# Project III: Side-Channel Analysis Demonstration of a DPA attack

Kurt Brenning
Email: burenning@gmail.com

Seth Andrews
Email: sethdandrews@gmail.com

***Abstract*—The DES encryption algorithm has been used in the past in order to improve system security. However, this encryption is no longer secure, and there are many known exploits which circumvent this encryption. We demonstrate one such-method using DPA side-channel analysis in order to extract the encryption key. This attack is demonstrated using power traces provided on dpacontest.org. This attack exploits detectable differences in power consumption to expose the encryption key.**

## I. Summary of Assignment and Requirements

This project is written for the fulfillment of PROJECT III: SIDE-CHANNEL ANALYSIS OF AN EMBEDDED CRYPTO DEVICE in ECE 5760 at Utah State University. The stated goal of this project is the following. "Using data from the DPA contest [1], implement a side-channel attack against a custom DES crypto-processor using a technique/method which has not been covered in class". This assignment is graded on the following criteria:

- Number of traces required to get key (sound method to measure this)
- successful recovery of key,
- novelty of improvement: does it make sense/why is it effective; effort and understanding,
- extra points: you gave me something unexpected (discretionary)

## II. Introduction

In order to fullfill the above requirements, we chose to implement a DPA-style attack on the secmatv3_20070924_des dataset provided. The DPA attack implemented relies on the example presented in Lecture 15 [2]. Specifically, this attack targets the xor computation in the last round of the DES algorithm. The attack uses the known ciphertext coupled with guessing 6-bits of the 16th round sub-key to separate the traces into two groups. One group represents cases in which the XOR gate receives a 0 at one input, while the other group represents cases in which that input is a 1. This input is referred to as the Difference bit, or D-bit. The mathematical difference between these two groups reveals enough information about the encryption key to allow for our attack to estimate the remainder of the key.

## III. Differential Power Analysis

"The DPA selection function D(C b Ks ) is defined as computing the value of bit 0 for b is less than 32 of the DES intermediate L at the beginning of the 16th round for ciphertext C , where the 6 key bits entering the S box corresponding to bit b are represented by $0<Ks<2^6$. Note that if Ks is incorrect, evaluating D(C b Ks ) will yield the correct value for bit b with probability P is small for each ciphertext." [3]

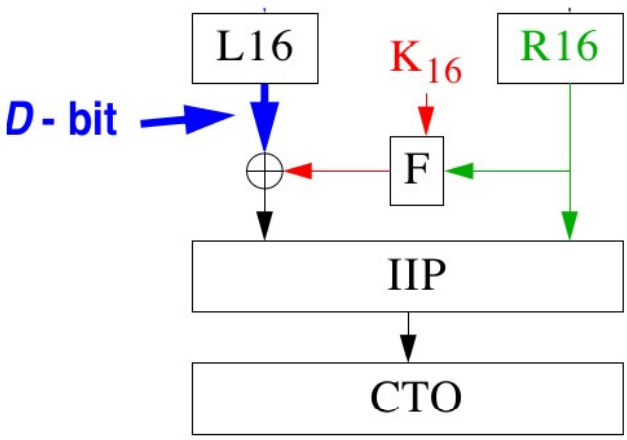


Fig. 1. Graphic explaining location of Dbit taken from lecture slides [2]

### A. DPA

Our DPA attack is organized into sequential steps as it is implemented in software.

*1) Calculate D-bit values:* Based on the ciphertext input and a key guess, we calculate each d-bit. The location of the d-bit is shown in Figure 1 obtained from the lecture slides. This explains that the d-bit is actually the left input to the xor gate. Because the ciphertext and key guess are known values, the other input and the output of the xor gate can be calculated. Our attack calculates the entire 32-bit input to the xor gate, using the same 6-bit key guess for all s-box inputs.

*2) Separate, average, and compute difference:* All 32 d-bits have been computed for all input traces. For each d-bit, the traces are separated into 2 groups: The traces in which that d-bit is 1, and the traces in which that d-bit is 0. Each group is averaged, and the 2 averages are subtracted from each other. This produces a single differential trace for each d-bit. These 32 differential traces are separated into 8 broups, based on which s-box output they are related to. Each group is added together to produce one aggregate differential trace for each s-box. Analyzing these traces allow for accurate identification of correct key guesses. This is an improvement over traditional

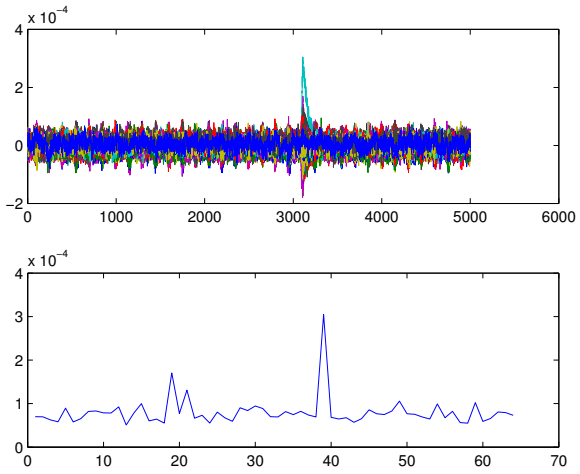


Fig. 2. Traces and maximums obtained using a single d-bit for each s-box (70,000 traces)

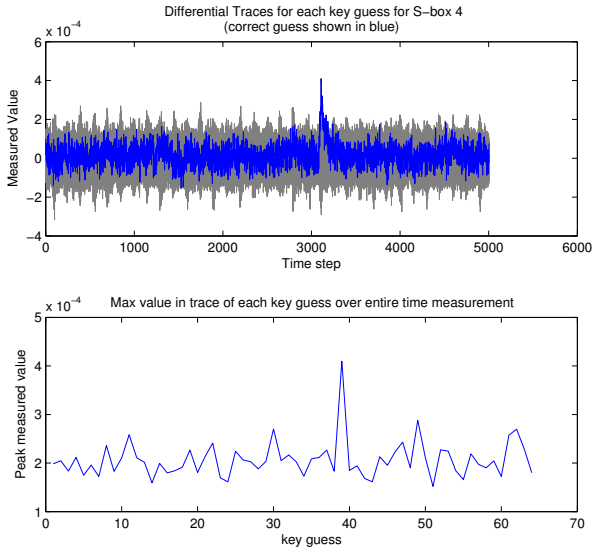


Fig. 3. Traces and maximums obtained using the summed results of 4 separate d-bit differentials for each s-box (3900 traces)

DPA where only 1 bit is considered.

*3) Compile key-guesses for all s-boxes:* This method produces 64 traces for each s-box (1 for each key guess). The 6-bit key guess for this s-box is selected to be the one which produced a trace with the maximum value for all traces. In this manner, we generate 8 6-bit key guesses which combine to form the sub-key for the final round of DES.

*4) Create subset of possible keys:* This sub-key is only 48-bits of the 64-bit key. 8-bits of the key are parity bits and can be calculated directly. It is necessary to use a brute-force method in order to obtain the other 8-bits of the encryption key. We therefore generate 256 possible 64-bit encryption keys based on the 48-bit subkey obtained from the d-bit computation.

*5) Test subset to identify correct key:* This subset of 256 keys are each tested by using a known plaintext ciphertext combination. We encrypt the plaintext into the ciphertext using each of the 256 encryption keys. We then check to see which, if any of the ciphertexts are correct. If we identify a key which produces a known ciphertext given a known plaintext, then we have successfully implemented our attack and the encryption key is compromised.

### B. Discussion of Optimization

By combining the 4 differential computations, as opposed to relying on only a single d-bit computation, a result can be obtained using far fewer traces than otherwise would be required. Figures 2 and 3 show this result. Using only a single d-bit, even including 70,000 traces for analysis, still resulted in a detectable maximum peak which was only 2x above the noise. In reality, to obtain the correct key required 16,000 traces in this baseline case. Adding this obtimization reduced the necessary traces to 4000. This is a significant improvement, and is easily implemented, as it does not require any further collection of data, just a small amount of additional computation.

## IV. Methodology

### A. Power Traces

The power traces utilized to demonstrate this DPA attack were obtained from the DPA contest website [1]. Specifically, we use the secmatv3_20070924_des dataset. These traces were taken from an unprotected DES crypto-processor implemented in ASIC. They were pre-formatted to be temporally aligned and to include the entirety of the DES encryption. There are a total of 81569 traces, each with 5003 single-precision floating point data points. All the traces used the same 64-bit encryption key (0x6B64796B64796B64), but each trace used a different plaintext to be encrypted. The plaintext, resulting ciphertext, and encryption key are provided with the data. The data is provided in a .bin format. In order to analyze the data in Matlab, we used the provided scripts to parse the provided data into a .csv file. We further converted the data into a matlab object with three elements. In this data, a new row represents a separate trace, while a new column represents a separate time-step ordered sequentially. We store both the plaintext and ciphertext for the traces in column-vectors with one element per trace. The trace data is converted into a 2-dimensional array with each trace stored on a separate row. This dataset requires a substantial number of traces to perform DPA, which caused issues both with testing the algorithm and developing it, due to the large running time needed to analyze all traces.

### B. Project Requirements

The following explains exactly how we meet the requirements for this project as set forth above.

*1) Identifying number of traces required to get key:* In order to quantify this number, we used a binary search algorithm to identify the minimum number of traces required to obtain the correct key result. After obtaining this minimum number (3982 traces), we re-ran our algorithm using a different set of 4000 traces until we had included all traces in our analysis at least once. We therefore verified that this is an accurate measurement for our algorithm.

*2) Verifying successful recovery of key:* In order to verify successful recovery of key, we used the given plaintext and ciphertext for a single trace, along with our key guess. Using the plaintext and the key guess, we computed the ciphertext through the DES algorithm. If the encrypted plaintext using our key guess produced the expected ciphertext, we conclude that our key guess was correct.

*3) novelty of improvement:* Why is our improvement effective? It leverages existing computation that has already completed in order to improve the resulting estimation. For a standard DPA attack, any one of the four d-bits could be selected. We simply chose to select all of them. We chose not to consider all of the d-bits simultaneously. In that case, the majority of the analyzed traces would have to be ignored. By effectively performing 4 separate attacks, then combining the results, we further increase the signal to noise ratio on the desired signal, while eliminating noise factors that may only be present in one of the 4 d-bit differentials.

## V. Conclusion

Using the traces provided on the dpacontest's website, we were able to successfully compromise the DES encryption algorithm using DPA. Using a baseline single-bit per s-box attack, our attack required almost 16000 traces to produce the correct encryption key. Summing 4 d-bit differentials for each s-box allowed us to reduce the required traces to 3982, a 4x improvement.

## References


[1] [Online]. Available: http://www.dpacontest.org

[2] [Online]. Available: https://spaces.usu.edu/display/usuece5930hpsec/Home

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in CryptologyCRYPTO99*. Springer, 1999, pp. 388–397.