The A to Z of Building a Testbed for Power Analysis Attacks : Documentation

Authors: Hasindu Gamaarachchi, Harsha Ganegoda and Roshan Ragel, Department of Computer Engineering, Faculty of Engineering, University of Peradeniya, 26 Dec 2015

Any bugs, issues or suggestions please email to hasindu2008@live.com

This document explains how to use the tool set for the power analysis testbed. It is highly recommended that you read the paper [Hasindu Gamaarachchi, Harsha Ganegoda and Roshan Ragel, "The A to Z of Building a Testbed for Power Analysis Attacks", 10th IEEE International Conference on Industrial and Information Systems 2015 (ICIIS)] to understand the background of the attack and the testbed.

1. Programming the PIC microcontroller

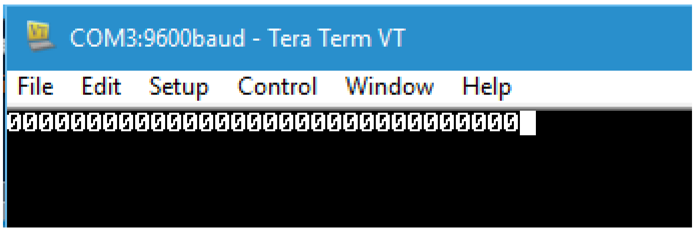
The folder named *1.pic_aes* contains the code which is needed to be programmed into the microcontroller. The encryption algorithm is 128 bit AES. It consists of two folders. *FTDI_USB* contains the files for using the PIC microcontroller with an USB to RS232 TTL converter. *Inbuilt_USB* contains the files for using the internal USB controller of the microcontroller.

When you descend to a folder you would see the CCS PIC C project files. The source code was tested on CCS PIC C version 4. The compiled hex file can be located at *main.hex*. This must be programmed on to a PIC18F2550 PIC microcontroller using an appropriate tool such as PICkit.

If you want to compile on your own, make sure you have CCS PIC C installed. Open *main.pjt* in a text editor and change all the paths there to point to the location in which the files reside on your computer. Then open this file through CCS PIC C. Go through the comments in the code to find the places you need to modify such as, the key, trigger etc. The default key is “0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0xD 0x0E 0x0F 0x10”.

Connect the circuit according to the diagrams given in the paper. Then connect the testbed to a computer via a USB port. If you visit the device manger in Windows, you will see it taken as an unknown device. Hence the appropriate driver must be installed. If you are using an FTDI 232RL based USB to RS232 TTL converter the driver can be downloaded from http://www.ftdichip.com/Drivers/VCP.htm. Otherwise download it from the manufacturer’s website. If you are using the inbuilt USB module of your microcontroller you can find the USB_CDC driver files under the PIC C installation directory (default is C:\Program Files (x86)\PICC\Drivers\NT,2000,XP,VISTA,7). After installing the drivers you will be able to see the device taken as a virtual COM port under device manager.

In order to test the functionality use a serial port terminal such as TeraTerm. Open a connection to the respective virtual COM port. Then type in the plain text in hexadecimal. Plain text is expected to be 128 bits and hence it is 32 hexadecimal digits. For example after entering the plain text “00000000000000000000000000000000 “ it will be echoed back as shown in Figure 1 below.


COM3:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
00000000000000000000000000000000
```

Figure 1 : After entering plain text sample

Now the device is doing encryption. Now it is required to trigger the oscilloscope based on the rectangular pulse coming via the trigger pin (see the paper for more information) of the microcontroller. A screenshot of our oscilloscope after triggering is shown in the figure 2. Yellow trace is the power trace and blue one is the trigger.

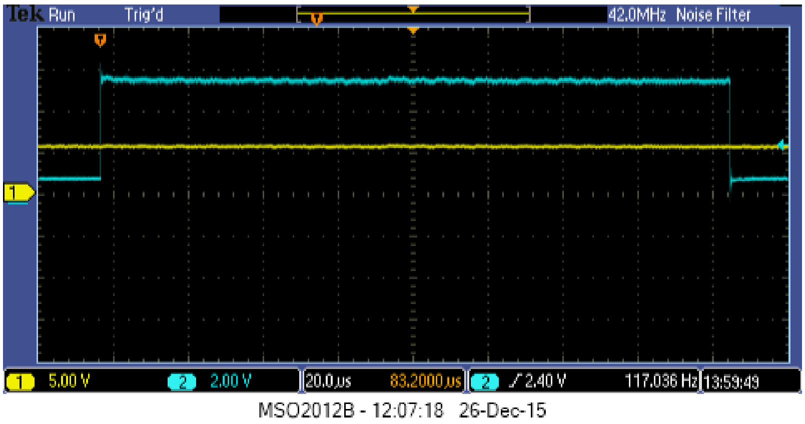
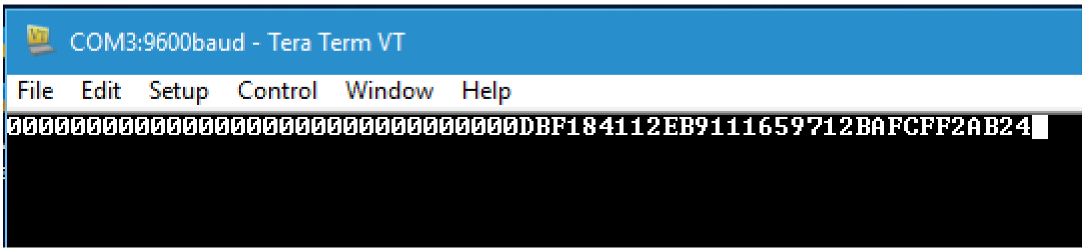


Figure 2: Screenshot of the oscilloscope

Now press any alpha numeric key on to TeraTerm and you will the encrypted text as shown in figure 3. The program for the microcontroller works such that it sits in an infinite loop to accept plain text do the encryption.


COM3:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
00000000000000000000000000000000DBF184112EB9111659712BAFCFF2AB24
```

Figure 3 : After obtaining cipher text

2. Generating plain text samples

In order to launch an attack, multiple power traces are required to be captured and hence multiple plain text samples are required. For 18FPIC2550 about 200 power traces would suffice. The program to generate plain text samples is located in the folder *2.generate_plain_text*. It is a C source code and precompiled versions for Windows is provided as *generate.exe*. *plain.txt* contains an already generated set of 100000 plain text samples. To generate your own plain text *generate.exe <no of plain text samples> <outputfile>* from command line.

It is highly recommended to test the encrypted output coming from the microcontroller to detect any errors. To do that, it is required to calculate the encrypted texts for each plain text sample. On the same folder is another folder called *calculate_encrypt*. The C source codes as well as precompiled binaries are there. Run as *encrypt.exe <no of plain text samples> <plain text file> <output file>*. Already computed encrypted text for the provided *plain.txt* is found in *ciphertest.txt.*
The default key is "0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0xD 0x0E 0x0F 0x10". Make sure you change it in the source code and recompile if you use a different key in the microcontroller.

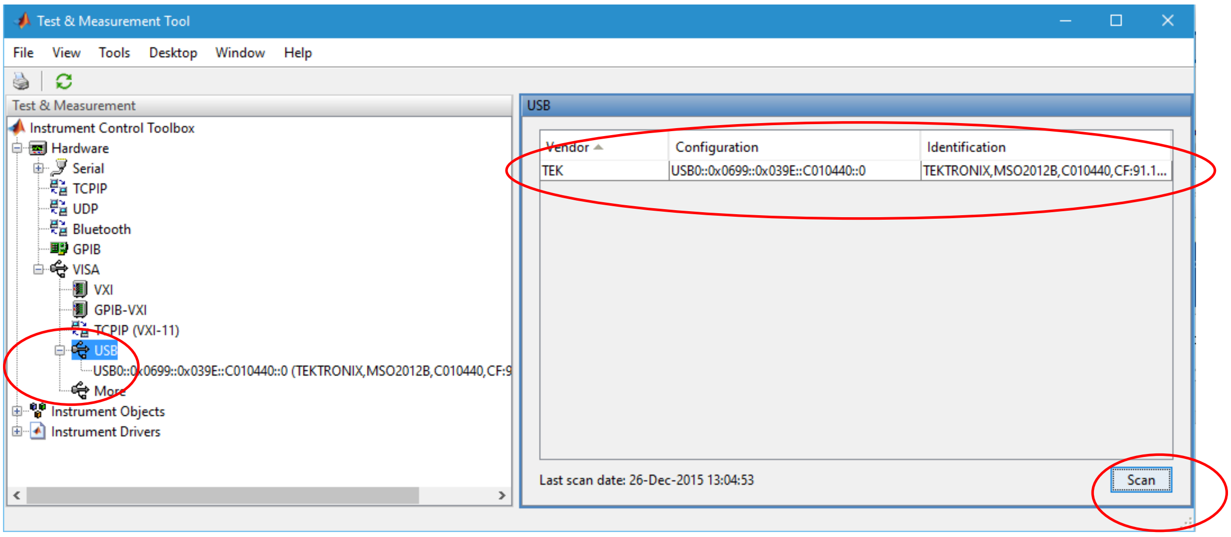
3. Collecting power traces

The provided power collection script is based on Matlab. We have tested it on Matlab 2013a. The folder called *3.collect_power* has the Matlab script in the name *collect.m*. Copy the file containing plain text (and the cipher text as well if you wish to verify the accuracy of the device which is highly recommended). Our sample files are already included there under the names *plain.txt* and *ciphertest.txt.*

Further go through the comments in the Matlab script to find and modify given parameters if required.

Now let's see how to collect power traces. But before power collection, oscilloscope must be connected to the computer and the drivers must be installed. The steps depend on the oscilloscope and hence check the manufacturers website. Then you need to get the oscilloscope detected to Matlab. We will explain how to do that for Tektronix oscilloscope MSO2012B oscilloscope, but the steps must not be that much different for other oscilloscopes.

First launch the *Instrument Control Toolbox* in Matlab. Under *Hardware -> VISA* click on *USB* and then click on *scan* as shown in figure 4 to detect the oscilloscope connected to your computer.

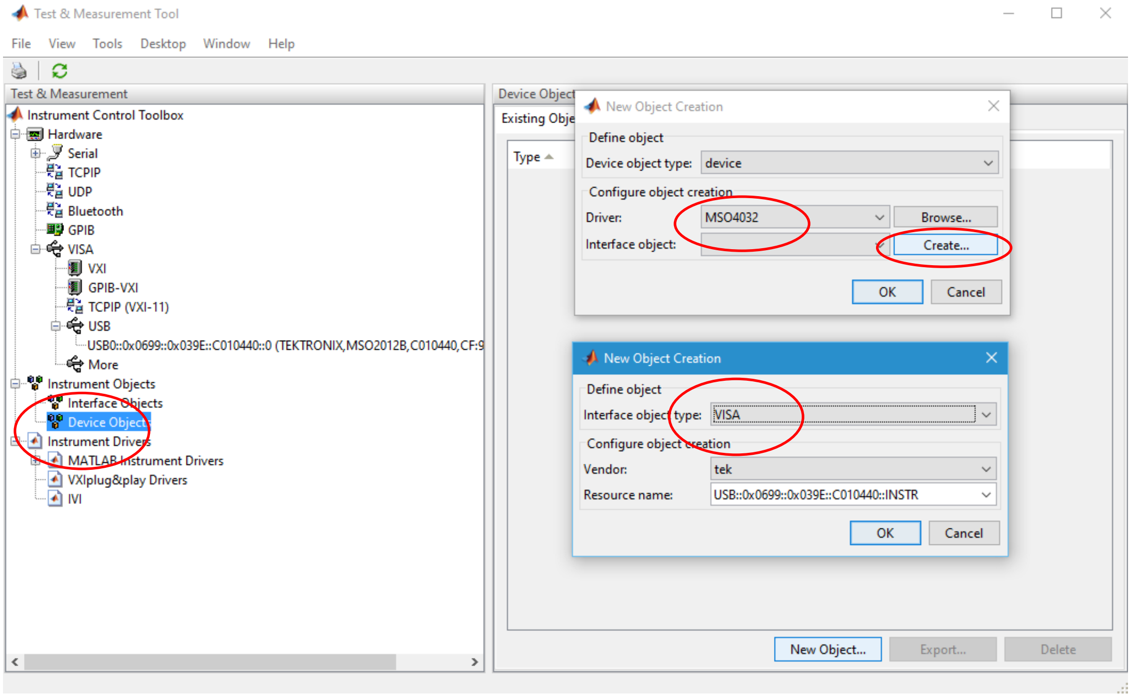


Vendor	Configuration	Identification
TEK	USB0::0x0699::0x039E::C010440::0	TEKTRONIX,MSO2012B,C010440,CF:91.1...

Last scan date: 26-Dec-2015 13:04:53

Figure 4 : Detecting oscilloscope to Matalb

Now it is required to create an Instrument Object. Click on *Device Objects under Instrument Objects* and click on *New Object* as elaborated in figure 5. A new dialog box will open where you will have to select the driver for your device. The default ones given are a few and hence you will have to download the appropriate NATIVE Matlab driver which is a single file having the extension *.mdd*. We couldn't find the exact driver for our oscilloscope but figured out that a driver for a similar oscilloscope such as MSO4032 works. If you cannot find such a working NATIVE driver you will have to create it by using the IVI driver of your oscilloscope using the Matlab command *makemid* as explained in the following article. http://in.mathworks.com/products/instrument/supported/matlab-instrument-drivers.html.



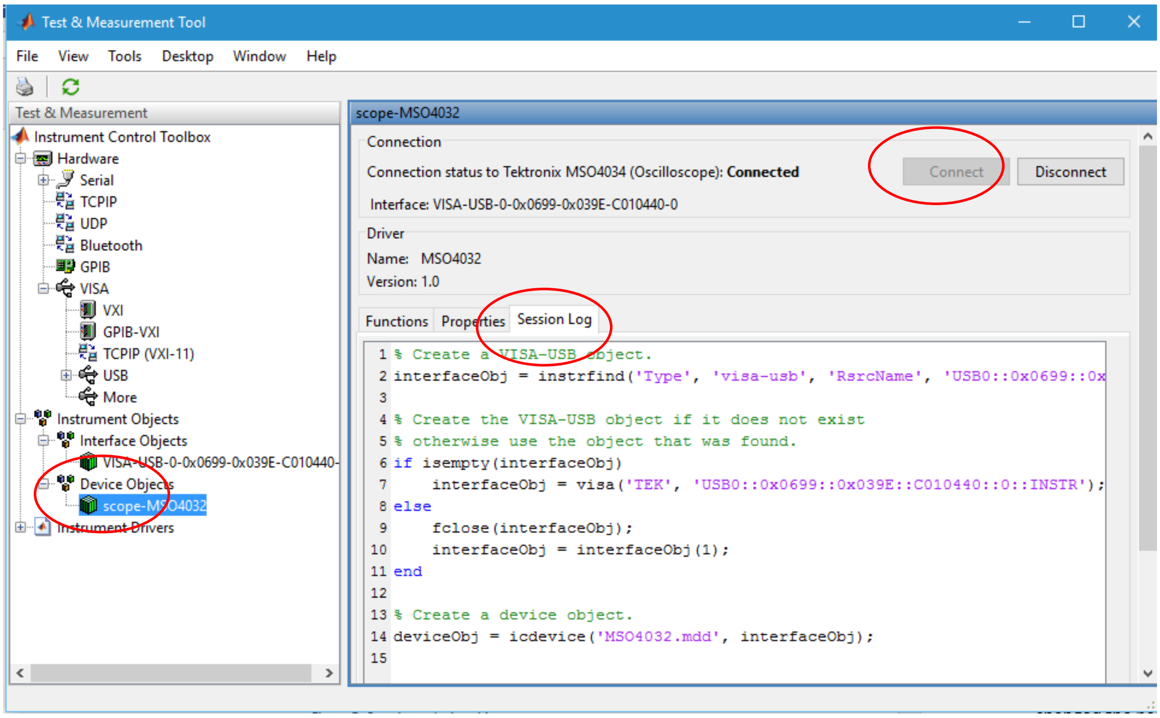
Define object
Device object type: device
Configure object creation
Driver: MSO4032
Interface object:

Define object
Interface object type: VISA
Configure object creation
Vendor: tek
Resource name: USB::0x0699::0x039E::C010440::INSTR

Figure 5: Creating a device object

After selecting the driver file, click *create* under *Interface object* in the same dialog box as shown in figure 5. This will open a new dialog box where it is required to select *VISA* and then select the correct *Resource Name.*

Now click on the created object and click *connect* as shown in figure 6. After it connects click on *Session Log* to view the code to connect to the oscilloscope. Then change the respective part in the *collect.m* oscilloscope according to this code.



Connection

Connection status to Tektronix MSO4034 (Oscilloscope): **Connected**

Interface: VISA-USB-0-0x0699-0x039E-C010440-0

Driver

Name: MSO4032

Version: 1.0

```
% Create a VISA-USB object.
interfaceObj = instrfind('Type', 'visa-usb', 'RsrcName', 'USB0::0x0699::0x

% Create the VISA-USB object if it does not exist
% otherwise use the object that was found.
if isempty(interfaceObj)
    interfaceObj = visa('TEK', 'USB0::0x0699::0x039E::C010440::0::INSTR');
else
    fclose(interfaceObj);
    interfaceObj = interfaceObj(1);
end

% Create a device object.
deviceObj = icdevice('MSO4032.mdd', interfaceObj);

```

Figure 6 : Connecting to the oscillsocope

Now the oscilloscope configuration is finished. But make sure you have the cryptographic device connected to the computer and change the parameter in *collect.m,* such that the virtual com port number in the script matches to the one which the cryptographic device is enumerated in your computer. If your oscilloscope is Tektronix MSO2012B now just running the script will work. But otherwise you may have to replace certain places in the script such as the input buffer size and the function to acquire the trace, with the ones to match your oscilloscope.

The folder named *sample_output* contains the files which are generated after executing a power trace colelction. *wave.dat* is the set of power traces. Power traces are dumped as binary data to save space. But if you like traces to be saved in ASCII there are changeable options in the Matlab script. *stat.txt* gives a summary of the collection such as number of sampling points and time etc. *lastposition.txt* saves how many power traces were successfully acquired. *cipher_device.txt* contains the cipher texts retuned by the microcontroller and will be useful to check if there is an error in encryption. *firstwave.txt* and *firstwave.fig* shows the plot of the first power trace and the trigger acquired. Check this to verify whether Matlab actually gets what is expected. *trigger.txt* is the waveform of the trigger used and will be useful if you later require to extract the points which are exactly under the trigger. The first line there contains the time values of the trigger wave form and the next line contains the voltage values. The file *clean.bat* can be used in Windows to remove all the files which are formed as an output of a power capture.

4. Obtaining the key

After you have obtained the power traces, it is the time for running the Correlation power analysis algorithm located under *4.analysis*. It has two versions, one written using CUDA C to be run on NVIDIA GPUs and one for running on multithreaded CPU written using pthreads library in C. Use *script.sh* to compile and run on Linux while use *script.bat* to compile and run on Windows. But make sure that appropriate libraries (CUDA tool kit / pthread libraries etc.) are already installed on your computer. The script should be launched with 7 arguments in Linux as shown below.

./script.sh <num_of_traces> <power_trace_file> <plain_text_file> <result_file> <power_trace_file_format> <sample_points> <sample_point_partition_size>

In Windows also it is same that except ./scrpt.sh use script.bat. For explanations of the arguments open and read the comments in the script files.

In the folder named *sample_output,* sample files for power traces (*wave.dat*) and plaintext (*plain.txt*) has been included together with the set of output generated after running the algorithm. The answer is in *result.txt.* The first row there is the most correlated key bytes and the second row is the respective correlation coefficients. The next rows show the next most correlated keys and the coefficients up to the fifth maximum. The correlation coefficients for all the key possibilities sorted in key order could be found *all_key_corr.txt* and *all_key_corr_sorted.txt* the same data sorted in correlation coefficient order.

5. Post Analysis

Correlation_vs_num_traces has tools to plot the variation of correlation coeffient with the number of power traces for each key byte. First run the *script.sh* in *1.generate_data* on the power traces you have collected. Unfortunately, we haven't written a script for windows at the moment. Make sure you change the parameters in the script to match yours. More information can be seen as comments in *script.sh.* Place all the files generated of the formal *all_*.txt* in *2.plot/source.* Now change the appropriate parameters in the Matlab script named *plot_graphs.m* under *2.plot* and run it. It will generate 16 graphs per each key byte under the folder *graphs.*

Correlation_vs_time has tools to plot the variation of correlation coefficient with time per each combination of keybyte and keyguess. First run the scrip.sh under *1.generate_data* after changing parameters there appropriately. You will have to increase the simultaneous open file limit in Linux to more than 4096 files to run this script. The copy all the files of the format *subbyte<x>_keyguess<y>* to *source* folder under *2.plot*. Edit parameters in *plot_graphs.m* and run it to get 496 graphs in *graphs* folder.

6. Troubleshooting

A small mistake can lead to problems and errors that would require hours to days to figure out the reason and therefore in this section we briefly describe some probable issues and the approaches to solve them.

- When using an over the shelf USB to RS232 TTL converter if the transmitted data are seen as some funny characters check if the clock frequency matches check the values set for the oscillator frequency in your code for PIC.

- If you are using the inbuilt USB controller in PIC there may be situations where the USB device is not even detected by PC. In such case check if you have provided a 48MHz clock to USB preferably by enabling PLL. Also check if you have connected 3.3V supply to VUSB. Note that this 3.3V should be between VUSB and Vss pins. If out are using the internal 3.3V regulator for VUSB which we do not use in our setup, try giving 3.3V power externally as we have done.

- Another common errors with respect to the internal USB controller are device not recognized error, communication suddenly freezing and data corruption during communication. In such cases check the accuracy of the oscillator you use. Do not use resonators but use crystal oscillators with proper capacitance values. Also check if a capacitor of proper value is connected to VUSB.

- If issues occur after you connect power measurement circuits, then check if you increased the power supply voltages to compensate the drop across the power measuring resistor. Also check whether you have connected the oscilloscope ground to the correct place.

- After measuring power if you cannot derive the correct key use the average function in the oscilloscope and check whether the power trace looks like Fig. 6 in the paper, if it is not and your average trace is almost null then check if the oscilloscope probes have been correctly connected without shorting the resistor. Also check if the power measurement resistor value is too small.

- If you only see few key bytes correct in your answer increase the number of power traces and try. Also read the power for through understanding and see the troubleshooting section.

- If the results are still wrong use the post analysis tools to analyses the power traces obtained to check how the correlation values changes with different parameters after making sure you have set parameters like the number of traces and number of samples in a trace in the code for analysis.