UMass Amherst

College of Engineering

Lab 1: Exploring Side-Channel Attacks

ECE 371: Intro to Security Engineering

Objectives

- Introduction to side-channel attacks
- Practice using Python code
- Learn how different side-channel attacks work, customize attacks and observe effects

Tools

- ChipWhisperer Analyzer
  - Takes in power trace data collected from hardware. *(Note that we have already collected traces for you using our own hardware and the ChipWhisperer Capture tool)*
  - Allows user to use default attacks or make their own custom attacks
  - Shows data on completed attack in terms of PGE and more

Lab Overview

1. Download trace files from Moodle
2. Install Chipwhisperer
3. Learn basics on how to use ChipWhisperer Analyzer tool
4. Attack each trace file using standard attacks provided by ChipWhisperer
5. Modify attacks to alter traces in preprocessing, etc. Run against each trace
6. Demo & Report

Part 1 - Download Traces

- You are provided with a trace file:
    - aes_unprotected
- You will need to perform various attacks on the trace file

Part 2 - Install Chipwhisperer

- We are using Chipwhisperer 4 (not 5) for this lab
- This is not installed on the lab computers currently.. you can either install on the computers yourself or use your personal laptops (which you must bring to Duda for demo!!!)
- If you install on the lab computers, install in Documents or Desktop
- Instructions found here: https://wiki.newae.com/Installing_ChipWhisperer

Part 2 - Install Chipwhisperer

- We are using **Chipwhisperer 4.0.2** (not 5 or any other version!!!)
- This is not installed on the lab computers currently.. you can either install on the computers yourself or use your personal laptops (which you must bring to Duda for demo if using!!!)
- Note that we are only using Chipwhisperer Analyzer, and we are not using a physical board, so you should not need to install drivers or anything
- Instructions found here, see next slides for more detail: https://wiki.newae.com/Installing_ChipWhisperer

Part 2 - Install Chipwhisperer

The following was tested on the computers in Duda, but a similar process should be okay for any computer

1. **Install prerequisites**, which includes Python 2.7 and a few libraries (pyqtgraph, configobj, pyusb, and PySide)

   For Windows, might need to set up WinPython to use Python, (which is already on Duda computers)

      Download WinPython version 2.7.10.3 (32 bit) from here:

      Install in C directory

      Add path to PATH environment variable: C:\WinPython-32bit-2.7.10.3

   In C:\WinPython-32bit-2.7.10.3, double click "WinPython Command Prompt" to launch Python terminal

Part 2 - Install Chipwhisperer

Once in the Python terminal, install the dependencies:

```
pip install pyqtgraph

pip install configobj

pip install pyusb

pip install PySide
```

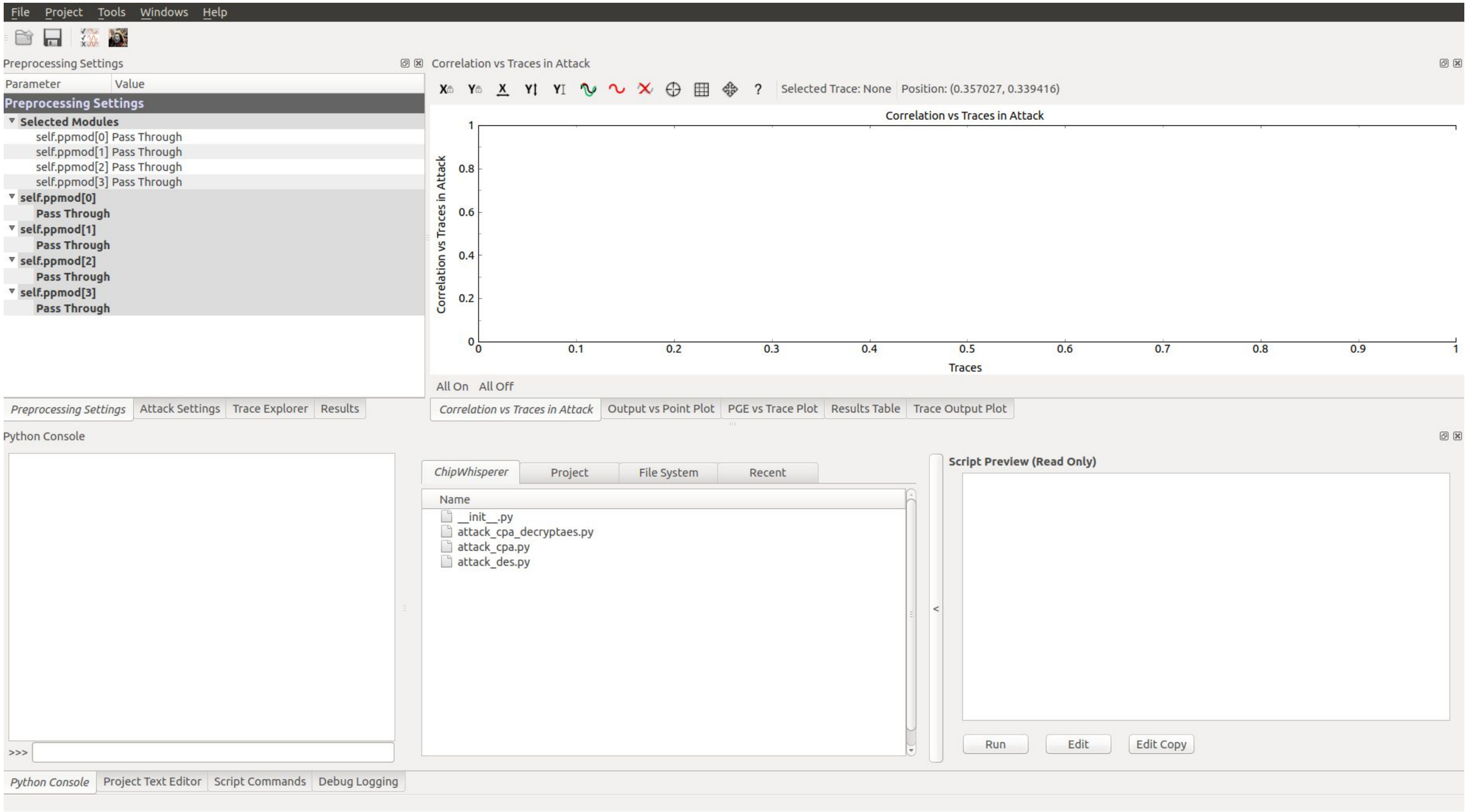
2. **Download Chipwhisperer 4.0.2**, from instructions click on "Installing Chipwhisperer from Releases"

   Click on Releases and navigate to 4.0.2, download zip file

   Extract zip file

3. **Open Chipwhisperer Analyzer** by opening to Python terminal, navigating to *chipwhisperer-4.0.2\software* folder, and opening the software with command *python CWAnalyzer.pyw*

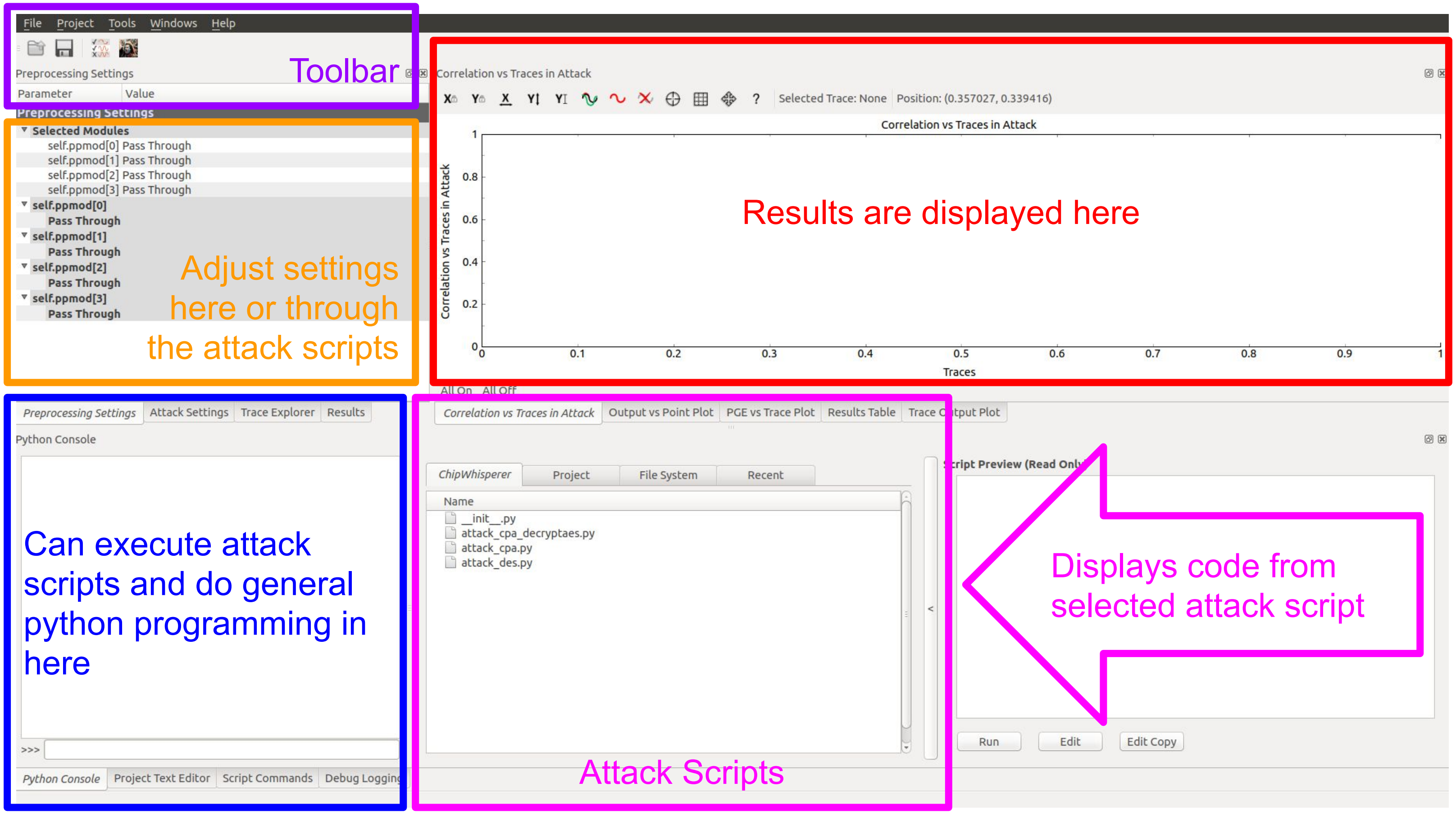
Part 3 - CW Analyzer



Parameter	Value
Preprocessing Settings	
Selected Modules	
self.ppmod[0]	Pass Through
self.ppmod[1]	Pass Through
self.ppmod[2]	Pass Through
self.ppmod[3]	Pass Through
self.ppmod[0]	
Pass Through	
self.ppmod[1]	
Pass Through	
self.ppmod[2]	
Pass Through	
self.ppmod[3]	
Pass Through	

- __init__.py
- attack_cpa_decryptaes.py
- attack_cpa.py
- attack_des.py

Part 3 - CW Analyzer



Toolbar

Adjust settings here or through the attack scripts

Results are displayed here

Can execute attack scripts and do general python programming in here

Attack Scripts

Displays code from selected attack script

Preprocessing Settings

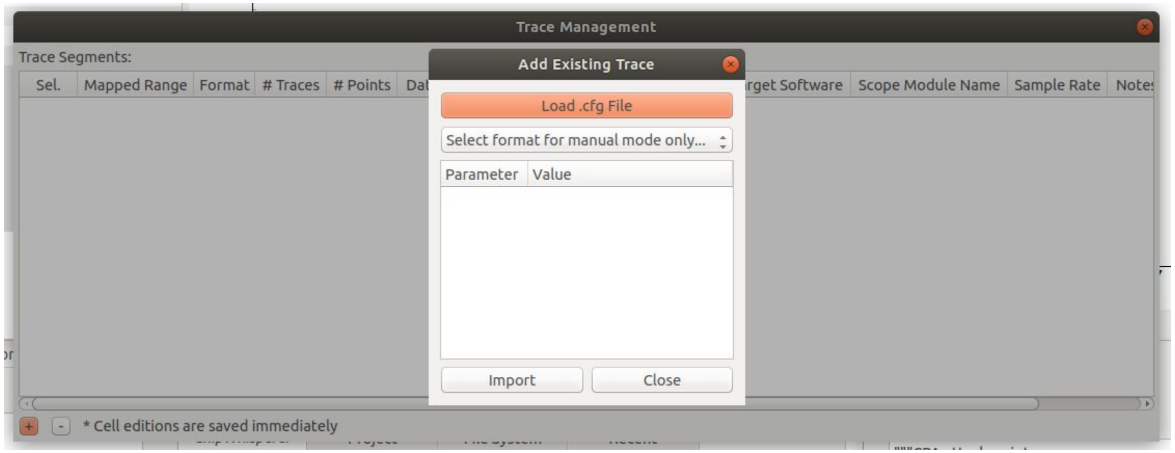
Parameter	Value
Preprocessing Settings	
Selected Modules	
self.ppmod[0] Pass Through	
self.ppmod[1] Pass Through	
self.ppmod[2] Pass Through	
self.ppmod[3] Pass Through	
self.ppmod[0]	
Pass Through	
self.ppmod[1]	
Pass Through	
self.ppmod[2]	
Pass Through	
self.ppmod[3]	
Pass Through	

Selected Trace: None Position: (0.357027, 0.339416)

```
__init__.py
attack_cpa_decryptaes.py
attack_cpa.py
attack_des.py
```

Part 4 - Loading Traces

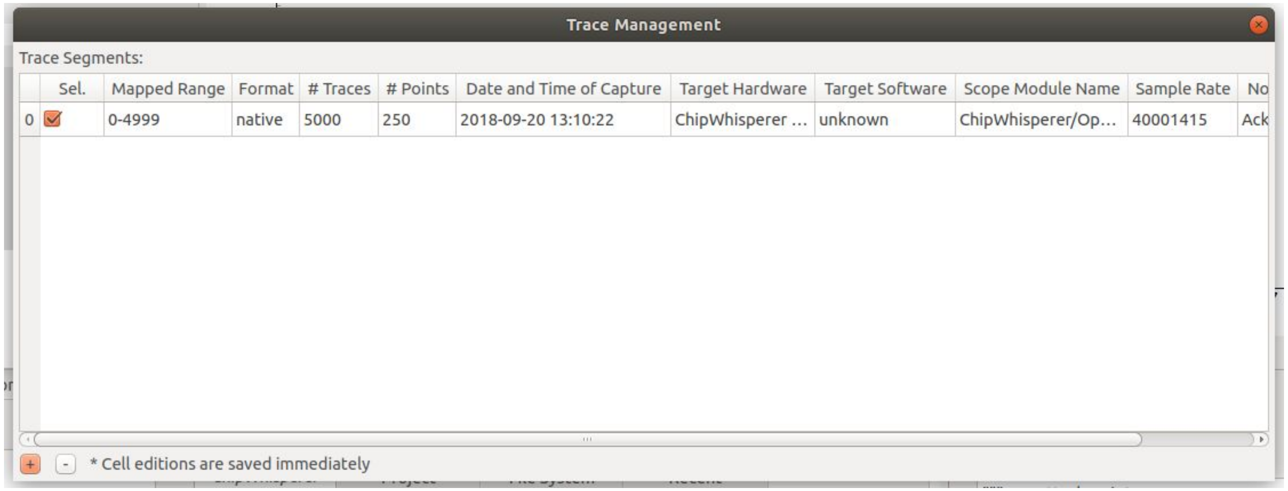
- Click on the Trace Manager icon shown in the Toolbar
- Pop up should appear. Click on little plus sign in bottom left corner.
- Click "Load .cfg File", navigate to folder containing traces and select appropriate .cfg file, then click "import"



* Cell editions are saved immediately

Part 4 - Loading Traces

- Now trace information should appear in Trace Management window, checkmark should appear in the select column
- Press x to exit the window, your traces have now been loaded



Trace Management

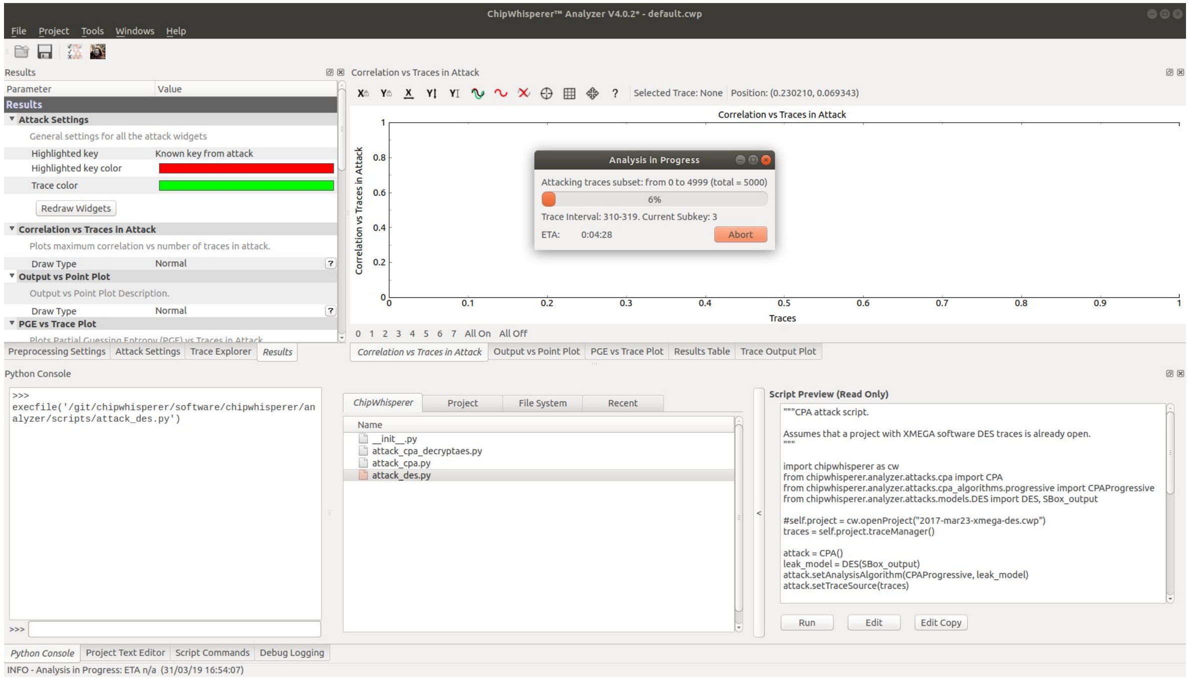
Trace Segments:

	Sel.	Mapped Range	Format	# Traces	# Points	Date and Time of Capture	Target Hardware	Target Software	Scope Module Name	Sample Rate	No
0	☑	0-4999	native	5000	250	2018-09-20 13:10:22	ChipWhisperer ...	unknown	ChipWhisperer/Op...	40001415	Ack

* Cell editions are saved immediately

Part 4 - Run Attack

- To execute an attack from an Attack Script, simply double click on the attack script!
- A progress bar should appear that also gives an estimation of the time remaining, it closes on its own when the attack is done



Parameter	Value
Results	
Attack Settings	
General settings for all the attack widgets	
Highlighted key	Known key from attack
Highlighted key color	🟥
Trace color	🟩
Redraw Widgets	
Correlation vs Traces in Attack	
Plots maximum correlation vs number of traces in attack.	
Draw Type	Normal
Output vs Point Plot	
Output vs Point Plot Description.	
Draw Type	Normal
PGE vs Trace Plot	
Plots Partial Guessing Entropy (PGE) vs Traces in Attack	

Analysis in Progress

Attacking traces subset: from 0 to 4999 (total = 5000)

6%

Trace Interval: 310-319. Current Subkey: 3

ETA: 0:04:28

Abort

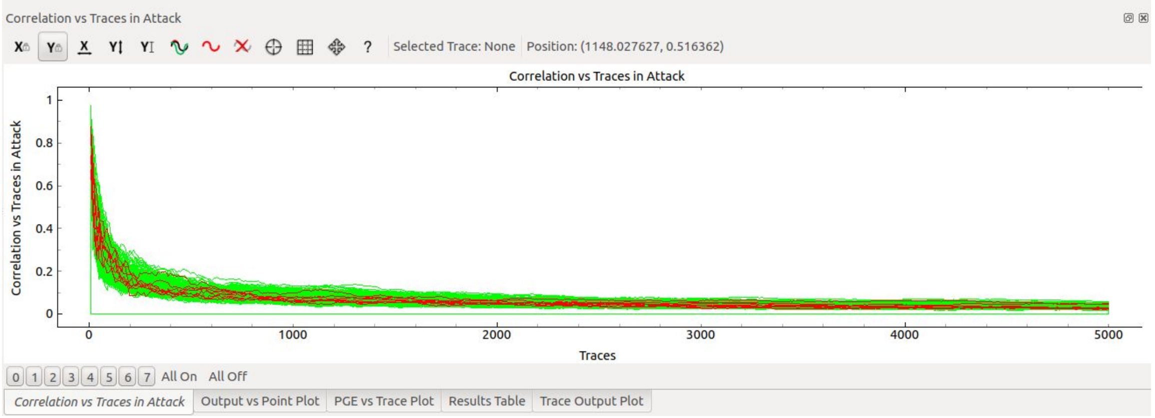
```
>>>
execfile('/git/chipwhisperer/software/chipwhisperer/analyzer/scripts/attack_des.py')
```

```
"""CPA attack script.

Assumes that a project with XMEGA software DES traces is already open.
"""

import chipwhisperer as cw
from chipwhisperer.analyzer.attacks.cpa import CPA
from chipwhisperer.analyzer.attacks.cpa_algorithms.progressive import CPAProgressive
from chipwhisperer.analyzer.attacks.models.DES import DES, SBox_output

#self.project = cw.openProject("2017-mar23-xmega-des.cwp")
traces = self.project.traceManager()

attack = CPA()
leak_model = DES(SBox_output)
attack.setAnalysisAlgorithm(CPAProgressive, leak_model)
attack.setTraceSource(traces)
```

```
INFO - Analysis in Progress: ETA n/a (31/03/19 16:54:07)
```

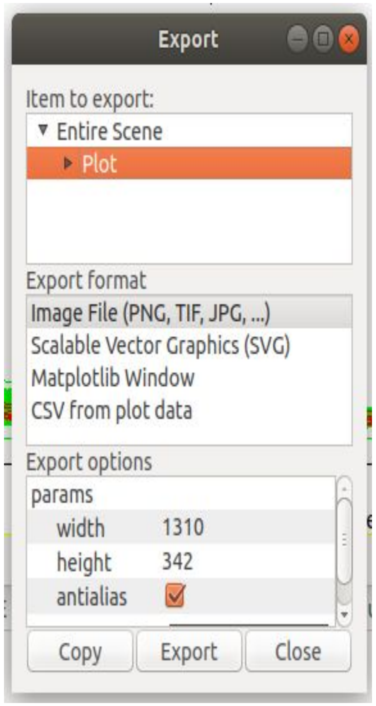
Part 4 - Analyze Data

- In the Results area, click "All On" in the bottom left to view results from all bytes, click individual numbers to see individual bytes
- Five different tabs are the five different results graphs


Correlation vs Traces in Attack
Selected Trace: None
Position: (1148.027627, 0.516362)
Correlation vs Traces in Attack
Correlation vs Traces in Attack
1
0.8
0.6
0.4
0.2
0
0
1000
2000
3000
4000
5000
Traces
0 1 2 3 4 5 6 7
All On
All Off
Correlation vs Traces in Attack
Output vs Point Plot
PGE vs Trace Plot
Results Table
Trace Output Plot

Part 4 - Notes

- You can right click on a graph to export the data as a CSV file or into one of several different image formats
- If the results tabs do not appear for you, go to Windows (in toolbar) to manually select the different results plots
- You can manually change the Attack Settings outside of the Python script, however I have found it to be unreliable in the past (may not execute these settings), so it is always best to change the Python script



Export

Item to export:

- Entire Scene
  - Plot

Export format

- Image File (PNG, TIF, JPG, ...)
- Scalable Vector Graphics (SVG)
- Matplotlib Window
- CSV from plot data

Export options

params

width	1310
height	342
antialias	☑

Part 5 - Modifying Attacks

- You will need to create Python scripts to create your own custom attacks
- To make your life simpler, never directly edit the default attack scripts, simply make a copy of the file and give it a different name, then make your own modifications
- The Attack Scripts themselves can be found in:

  chipwhisperer/software/chipwhisperer/analyzer/scripts
- Run your custom attacks the same way as the default attacks

Part 5 - Python Indentation

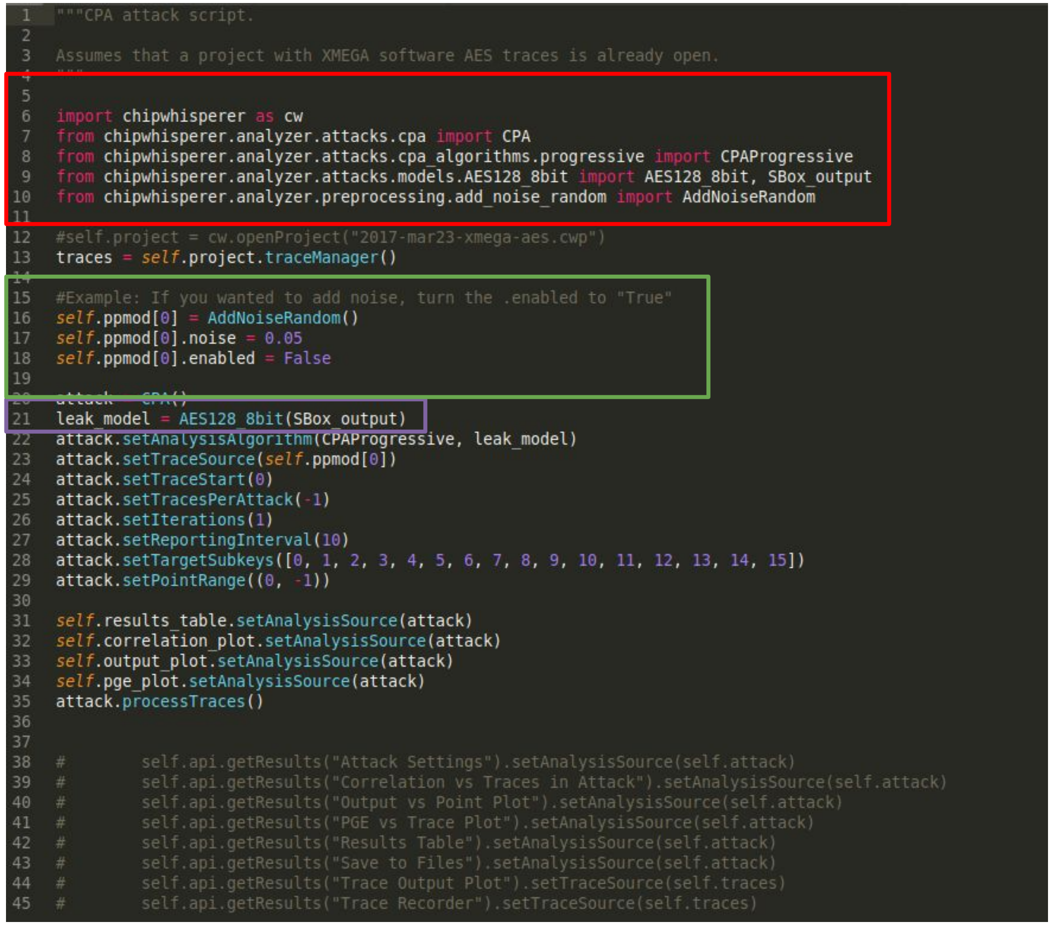
- You may find it easier to write your code in a text editor such as Notepad++ or Sublime
- In Sublime, indentation errors can be fixed by selecting all of the text and going to:
  - View -> Indentation -> Convert Indentation to Spaces to ensure that existing tabs are converted to spaces
  - View -> Indentation -> Indent Using Spaces to ensure that in the future all tabs will be represented as spaces
- Similar steps can be done in Notepad++:
  - Preferences -> Language Menu/Tab Settings

Part 5 - ChipWhisperer Script Documentation

- Some resources to help you with modifying attack scripts:
    - https://chipwhisperer.readthedocs.io/en/latest/
    - https://github.com/newaetech/chipwhisperer

Part 5 - Hints

- All of the Imports are scripts that exist in the chipwhisperer source code and are local to your machine
- Preprocessing can be added, more than just noise is possible
- Leak Model determines where CPA is attacking



```python
"""CPA attack script.

Assumes that a project with XMEGA software AES traces is already open.
"""

import chipwhisperer as cw
from chipwhisperer.analyzer.attacks.cpa import CPA
from chipwhisperer.analyzer.attacks.cpa_algorithms.progressive import CPAProgressive
from chipwhisperer.analyzer.attacks.models.AES128_8bit import AES128_8bit, SBox_output
from chipwhisperer.analyzer.preprocessing.add_noise_random import AddNoiseRandom

#self.project = cw.openProject("2017-mar23-xmega-aes.cwp")
traces = self.project.traceManager()

#Example: If you wanted to add noise, turn the .enabled to "True"
self.ppmod[0] = AddNoiseRandom()
self.ppmod[0].noise = 0.05
self.ppmod[0].enabled = False

attack = CPA()
leak_model = AES128_8bit(SBox_output)
attack.setAnalysisAlgorithm(CPAProgressive, leak_model)
attack.setTraceSource(self.ppmod[0])
attack.setTraceStart(0)
attack.setTracesPerAttack(-1)
attack.setIterations(1)
attack.setReportingInterval(10)
attack.setTargetSubkeys([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])
attack.setPointRange((0, -1))

self.results_table.setAnalysisSource(attack)
self.correlation_plot.setAnalysisSource(attack)
self.output_plot.setAnalysisSource(attack)
self.pge_plot.setAnalysisSource(attack)
attack.processTraces()


#        self.api.getResults("Attack Settings").setAnalysisSource(self.attack)
#        self.api.getResults("Correlation vs Traces in Attack").setAnalysisSource(self.attack)
#        self.api.getResults("Output vs Point Plot").setAnalysisSource(self.attack)
#        self.api.getResults("PGE vs Trace Plot").setAnalysisSource(self.attack)
#        self.api.getResults("Results Table").setAnalysisSource(self.attack)
#        self.api.getResults("Save to Files").setAnalysisSource(self.attack)
#        self.api.getResults("Trace Output Plot").setTraceSource(self.traces)
#        self.api.getResults("Trace Recorder").setTraceSource(self.traces)
```

Part 6 - Demo & Report

- The attacks are as follows:
    - Default DPA attack (Script named: attack_dpa.py)
    - Default CPA attack (Script named:_cpa.py)
    - CPA with Last Round State key leakage instead of the SBox Output
    - CPA with Last Round State key leakage with noise
    - CPA with Last Round State key leakage with clock jitter
- Where noise and jitter are as follows:
    - Noise = (Sum of last digit of Spire ID for all group members + 1)/(Number of Group Members * 100)
    - Jitter = ((Sum of first digit of Spire ID for all group members)/(Number of Group Members)) mod 5

Part 6 - Demo & Report

- For each attack, you must provide the following information in your report:
    - Images of the PGE vs Trace Plot and at least the first six rows of the Results Table
    - Trace number of the first time where PGE <= 5 and PGE = 0 for all bytes. Compile this information for all the attacks into one table.
    - Explanation of whether or not the attack was successful.
    - Additionally, you must submit the 3 attack scripts that you modified to Turnitin.
- You will also be required to demo your project
- You must also write a brief report explaining what you did in this lab, including images and explanation of your code and results.

Best of luck!