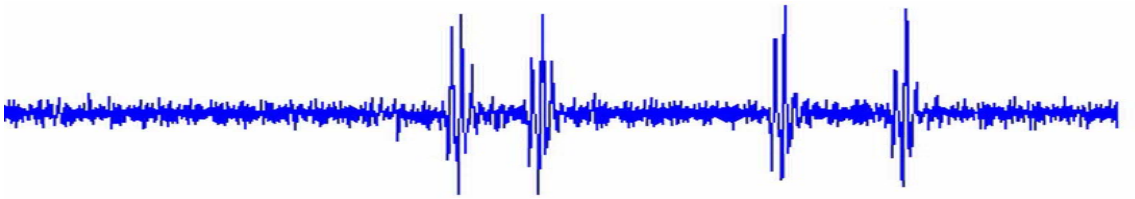
Technische Universität Graz
Institut for Applied Information Processing and Communications – IAIK
Inffeldgasse 16a
A–8010 GRAZ

IAIK
TU Graz

SCA-Lab Technical Report Series



Guided Analysis of WS1

Version 1.0

Elisabeth Oswald

Technical Report
IAIK - TR 2007/07/25

http://www.iaik.tu-graz.ac.at/research/sca-lab/index.php

Guided Analysis of WS1

Elisabeth Oswald

July 26, 2007

1 Getting Ready

In order to follow this exercise, you need to download some stuff. All data that is required for the DPA, such as power measurements and input data is stored in WS1.zip, which can be found at `http:\\www.dpabook.org/onlinematerial/matlabws/`. Download this file, unzip it (it contains a workspace called **WS1.mat**) and store it in a directory of your choice. You also need a script to analyze the data and to plot the result. The analysis script is called demo_dpa.txt. It can be found at `http:\\www.dpabook.org/onlinematerial/matlabscripts/`. After downloading you have to rename the file from demo_dpa.txt to **demo_dpa.m**. There are also scripts for plotting called **show_plots.m** (Matlab) and **octave_plot.m** (Octave). Download them as well (change the file extension from .txt to .m). Make sure that all .m files and the workspace (the .mat) file end up in the same directory.

In order to run the scripts you either use Matlab (which is what we strongly recommend) or you use Octave. Octave is Freeware and you can find binaries for different operating systems at `http:\\www.octave.org`. If you use Octave under Windows it is advisable to avoid empty spaces in directory names. For plotting under Octave with our script, gnuplot must support the multiplot feature. You either have to get the ap

The scripts have been tested under Matlab v.7.0.1 and Octave v2.1.73 under Windows (without cygwin). The exercises are divided into two simple and advanced exercises. For the simple exercise, you essentially only have to call the demo_dpa.m script or have very little changes. For the advanced exercises, you have to make more modifications and you have to put more effort into understanding the results. In order to be able to follow the examples, you have to have read until Chapter 6 of the book. In other words, you have to be familiar with the concept of DPA attacks, including the understanding of the DPA attack as originally described by Kocher et al. and correlation analysis. You should also read through README-WS1.txt to understand what we have measured and you might want to consult the Appendix of the book in order to get a detailed description of the AES implementation under attack.

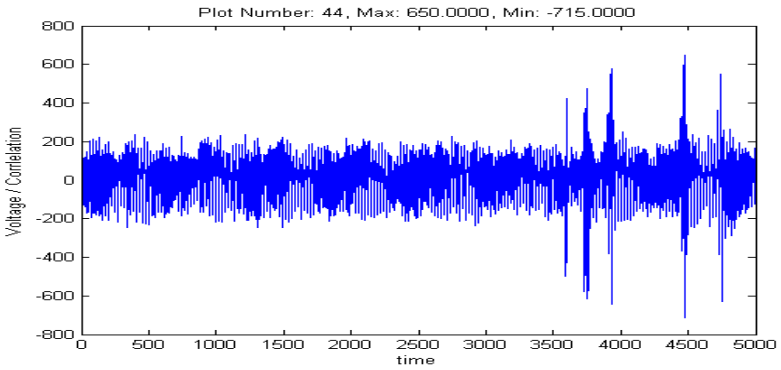


Figure 1: Differential plot of key with decimal value 43

2 Performing a DPA

In the first step, you start either Matlab or Octave and change within Matlab (Octave) into the directory in which WS1.mat and the analysis scripts are located. You can change the directory either via the GUI or by using the **cd** command. For instance, you can type **cd C:\mydirectory**. For Ocatve users: you can check with **pwd** what your current working directory is and with **ls** you can get a directory listing.

When you open demo_dpa.m with some text editor, you can have a look at how the script works. You can also read some help text. By typing **help demo_dpa.m** this help text is displayed in Matlab (Octave).

In the next step, you simply start the demo_dpa.m script by typing:
keys=demo_dpa('WS1.mat',1,256,'kocher');. Make sure that you type **;** as this forces Matlab (Octave) to suppress the output on the screen. The script then performs a DPA attack on the output of the AES SubBytes operation. In the attack, the MSB of the intermediate value is predicted. It is used then, to split the measurements into two sets of which the means are computed and subtracted. The resulting 256 differential traces are returned in the variable **keys**.

In order to determine the key, you have to go through all 256 traces and look which one displays the most significant peaks. You can do that by typing
show_plots(keys, 1,256,4,5000) if you use Matlab and by typing
octave_plot(keys,1,256,2,2) in Octave. These functions open multiple plots at once. You can step through all keys by pressing some button (while you have the focus on the command line window in either Matlab or Octave). The correct key has decimal value 43, hence it corresponds to the differential trace in row 44 of the variable **keys**. The plot of that differential trace should look similar to Figure (the y-scale might be different, the caption is different between Matlab and Octave plotting).

Note that if you run Octave under Windows and gnuplot crashes, then it is better you restart Octave. In order to save the variable **keys** you have to type **save -mat keys.mat keys**. In order to load **keys** later on you have to type **load -mat keys.mat** .If you have problems with that first task (i.e. the scripts don't run) then you have a problem with your

version of Octave and/or gnuplot.

3 Simple Exercises

3.1 Attacking Different Bits—Kocher Method

Open the file demo_dpa.m and look at line 66 which should be
power_consumption = bitget(after_sbox,1);

This line describes the leakage model (power model) that we have defined. It means that we calculate the power consumption of the intermediate values that are stored in the variable **after_sbox** by looking only at bit 1. Try out the other bits as well. Are all of them leaking information?

3.2 Attacking Multiple Bits—Correlation Method

Run the script with the correlation method:
keys=demo_dpa('WS1.mat',1,256,'correlation');

Browse through the result and compare it with the Kocher method. Is it better (is the key easier to detect) or worse? Why is there a difference?

4 Advanced Exercises

4.1 Kocher vs. Correlation Method

Rewrite the script such that the Kocher Method uses several intermediate bits for the classification. Compare your attack with the correlation attack. Which one works better? Which one is easier?

4.2 Attacking AddRoundKey

Rewrite the script such that you can attack the output of the AddRoundKey operation. Attack this operation then with the Kocher method and the correlation method. Use in both cases single and multiple bits. What works, what not, and why?

4.3 Finding Data—Make a Map

Rewrite the script such that you can find the plaintext in the power traces. Produce a map of the measurements, i.e., make a plot that shows where the plaintext byte correlates, the result of AddRoundKey correlates and the result of SubBytes correlates.

5 Answers

Section 3.1 You will observe that the bits 1, 3, and 4 leak information that is detectable with the given 200 measurements. The other bits do not leak enough that 200 measurements can exploit it in this particular attack. We can conclude that different bits in the same register in this microcontroller leak different amounts of information.

Section 3.2 You will observe that it is easier to detect the key because there are less incorrect keys that produce peaks. In addition, the peaks for key 43 are close to ± 1, which indicates that we our predicted power consumption matches the measured power consumption almost perfectly. The reason why this attack works much better is that we use a much better power model. Line 85 shows that we correlate the Hamming weight to the measured power consumption:
power_consumption = byte_Hamming_weight(after_sbox+1);

Chapter 5 of the book points out that this particular microcontroller leaks the Hamming weight of the intermediate values. Consequently, our power model matches perfectly to the measured power consumption.

Section 4.1 You have to change line 66 in order to get out more bits from the intermediate result. Then you have to change line 74 (remove the $== 1$ and $== 0$ and replace it by $> c$ (c is some constant) $< d$). You will observe that if you set $c \neq d$ then you have to discard the measurements that don't fall into the two categories. Hence, the mean values of the distributions will be farther away but because you loose measurements, you have a higher variance.

The correlation coefficient allows us to exploit multiple bits in an efficient manner because we don't need to find a way to split up the traces into two classes. The correlation coefficient is the most natural way to compare the predicted with the measured power consumption. The other advantage of using the correlation coefficient is that we can easily compare the height of the peaks in different differential traces. Hence, the correlation coefficient is easier to apply in this case.

Section 4.2 Instead of working with the variable **after_sbox** you have to work with the SubBytes input (just remove the SubBytes operation in line 52).

You will observe that using the Hamming weight as power model with the correlation coefficient as statistical method gives the best result. Attacking single bits with the Kocher method will also work. Attacking multiple bits with Kocher does not work. This is because the effect of the incorrectly guessed bits cancel the effect of the correctly guessed bits.

Section 4.3 You have to correlate the Hamming weights of the inputs, the result of AddRoundKey and the result of SubBytes to the measurements. Then, you have to figure out how to print all three results in one window. Check the plot function for that.