ECE 5930 Report: Side Channel Analysis – CPA Attack

Seth Richter and Tahmoures Shabnian
Deptartment of Electrical and Computer Engineering
Utah State University
Logan, Utah 84322
e-mail: sethrrichter@gmail.com, tahmoures1987@gmail.com

***Abstract*—Side Channel Analysis (SCA) attacks can be used to uncover a cryptographic key by analyzing the power traces of a cryptographic device. The correlation power analysis (CPA) SCA attack correlates the hamming distance of register writes (in this case, during the first round between the original input and input to the SBOX's) to the power used at that point in the trace. We implemented a CPA SCA in Matlab that was able to successfully find the key, then improved on this base implementation with two optimizations to minimize the total number of power traces required to find the key.**

I. Introduction

DES is an encryption algorithm which iterates through 16 rounds, each with a 48-bit round key. Each round key is broken into 8 different 6-bit subkeys, which are xor-ed with a portion of the message. The hamming distance of the message before and after the xor-ing is correlated a point on the power trace of the device used, which makes DES particularly susceptible to side channel analysis (SCA) attacks. We use the correlation power analysis (CPA) SCA attack to determine the round key of the first round of the DES algorithm by correlating all possible guesses of each 6-bit subkey with a point on the power trace. After obtaining the round key, it is trivial to guess the remaining 8-bits to obtain the original DES key. When performing this attack, the attacker attempts to minimize the number of traces needed to obtain the key. With this in mind, we have implemented two optimizations on the base CPA attack.

A. Related work

[1] is the original paper on SCA attacks, and was used to gain an overall understanding of SCA attacks and how they work. The class slides were used to aid in our understanding of CPA and DES [2]. [3] and the DPA contest website [4] were the original sources of our optimizations. We used the traces from the SecmatV1 SoC cryptoprocessor, running DES from the 2006 competition.

B. Structure of paper

Section II describes our base CPA implementation. Section III describes our optimizations to the base CPA implementation. Section IV gives the conclusion.

II. Original CPA Implementation

The first step of our CPA implementation was to find the point in the power traces which correlates to the hamming distance of the register analyzed. This was easily accomplished by computing the hamming distance of the right half of the message when using the correct key. After computing the hamming distances for 100 traces, the traces and hamming distances were correlated together1. The location of the point of maximum correlation between the traces and hamming distances corresponds to the point in the power trace to use for the rest of the CPA attack. This point was the 5744th time-step of data in our power traces.

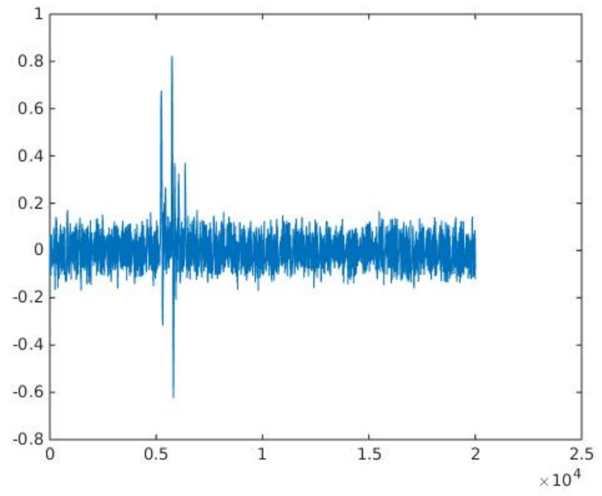

Figure 1: The point of the maximum value is the timestep correlated with the register write.

Our original CPA implementation accepts the power traces in the order given. For each trace, the program calculates the hamming distance for each of the possible 8 6-bit round one subkeys corresponding to the original message of the power trace. The program then correlates these hamming distances with the power trace corresponding to that message, deciding on the "correct" subkeys. The program repeats these steps for each new power trace, recomputing the "correct" subkeys based the correlation of all previous hamming distances to previous traces.

Table I: Average traces required over 8 runs of 10000 different traces each run.

Implementation	Number of Traces (includes extra 100)	Standard Deviation
Base CPA	379.375	158.56
Optimization 1	286.5	30.28
Optimization 2	125.75	6.94
Both Optimizations	137.0	8.90

This is repeated until the "correct" subkeys has stabilized for 100 iterations, resulting in the actual correct subkeys2. These subkeys can then be used to construct the original DES key. Our CPA implementation was able to determine the correct subkeys after an average of 379.4 iterations, as seen in I (279.4 iterations to find correct subkeys, then 100 iterations with no change).

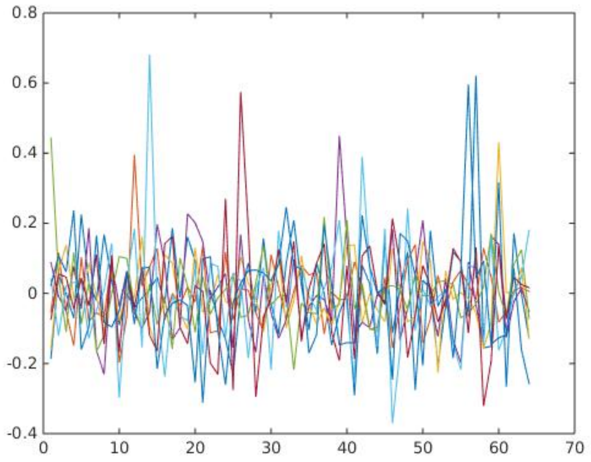

Figure 2: The x-value of the 8 maximums correspond to the 8 subkey values of DES round 1.

Our program calculated the hamming distance of each message by passing the message to the given DES implementation and stopping the program early in the first round, returning the calculated hamming distances for each of the possible values of the subkeys (2^6 possible values per subkey).

III. Optimizations

We implemented two different optimizations beyond our base CPA implementation. The number of traces required to recover the key can be seen in I.

A. *Optimization 1*

Our first optimization's goal is to increase the correlation between subkey value's hamming distances and the power traces. Rather than compute the hamming distances of each subkey's possible values individually (2^6 possible values and hamming distances), we instead made the length of each subkey 12 (2^{12} possible values and hamming distances). This increased the computational time required to compute hamming distances, but reduced the total number of traces required to find the correct subkeys. This occurs because each longer subkey has a more unique hamming distance compared to the hamming distances of multiple subkeys, which in turn results in a higher correlation between the power trace and hamming distances. With this optimization, we were able to determine the correct subkey within an average of 286.5 traces (186.5 iterations to find the correct subkey, then 100 iterations with no change)I, a 24.5% increase over the standard CPA implementation3. While this optimization decreases the required number of traces, it significantly increases the time required to compute hamming distances of 2^{12} possible subkey values. However, the time required to successfully perform this attack was still under two hours, and the limiting factor of SCAs is the number of traces required to determine the key, not the time required to analyze obtained traces. The idea for this optimization came originally from submission #6 under the fixed order submissions in the DPA 2008/2009 contest Hall of Fame [4].

B. *Optimization 2*

This optimization involves selecting the order of traces to use for the correlating. The order of traces used involves alternating each iteration between selecting the trace with the highest value at the point correlated to hamming distance of the register and selecting the trace with the lowest value at the same point. This idea originally came from [3], which is a paper specifying improvements on the DPA algorithm. Our basis for using this optimization with the CPA attack comes from the idea of eliminating the effect of noise in the power trace. Noise in the power trace may cause the power used to fluctuate up and down from the actual value correlated to the hamming distance. Assuming, for example, that 50% of the time the noise is within the range that the trace correlates most closely with the correct hamming distance, 25% of the time the noise is a large positive value which causes the trace to correlate most closely with the correct hamming distance plus 1, and 25% of the time the noise is a large negative value which causes the trace to correlate most closely with the correct hamming distance minus 1. In this instance, 50% of the time the noise will cause an improper correlation. However, if the trace is very large at the correlating point, the correct correlation is with the largest possible hamming distance, meaning that 25% of the time, when there is a large positive noise, the trace will still correlate with the largest possible hamming distance, reducing the effect of noise from 50% of the time correlating to an incorrect hamming distance to doing so only 25% of the time. However, when we implemented this using only the largest traces, we found the number of traces required to be significantly larger than that required when using random traces. We attribute this to the fact that having little variation between traces made it difficult to effectively correlate the similar traces to different hamming distances. This is solved by alternating between the largest and smallest traces, causing there to be a large difference between traces and allowing for accurate correlation to hamming distances.

We implemented this optimization by loading 10000 traces

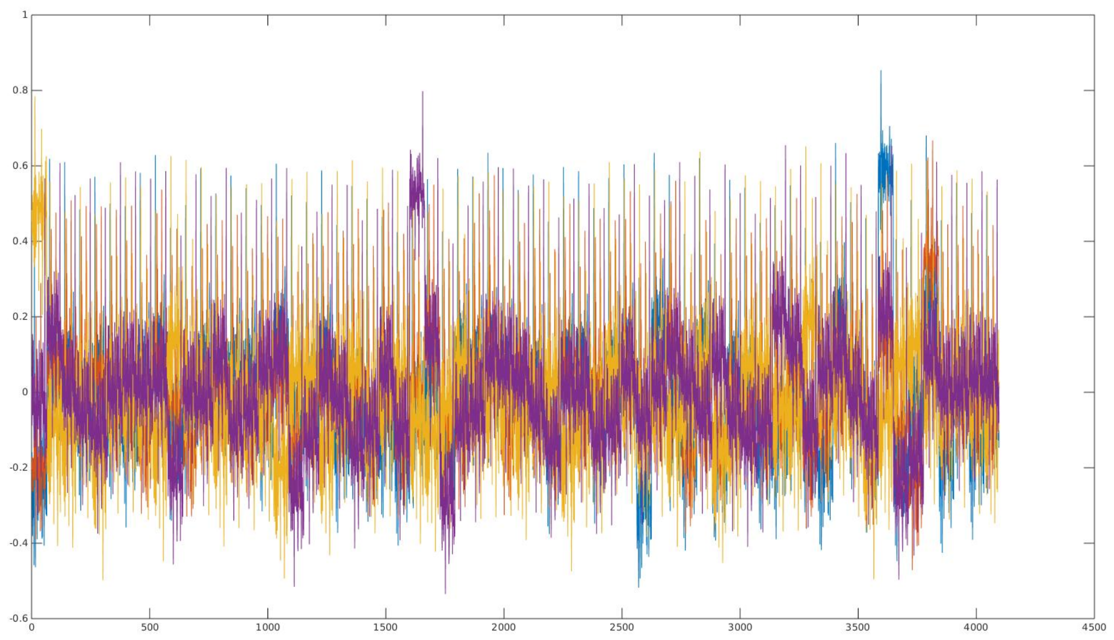

Figure 3: Instead of having 8 maximums corresponding to values ranging from 1-64, there are now 4 maximums corresponding to values ranging from 1-4096, in this example at x-values 13, 1655, 3595, and 3814.

into Matlab, then alternating each iteration of CPA between the largest and smallest of these traces at the point correlated earlier to the hamming distance. We saw much greater improvement than what we originally expected from this optimization – we were able to find the correct subkeys in only and average of 125.8 iterations of CPA (25.8 iterations to find the subkeys, then 100 iterations with no change)I. This corresponds to a 66.9% improvement compared to the original CPA implementation.

IV. Conclusion

Using the base CPA implementation, we were able to correctly recover the key. Through the two optimizations, we were able to significantly reduce the number of traces required to find the key. We would have expected both optimizations, when implemented simultaneously, to be more effective than the optimizations implemented individually. However, this is not the case (137.0 average traces required for both implementations). We would hypothesize that the reason for this is because, although optimization 1 would eventually increase the correlation between hamming distances and subkeys, since optimization 2 results in such minimal number of traces to find the key, not enough traces have been analyzed for the increased correlation to speed up the findings. Instead, optimization 1, for a small number of traces, will result in each hamming distance being closer together (since each subkey has a max hamming distance of 12 instead of 6) relative to the power used in the trace. We believe that the hamming distances' closeness will eventually be offset by the increased correlation, but just does not happen early enough to provide increase when combining optimizations 1 and 2 since the number of traces required for optimization 1 is already close to the minimal number required for a perfect implementation (125.8 as compared to 101). Additionally, we found that there were very large standard deviations in the average number of traces required for the implementations, particularly for the base CPA implementation (standard deviation of 158.6), which occurs due to getting "unlucky" in the traces analyzed, resulting in significantly more traces being required before converging to a key. This was lowered when using optimization 2 by choosing only "good" traces to observe and analyzing the "unlucky" traces last.

References

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in CryptologyâĂŤCRYPTOâĂŹ99*. Springer, 1999, pp. 388–397.
[2] "Usu ece 5930: Hardware and physical security," https://spaces.usu.edu/display/usuece5930hpsec/Home, accessed: 2015-03-23.
[3] V. Lomné, A. Dehbaoui, P. Maurine, L. Torres, and M. Robert, "Differential power analysis enhancement with statistical preprocessing," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2010, pp. 1301–1304.
[4] "Dpa contest website," www.dpacontest.org, accessed: 2015-03-23.