being high. The process could give a false positive due to a short wide spike but the probability that a higher peak having a higher area is better. This process would be more efficient on a high performance computer because calculating the area might involve integration or a complex calculation but it would be slow on a regular computer. Another thing to note is that the area would have to be calculated twice, one for the positive y-axis and a second for the negative y-axis. The cutoff for the axis would be at the y=0 correlation point due to the correlation being centered there. The method would be affective but at the cost of high resource usage.

III. Results

A correlation of the key was found by using the correlation coefficient obtained from 64 key guesses with the power trace. The code was ran for 250 traces for different Sboxes. Using only a portion of the traces the code was able to run more efficiently. Below are some graphs of the correlation of the key guesses.

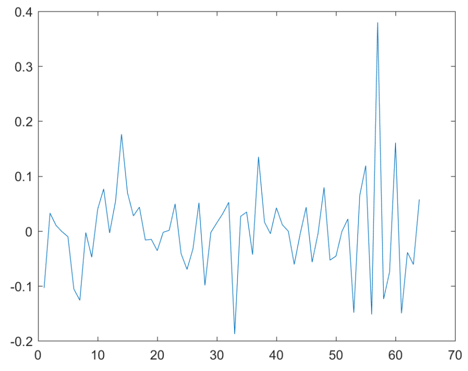

Figure 2. Correlation of Key guess for 250 Traces of the first Sbox.

The figure above shows that for 64 key guesses the right key was key guess 58. This key guess was correlated to the power usage from round 1 using one Sbox.

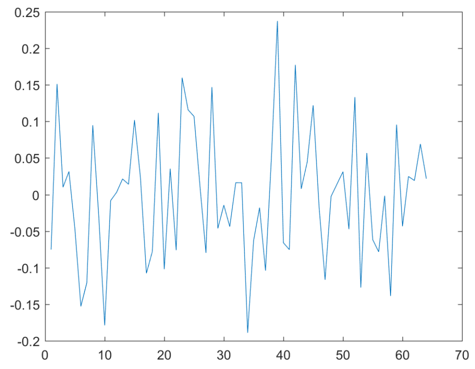

Figure 3. Correlation of Key guess for 250 Traces of the fourth Sbox

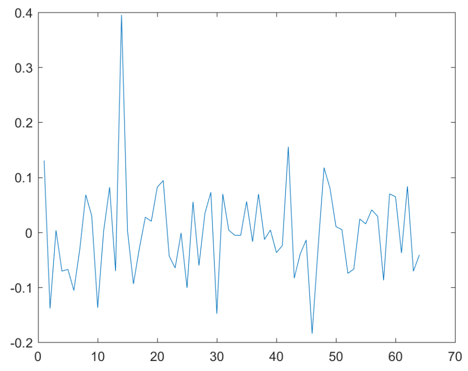

Figure 4. Correlation of Key guess for 250 Traces of the sixth Sbox

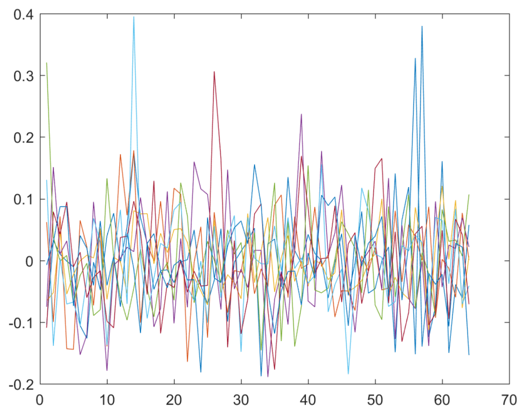

Figure 5. Correlation of Key guess for 250 Traces of the all Sboxes

The figure above shows all Sboxes with every possible guess. The right keys are the 8 highest peaks that correspond to the highest correlation with power usage.

IV. Conclusion

The program written in Matlab generated the 56-bit key. This 56-bit key is actual key before the parity bits are added. Due to time constraints the last 8-bits where not found. Using a brute force attack on the last bits would give the full key. No optimizations where implemented but the only way to optimize the algorithm is to use sorted messages that give the biggest change in Hamming Distance in the first key guesses or to find the minimum number of power traces needed to find the right correlation values. The hardest part of the project was to understand how the Hamming Distance needed to be calculated. The DES implementation ran for 64 guess for each 8 S-Boxes so the Hamming Distance was calculated by counting only four bits of the message for each guess, this process was difficult to very given that there are numerous bits to keep track of.

Another thing noted was that it is relatively easy to find the cypher-key given the power traces and messages. In an actual scenario the power traces would be very difficult to obtain and they cypher-key would almost be impossible to find. DES algorithms are still have a potential to be broken but it is difficult with a CPA attack.