Side-Channel Attack on an Embedded System

Ali A. Al-Hashimi, David Petrizze, Michael Schena
Deptartment of Electrical and Computer Engineering
Utah Stat University
Logan, Utah 84322
e-mail: ali.2014@aggiemail.usu.edu, dpetrizze@gmail.com, michael.schena@aggiemail.usu.edu

***Abstract*—This paper describes a side-channel attack implemented on a crypto-processor running a DES encryption algorithm. The objective of this attack is to extract the encryption key, used by the DES algorithm, using power traces measured from the crypto-device through running the DES, as well as Differential Power Analysis (DPA) technique. Additionally, the attack was modified and compared with a baseline DPA results.**

I. Introduction

DES (Standard Encryption Standard) basically receives an input text and produces a cipher text. The whole process depends mainly on the encryption key. The extraction of that key is the goal of the mounted side-channel attack. Each crypto-processor will have a certain side-channel, such as power or electromagnetic emissions, while it is performing the encryption process. Side-channel analysis exploits that inherent property and performs different types of attacks based on the type of the side-channel. In this paper, DPA was chosen to mount the attack. The input message(s) as well as the corresponding cipher text(s) were also needed. Power traces were taken from the DPA Contest website along with the other, already mentioned, requirements.

A. Related work

In order to understand the DPA attack, Contre-mesures geometriques [1] was referred to for the power model of an XOR gate and the register-hamming-distance based DPA implementation. The power traces used were taken from the version one DPA Contest [2].

B. Structure of paper

Section I gives a general introduction about the DPA attack. Section II describes the software implementation of the side-channel attack and how the power traces were used to execute it along with the results. In addition, this section explains the modifications made in order to optimize the attack. Finally, section III draws a general conclusion.

II. Software implementation

MATLAB functions were used to implement two methods, baseline and improved, of DPA style attacks, utilizing the DPA Contest v1 power traces. For each possible subkey, the function would iterate over a given number of traces and partition them into two partitions. For each partition, the power traces would then be averaged, and the two averages were differenced. The differences obtained for each key were then compared to choose a key guess. The two methods were then compared based on which found the correct key with the fewest power traces.

To partition the power traces two different power models were used for the two different methods. For the baseline method, the hamming weight of the output of an exclusive or function in the first round of DES was used, and for the improved method, the hamming distance between the value in the LR register before and after the first round of DES. To determine these values, the first round of DES was simulated for each key guess and power trace message pair. Each trace was partitioned based on the four bits corresponding to the specific SBOX inputs being guessed. For the baseline method, each trace was placed in one partition if its bits' resulting hamming weight was four and another partition if their hamming weight was zero. For the improved method, each trace was partition based on its bits' resulting hamming distance being greater than, less than, or equal to two. When the hamming distance was equal to two, the trace was placed into both partitions, because doing so should reduce any zero centered noise.

After the traces were partitioned, the partitions were averaged and their difference was taken. When the difference was taken the partition of low hamming weight/distance was subtracted from the high hamming weight/distance partition, because our power model this difference should result in a positive spike when the correct key is guessed. These difference traces were the basis for each subkey choice. To reduce computation and increase accuracy, the difference traces were only generated for approximately four hundred samples. For the baseline method these four hundred samples covered the time that was found to correspond to the XOR in question, and for the improved method they covered the write back at the end of round one. The difference trace containing the maximum value was then considered the winner and its corresponding key guess was chosen as the key for the SBOX in question.

III. Results

Our baseline results were unfortunately sub-par. The baseline method implementation, failed to generate any meaningful information within 3000 traces. The differential power traces resulting from our baseline method are shown in Figure 1 Our improved method, however, quickly returned useful data. We attribute the baseline method's inability to generate a significant difference in power to the extremely low change in power drawn by the XOR gate itself. After days of work, we were unwilling to increase our sample set beyond 3000 traces, and implemented

Figure 1: The differential power traces over round one for all 64 key guesses for a single SBOX using the baseline method. (blue is the correct guess, red is all others)

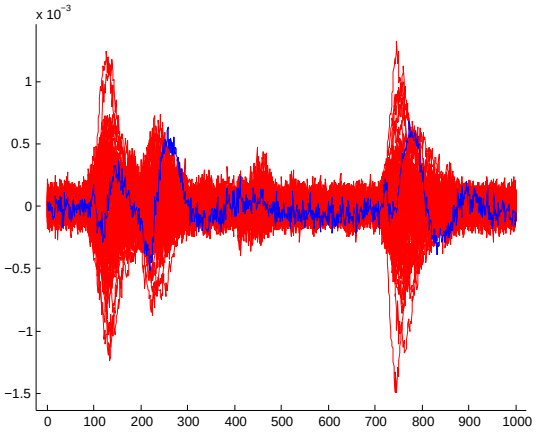
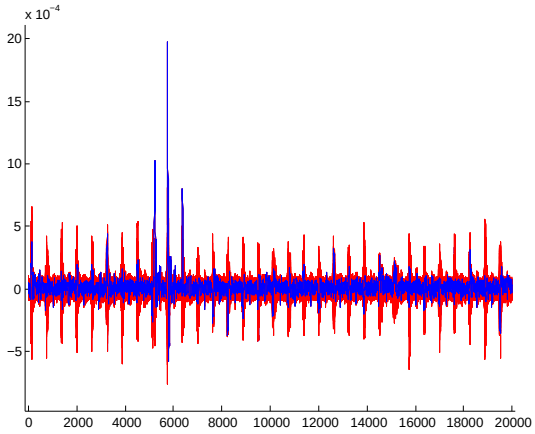


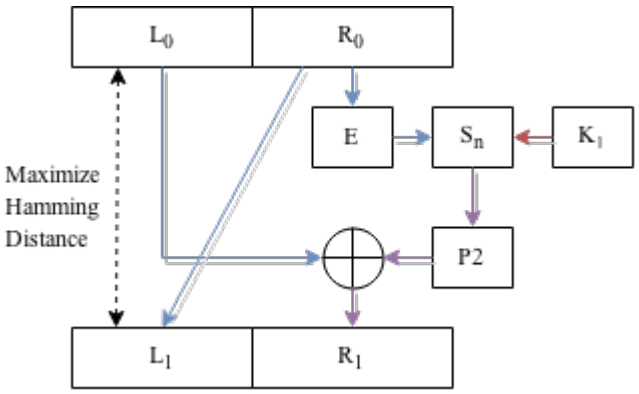
Figure 2: The differential power traces for all 64 key guesses for a single SBOX using our improved method. (blue is the correct guess, red is all others)



the improved method based on the Hamming-Distance power model.

The improved method instantly yielded significant results when we tested it with 2000 traces. At the point the R register was being rewritten, a spike in the differential power trace clearly illustrated a useful partition function. As shown in Figure 2, the height of this peak, shown in blue, was approximately twice that of the peaks for the incorrect key guesses, shown in red. We attribute this to our function correctly predicting high and low power with a higher probability for a correct key guess than for an incorrect key guess. This trend was shown to apply to all S-Boxes, resulting in an entire successful key guess.

Figure 3: The first round of DES. Shown in blue, directly controllable bits. In red, unknown key bits. Purple depicts bits whose values can still be cycled.



Once we had a working model, we tested it with fewer and fewer traces via binary search. We then averaged our results across 50 iterations of the search, given random traces for each iteration. We found the true average of our implementation to be [295.2] traces. While this is high compared to the DPA Contest's hall of fame [2], we felt we did well, compared to an entirely non-working implementation given 3000 traces.

A. *Possible Improvements*

We theorize that feeding our algorithm random traces is less than ideal, and that our algorithm could be improved by selecting traces with messages matching certain criteria. We found that as we reduced the number of traces, and depending on the random subset, one S-Box would continually report the wrong key guess, while the others remained correct.

We believe this was due to a lack of variation in the output of a given S-Box, which, in turn, would fail to generate a significant variation in the Hamming distance between R and R'. In the end, this would cause the algorithm to fail to distinguish the correct key guess from the others, simply because it didn't have much to work with.

This problem can be remedied by ensuring all combinations of the (properly permuted) bits of R are fed into a given S-Box. Additionally, L and the remaining bits of R should be held constant. This will ensure the S-Box produces its full range of outputs, leading to traces with a high correlation to Hamming distances between R and R'. This is illustrated in Figure 3.

Unfortunately, the number of traces proposed here is 8*64 = 512 traces, and our algorithm already guessed the key with fewer. Therefore, rather than using the entire set of combinations, it should be ensured that the critical R-Bits should not be repeated between messages fed to an S-Box. This will maximize variation while remaining within trace constraints.

Additionally, it is possible that the full set of an S-Box's outputs may be produced solely based on R-Bits, despite an unknown key input. This would, in theory, require fewer combinations of the R-Bits. This, we have yet to investigate.

REFERENCES

[1] Sylvain GUILLEY, "Contre-mesures gÂt'eomÂt'etriques aux attaques exploitant les canaux cachÂt'es," https://spaces.usu.edu/display/usuece5930hpsec/sideChannelAnalysis, 2007.

[2] "DPA contest website," www.dpacontest.org/home/, 2015.