A toolkit to perform DPA on ARM processors

Nicolas Cointe

27 juillet 2018

1 Introduction

As a critical element of the new GPON networks, ONUs are interesting targets to collect data as they are often let outside of the users' safe infrastructure to ease the maintenance by operators. Breaking their secret keys may allow any other user of the network to observe the communications without any modification of its topology.

This document presents a short overview of a toolkit designed to ease DPA on different types of processors and visualize data. Le first section shows how to break the key on a STM32 and the second section explains different methods to analyze the signal emitted by a raspberry Pi, which is considered as a step between slow microcontrollers and fast SoCs in ONUs.

2 DPA on STM32

Specifications of the component

The platform used is a STM32 VL DISCOVERY board with a STM32F100RBT6 microcontroller with 24 MHz ARM Cortex-M3 core processor, 128 KB flash, 8 KB RAM in LQFP64 package. The board is powered by USB and communicates with a computer through a serial interface.

Acquisition

On the chip, we put a 3 GHz bandwidth probe plugged to an oscilloscope [1] to measure the electromagnetic emission. Then, the computer and the oscilloscope have to be connected together and the IP address of the oscilloscope should be written in the beginning of the script. A file `plaintext.txt` should also exists and contains enough sixteen bytes long messages in ASCII hexadecimal to perform the encryption.

Then, launch the script with :

```
bash DPA.sh STM32 500 LECROY
```

The first parameter indicates the target (`STM32` or `PI`). The second one specify the number of measures to acquire (any integer value between 10 and 99 millions). The last parameters indicates the oscilloscope to use (`PICOSCOPE` or `LECROY`). The user can also add the keyword `onlystore` as a fourth argument to perform an acquisition without trying to break the key. The script will verify the existence (and rebuild if necessary) of the various programs used for the experiment. Then, a loop will repeat the procedure shown on figure 1.

for N in 0 to NumOfMeasures **do**
  Prepare the oscilloscope for acquisition
  Load a plaintext in the chip
  Acquire the data during AES execution
  Load the data from the oscilloscope
end for

FIGURE 1 – Main loop of the acquisition

Breaking the key

If the script `DPA.sh` is launched without the `onlystore` keyword, the script will try to break the key automatically. Otherwise, it is also possible to do it manually with the following command :

```
bash cmd.sh data-STM32 LECROY
```

The result should look like the tabular on figure 2. The most relevant key is visible in the second column and the third one (resp. in decimal and hexadecimal representation). The correlation value for this key is the first value of the fourth column (highest is the best). It must be compared with the average and the variance of the correlations with the other possible keys (moy and var values). The last column indicates the position of the correlation peak.

```
Format LECROY
Byte=0, MSGs=500, SAMPLEs=250000
0  1   01 0.538144/[moy=0.036038 var=0.000745] 124615/22461
1  35  23 0.589772/[moy=0.035207 var=0.000707] 2141115/224111
2  69  45 0.551312/[moy=0.035506 var=0.000714] 2316065/241606
3  103 67 0.529618/[moy=0.035521 var=0.000712] 499625/59962
4  137 89 0.462163/[moy=0.035738 var=0.000721] 624595/72459
5  171 ab 0.571434/[moy=0.036249 var=0.000751] 2174445/227444
6  205 cd 0.548199/[moy=0.036373 var=0.000750] 2391105/249110
7  239 ef 0.524469/[moy=0.035291 var=0.000709] 999545/109954
8  18  12 0.479484/[moy=0.035327 var=0.000710] 1124525/122452
9  52  34 0.534187/[moy=0.035002 var=0.000700] 2207775/230777
10 86  56 0.558173/[moy=0.035720 var=0.000727] 2282765/238276
11 120 78 0.469017/[moy=0.035997 var=0.000733] 1499565/159956
12 154 9a 0.545258/[moy=0.035973 var=0.000741] 1624535/172453
13 188 bc 0.561869/[moy=0.035778 var=0.000734] 2241105/234110
14 222 de 0.532198/[moy=0.035735 var=0.000728] 2357775/245777
15 240 f0 0.558947/[moy=0.035305 var=0.000713] 1999495/209949
```

FIGURE 2 – Result of the attack on a STM32

1. Lecroy Wavesurfer 10M, 1GHz bandwidth, 10 MSa/s

The script also produce a graphical representation of the results as shown in figure 3. The grey lines in the background are a few examples of the signal, where we can distinguish the sixteen steps of the SubByte. The sixteen colorful lines in the foreground represents the evolution of the correlation for each byte of the obtained key.

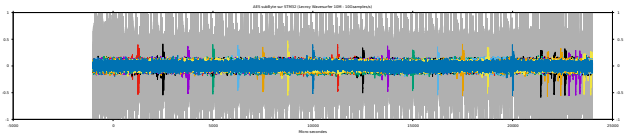

FIGURE 3 – Graphical representation of the correlation on a STM32

With these tools and software, our experimentations were always successful with at least 175 measurements. Data is available in the download section of the author's website to allows you to reproduce these results.

3 DPA on Raspberry 3

To approach the real conditions of breaking the key on an operational ONU, we use in the second part of these experiments a raspberry Pi 3 model B (see section 4 to see a comparison between these CPUs). Due to the specifications of our oscilloscope, the processor has been underclocked at 600MHz and the AES process has been assigned on the CPU number 3 to have more details on the waveform. A lower clock speed implies a higher precision in the acquisition, but a lower electromagnetic emission (and less details on the amplitude of the final measurements), and CPU3 seems to be the easiest to observe. Therefore, this settings are here considered as a good tradeoff between realism and feasibility of the acquisition.

Specifications of the component

The following experiments has been realized on a Raspberry Pi 3 model B+ v1.2, based on a Broadcom BCM2837 Quad-core processor (ARMv8 Cortex-A53), 1.2GHz (down to 600MHz). To reduce the CPU frequency, just add `arm_freq=600` in `/boot/config.txt`. The AES encryption is running on a classic Raspbian operating system. To find the position of the CPU, we used an empirical approach to maximize the visibility of the SubByte's emission. The figure 4 shows the classic sixteen-peaks-pattern of a SubByte. We use GPIOs of the raspberry Pi board to trigger the oscilloscope and start the acquisition.

Acquisition

To launch the acquisition of the data, plus the raspberry on the network, add its IP address in the the script presented in Section 2 and run the acquisition with the dedicated parameter :

```
bash DPA.sh PI 100000 LECROY
```

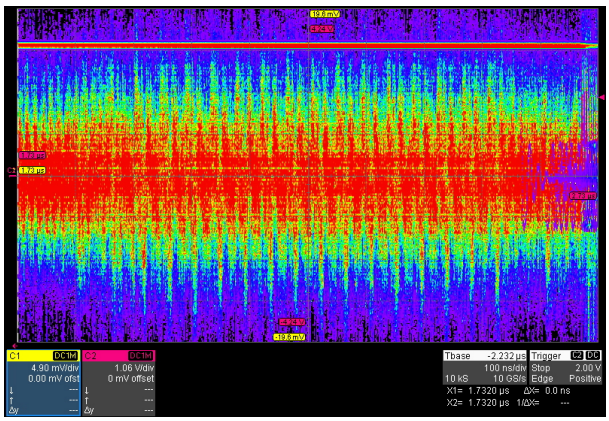

FIGURE 4 – Classic electromagnetic emission of a SubByte observed on the oscilloscope

Preprocessing the data

As it, the data can not be used to break the key with the simple method presented in Section 2. Plotting the result of a naive attack (see Figure 5) on a set of raw measures shows how bad is the matching between the subbytes' electromanetic emmissions.

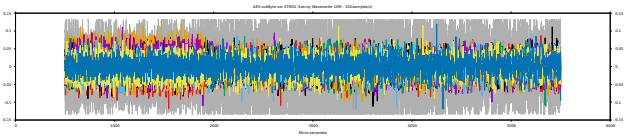

FIGURE 5 – Result of a naive approach

This result is easy to explain due to the time-sharing : the AES encryption might be interrupted by any other process and then generate a time variation in the execution. To observe the impact of this factor, it is possible to see the temporal distribution of the triggers as shown on Figure 6.

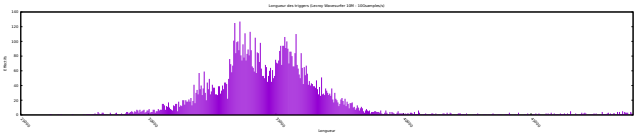

FIGURE 6 – Position of the triggers in the measurement

If the interruptions happens only before the SubByte, the signal should be shifted on the right and it is easy to measure this movement to select only the not-shifted measures or to align the signals. But if the interruptions happens during the SubByte, this operation is insufficient to address the issue. It is then necessary to use a recognition mechanism to detect these events and eliminate (or rectify) the corrupted signals. The QoS (for Quality of Signal) tool provides various options to select the best traces according to some criteria as presented on Figure 7 (you may use the `QoS --help` command for more information).

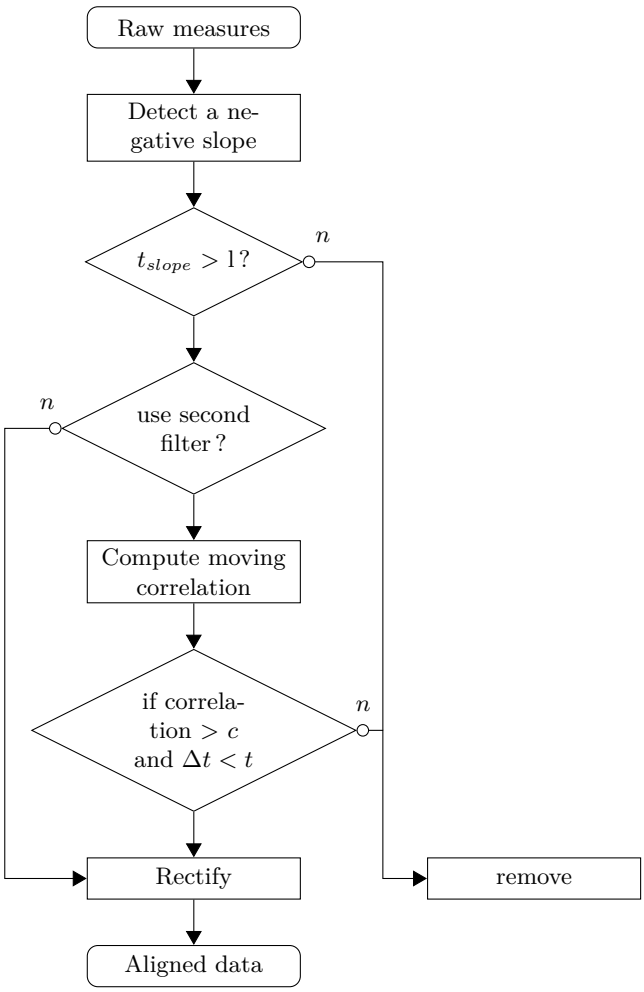

FIGURE 7 – Flowchart representing the filtering of the measurement

Firstly, QoS observes the trigger signal to detect the position t_{slope} of the slope. Then this position is compared with a limit l given by the user to eliminate the signals if they are too much interrupted. A second filter consists in calculating a moving correlation between a part of the signal and the classic sixteen-peaks-pattern. If the correlation is high enough or found at a time t in a window situated before the trigger, the signal is accepted. The last step aligns the signal with the other measurements according to the correlation (or only t_{slope} if the user don't want to use the second filter).

Breaking the key

Even if the filtering and rectification looks like a good progress to select and align the measurements (compare Figure 5 to Figure 8), these operations (and maybe the size of the data set, the precision and bandwidth of the oscilloscope and so on) are not enough to break the key with the method presented in section 2.

They are many options to explore in a future work, such as :

- Simply increase the size of the data set. In the literature, it is common to use data sets of millions messages. The results presented in this document are only based on less than thirty thousands messages after the preprocessing.
- Impeach the interruption of the encryption, to acquire the data in a situation more similar to the STM32 (but also less realistic in comparison with a commercial ONU). *Bare metal programming* on raspberry Pi is an option.
- Find another part of the signal to use as a trigger to synchronize the messages. The call of the function looks like a good option.

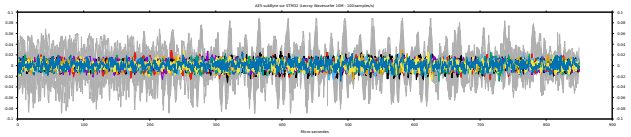

FIGURE 8 – Graphical representation of the correlation on a Raspberry Pi

4 Conclusion

This document shows how to use this toolkit to easily perform a DPA on a STM32 processor with a fast (require only 175 messages) and efficient method. Then we presented how hard it is to deal with interruptions in a more realistic context (on a Linux OS, 600MHz ARMv8 CPU).

Even if this method is not efficient to break the key with this dataset, this work explores many options to improve the quality of the data. This toolkit allows the user to measure and visualize the impact of the interruptions on the acquisition and then select the most relevant signals and rectify their position.