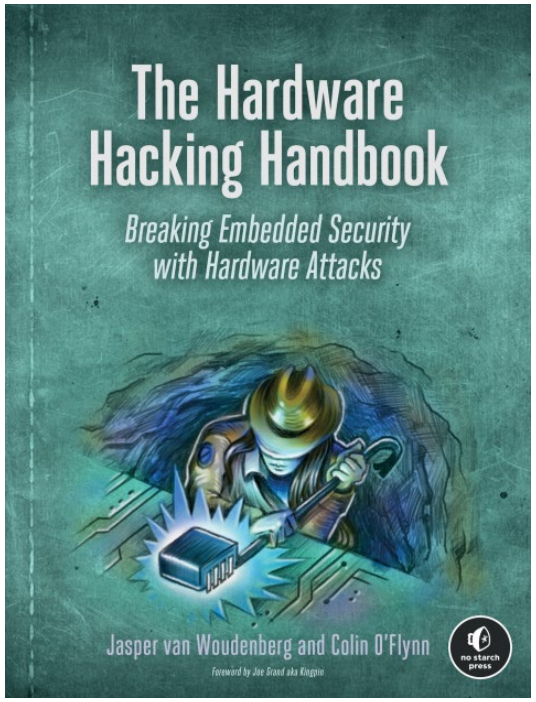
Hands on with Non-Invasive Hardware Security Tooling

Colin O'Flynn

New England Hardware Security Day 2022

About Me & This Talk



- Started ChipWhisperer project
  - Power analysis, fault injection, including hardware & software.
  - Variety of open-source & not tools
  - Now a company supporting 6 people – with some local connection coming soon (Cambridge/Boston area), ***if you are looking for work please get in touch!***
- Was assistant professor at Dalhousie University (now adjunct to do ChipWhisperer stuff full-time instead)
- Co-author of "The Hardware Hacking Handbook" alongside *Jasper Van Woudenberg*
  - Published with No Starch Press (physical book Nov / 2021)

Links to material on blog post at **oflynn.com**

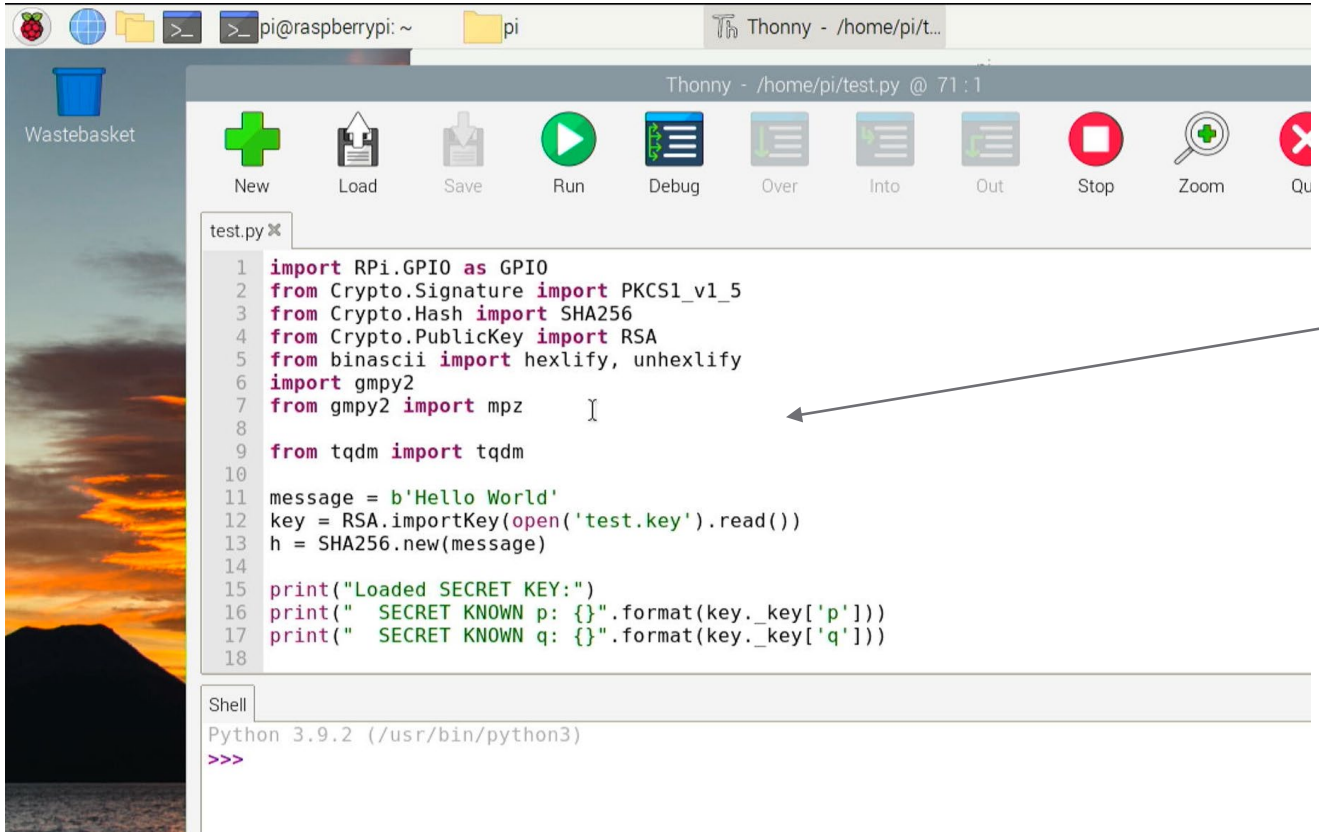
Topics in this demo-focused talk

- Fault injection on Raspberry Pi 3 Model B+:
  - Faulting RSA signing operation to recover private key
  - EMFI
- RISC-V Soft Core
- ECC / FPGA Attacks

Fault Injection on Raspberry Pi 3 Model B+

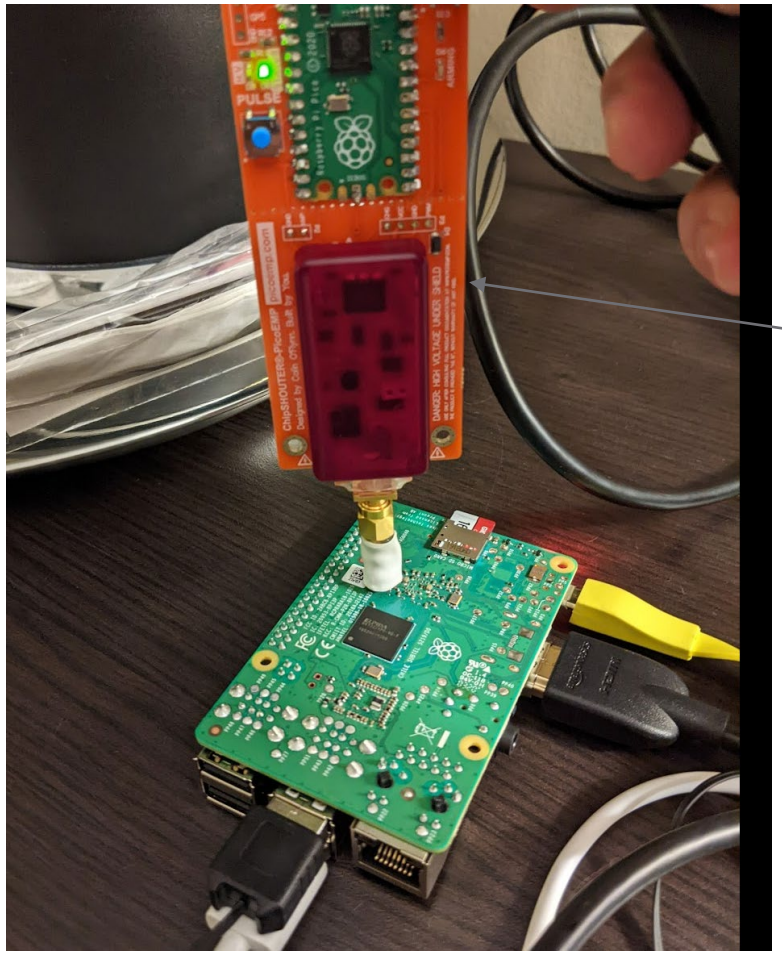
Objective: DFA on RSA (from Python!)

R-Pi as Target & Platform



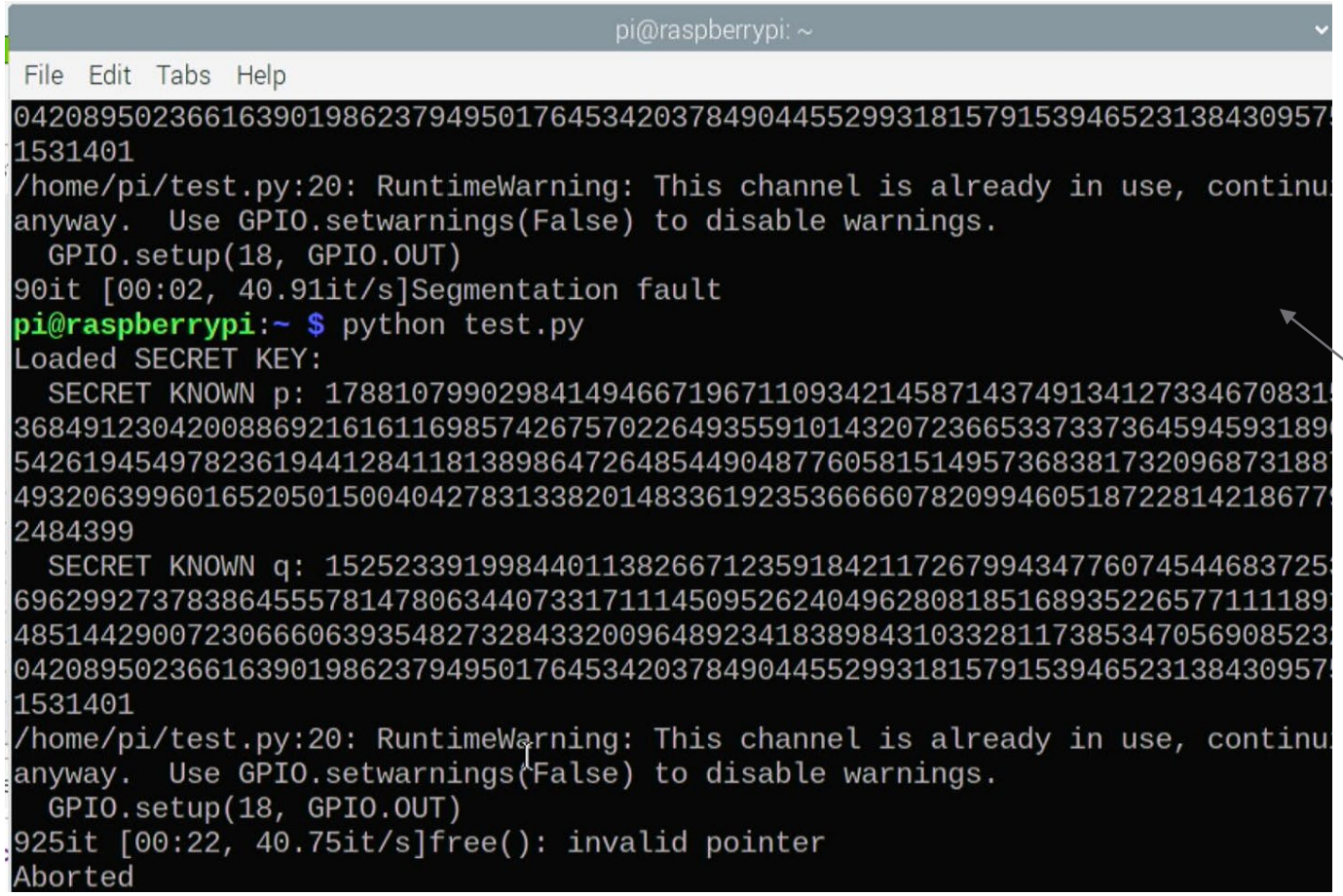
We're going to run some Python code on a Raspberry Pi Model 3 B+.

R-Pi as Target & Platform

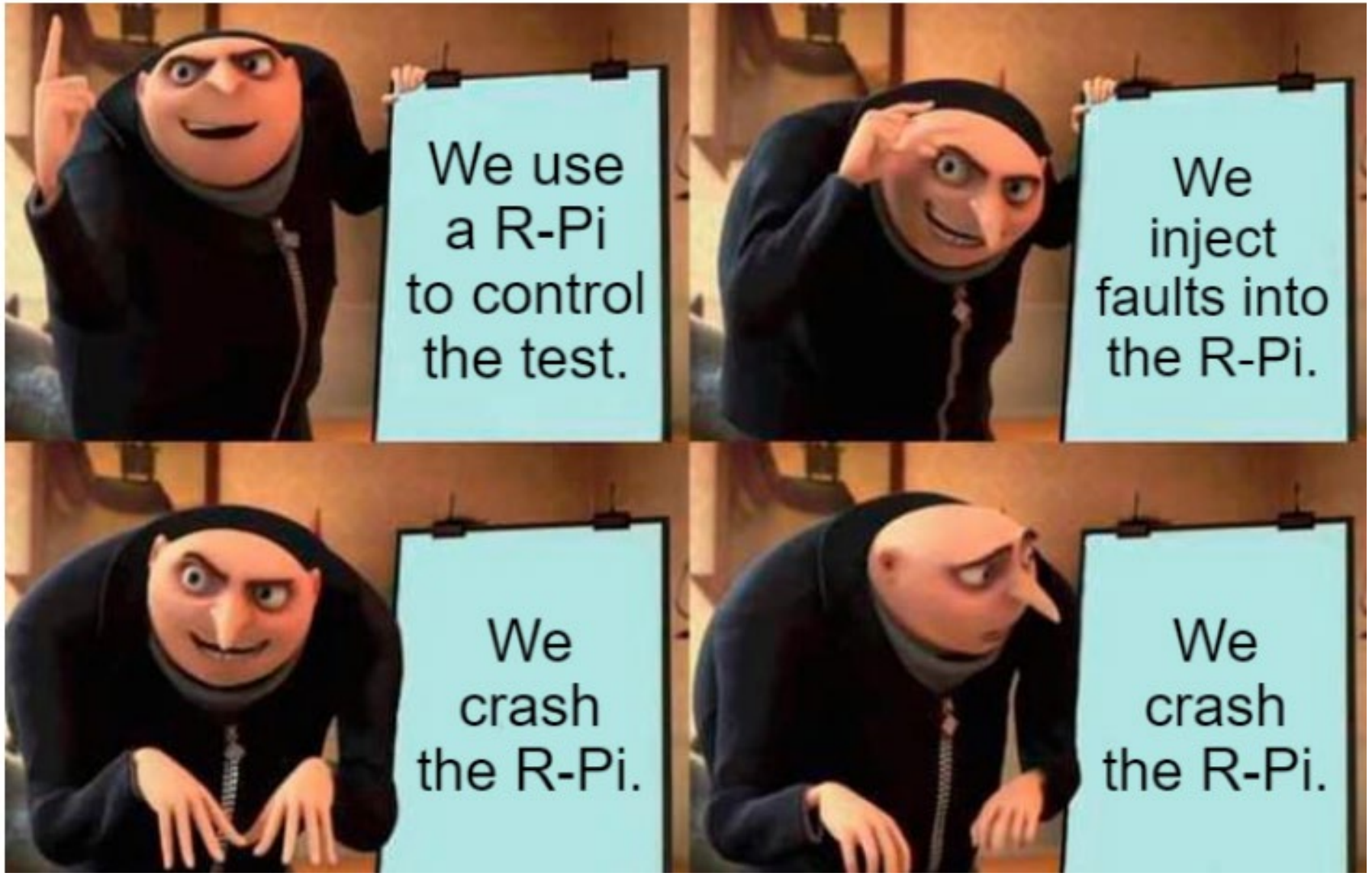


We’re going to inject faults into a Raspberry Pi Model B 3+

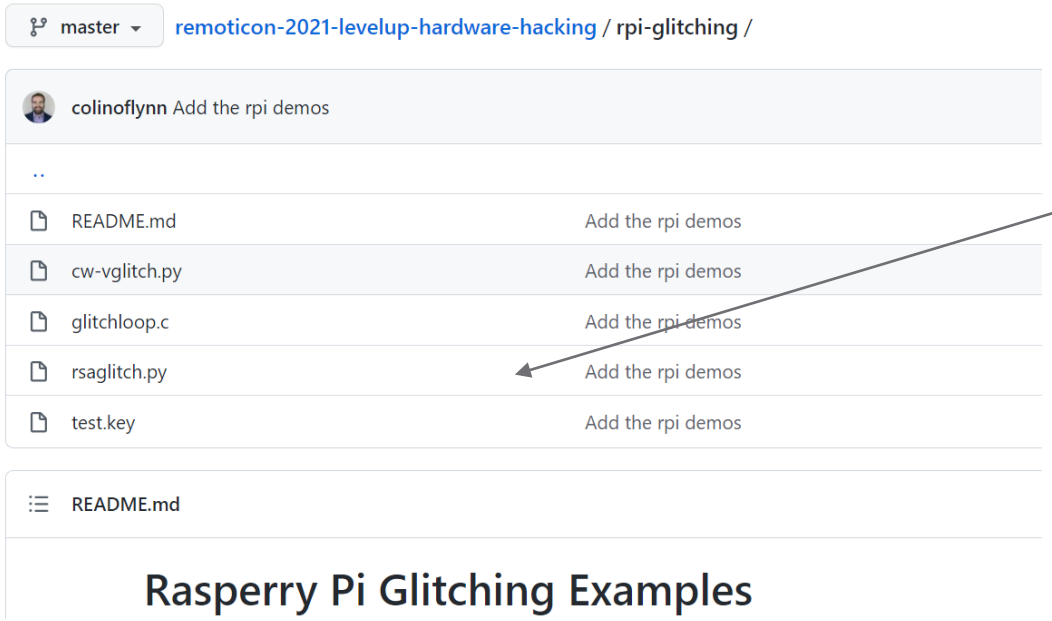
R-Pi as Target & Platform



We're going to crash a Raspberry Pi Model B 3+


We use a R-Pi to control the test.
We inject faults into the R-Pi.
We crash the R-Pi.
We crash the R-Pi.

My Code for R-Pi

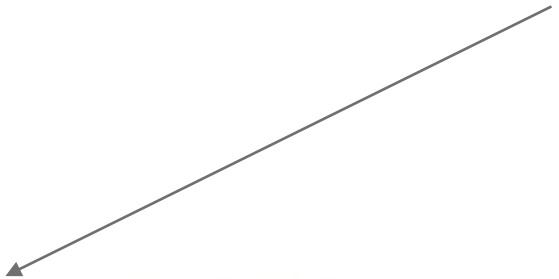


Running this code.

https://github.com/colinoflynn/remoticon-2021-levelup-hardware-hacking/tree/master/rpi-glitching

Follow Along with Co-Lab / Python

```
31  while True:
32          GPIO.output(18, GPIO.LOW)
33          GPIO.output(18, GPIO.HIGH)
34          output = p.sign(h)
35          GPIO.output(18, GPIO.LOW)
36
37          #output = b'\x93\x07\xc0\x02\xc9\x85\n\xb3lY\xad\xb4hY\
38          #output = b'\xc9\xa0\x14\x0f\xad\xd3\xb2\x9dK\x15\xe7Zg
39
```



These are example *faulty* outputs – if you uncomment this code acts as if you received such a faulty output!

See if you can get the p/q recovery. If so you can run this yourself.

WATCH ENVIRONMENT SETUP: Need <u>specific version of pycryptodome</u>!

The issue is fixed in any recent version.

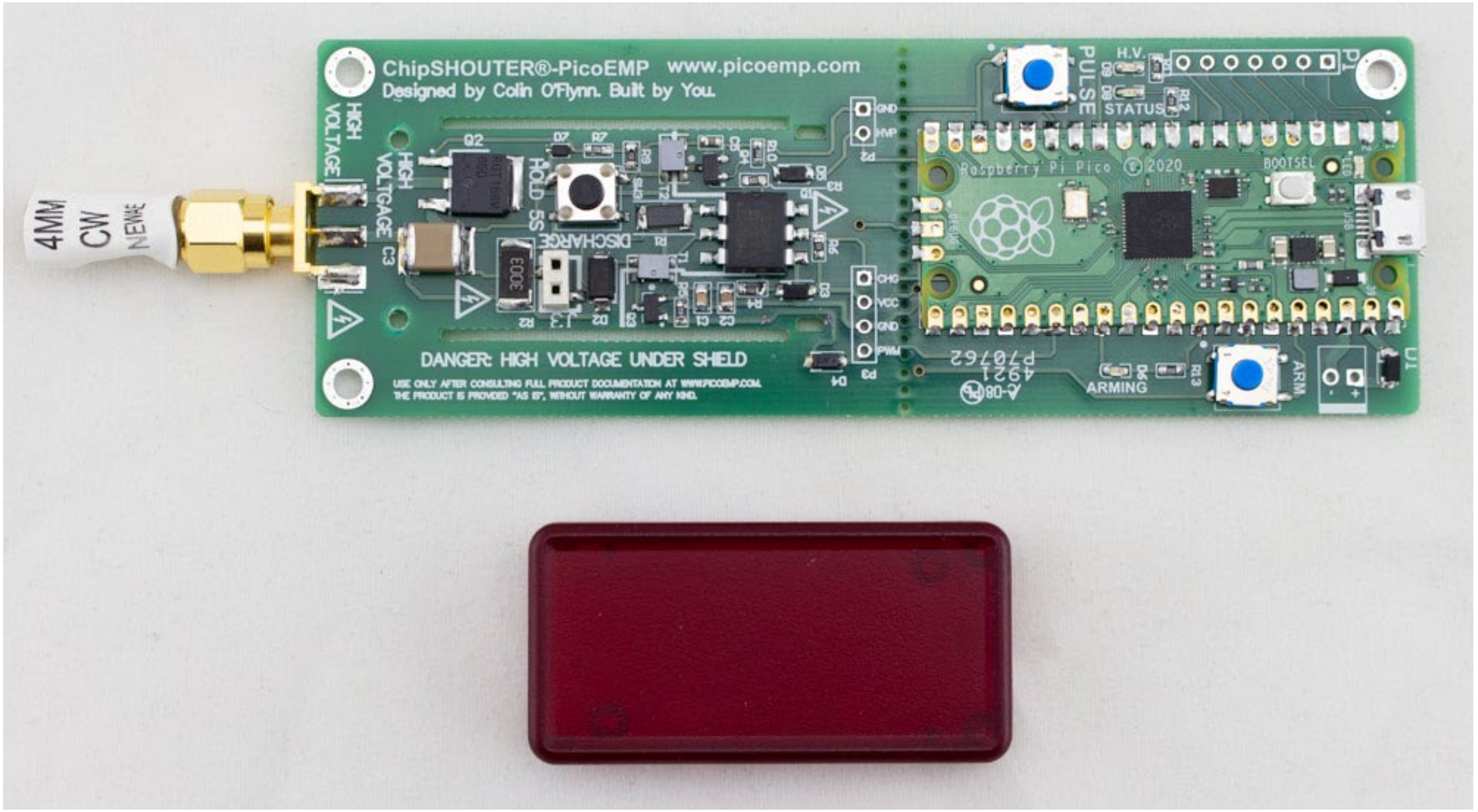
Cheap EMFI Tooling

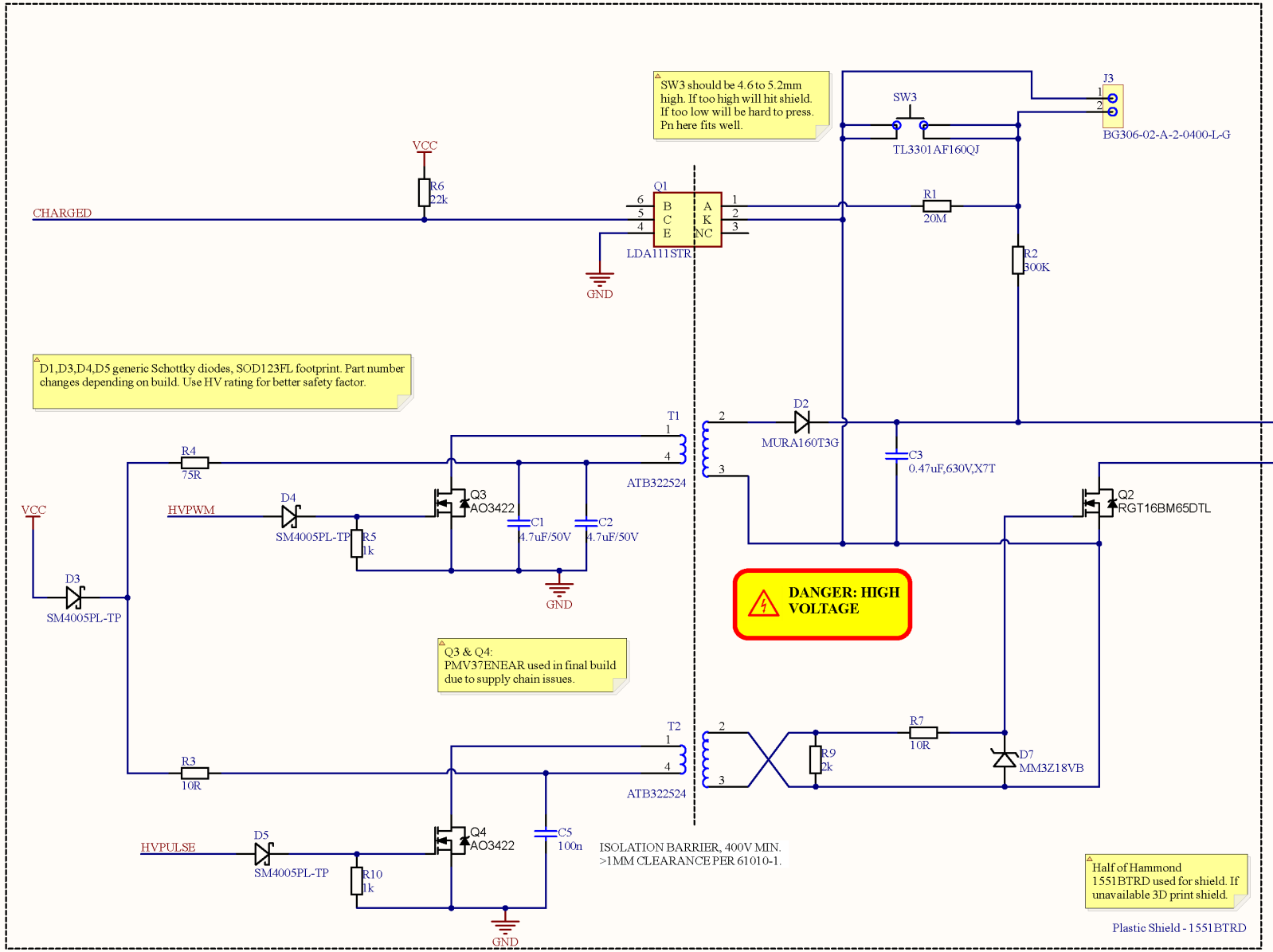
Initial “Safe” Version

Hackaday Remoticon 2021

https://github.com/colinoflynn/remoticon-2021-levelup-hardware-hacking/tree/master/dangerous-emfi

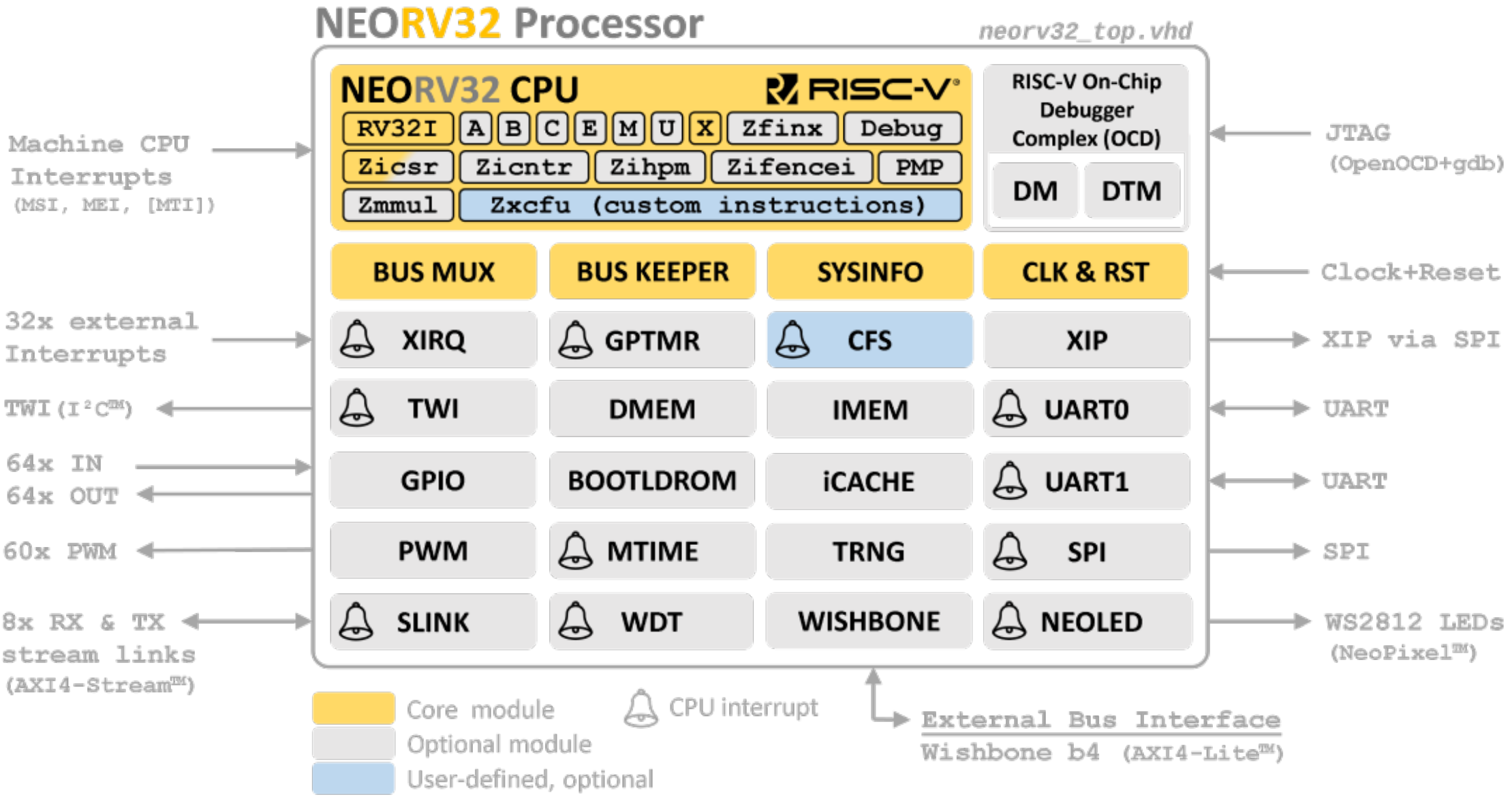
PicoEMP

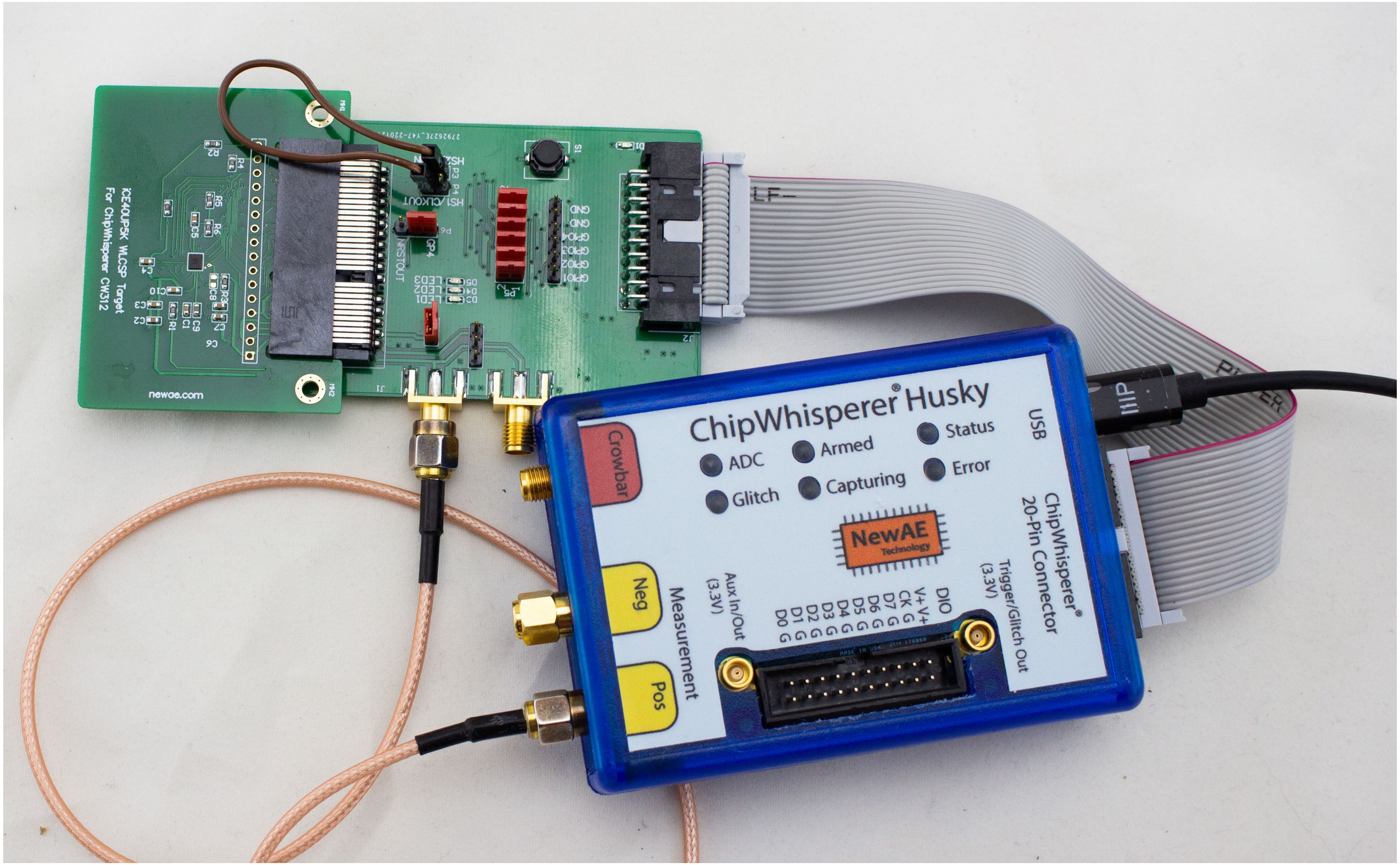

ChipSHOUTER®-PicoEMP www.picoemp.com
Designed by Colin O'Flynn. Built by You.
DANGER: HIGH VOLTAGE UNDER SHIELD
USE ONLY AFTER CONSULTING FULL PRODUCT DOCUMENTATION AT WWW.PICOEMP.COM.
THE PRODUCT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND.
HIGH VOLTAGE
HIGH VOLTAGE
4MM CW
HOLD 5S
DISCHARGE
PULSE
STATUS
H.V.
ARMING
ARM
BOOTSEL
Raspberry Pi Pico © 2020
GND
HVP
CHG
VCC
GND
PWM



Title: **High Voltage Circuitry**		Approved: YES	
Rev: 04	Project: PicoEMP .	License: CC BY-SA 3.0	
Date: 2021-12-16	Time: 8:53:16 PM | Sheet 1 of 2	Copyright © NewAE Technology Inc.	NewAE.com
File: chipshouter-pico-hv.SchDoc		Copyright (C) Colin O'Flynn, 2021	

RISC-V Soft-Core Attack


NEORV32 Processor
neorv32_top.vhd
NEORV32 CPU
RISC-V
RV32I
A
B
C
E
M
U
X
Zfinx
Debug
Zicsr
Zicntr
Zihpm
Zifencei
PMP
Zmmul
Zxcfu (custom instructions)
RISC-V On-Chip Debugger Complex (OCD)
DM
DTM
BUS MUX
BUS KEEPER
SYSINFO
CLK & RST
XIRQ
GPTMR
CFS
XIP
TWI
DMEM
IMEM
UART0
GPIO
BOOTLDROM
iCACHE
UART1
PWM
MTIME
TRNG
SPI
SLINK
WDT
WISHBONE
NEOLED
Machine CPU Interrupts
(MSI, MEI, [MTI])
32x external Interrupts
TWI (I²C™)
64x IN
64x OUT
60x PWM
8x RX & TX stream links
(AXI4-Stream™)
JTAG
(OpenOCD+gdb)
Clock+Reset
XIP via SPI
UART
UART
SPI
WS2812 LEDs
(NeoPixel™)
Core module
Optional module
User-defined, optional
CPU interrupt
External Bus Interface
Wishbone b4 (AXI4-Lite™)


ChipWhisperer® Husky
ADC
Armed
Status
Glitch
Capturing
Error
USB
Crowbar
NewAE
Technology
ChipWhisperer®
20-Pin Connector
Trigger/Glitch Out
(3.3V)
Aux In/Out
(3.3V)
Measurement
Neg
Pos
DIO
V+ V+
CK G
D7 G
D6 G
D5 G
D4 G
D3 G
D2 G
D1 G
D0 G
iCE40UP5K WLCSP Target
For ChipWhisperer CW312
newae.com

Example – Simple AES Attack

Rebuilding your RISC-V Core

```
Info: Slack histogram:
Info:  legend: * represents 21 endpoint(s)
Info:          + represents [1,21) endpoint(s)
Info: [ 39029,  41053) |*+
Info: [ 41053,  43077) |+
Info: [ 43077,  45101) |+
Info: [ 45101,  47125) |+
Info: [ 47125,  49149) |**********+
Info: [ 49149,  51173) |*****+
Info: [ 51173,  53197) |****+
Info: [ 53197,  55221) |***+
Info: [ 55221,  57245) |****+
Info: [ 57245,  59269) |******+
Info: [ 59269,  61293) |*************+
Info: [ 61293,  63317) |**********************+
Info: [ 63317,  65341) |*************+
Info: [ 65341,  67365) |******************+
Info: [ 67365,  69389) |**********************+
Info: [ 69389,  71413) |****************+
Info: [ 71413,  73437) |*********+
Info: [ 73437,  75461) |************************************************************
Info: [ 75461,  77485) |****************************+
Info: [ 77485,  79509) |*******************************************+
6 warnings, 0 errors

Info: Program finished normally.
icepack neorv32_iCE40CW312_MinimalBoot.asc neorv32_iCE40CW312_MinimalBoot.bit
make[3]: Leaving directory '/c/dev/neorv32-setups/osflow'
IMPL="${BITSTREAM%%.*}"; for item in ".bit" ".svf"; do \
  if [ -f "./$IMPL$item" ]; then \
    mv "./$IMPL$item" ./; \
  fi \
done
make[2]: Leaving directory '/c/dev/neorv32-setups/osflow'
make[1]: Leaving directory '/c/dev/neorv32-setups/osflow'
```

Rebuilding your Firmware

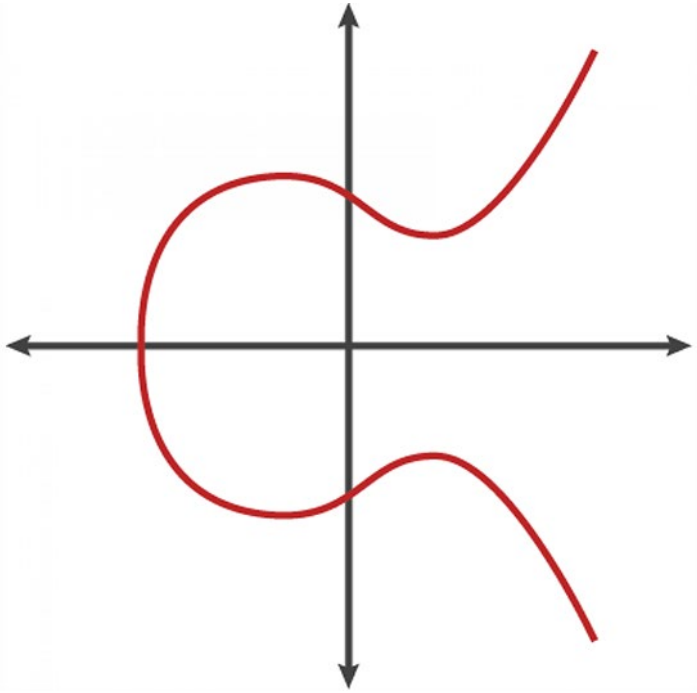
```
Creating load file for EEPROM: simpleserial-aes-CW308_NEORV32.eep
riscv32-unknown-elf-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" \
--change-section-lma .eeprom=0 --no-change-warnings -O ihex simpleserial-aes-CW308_NEORV32.elf simp
ORV32.eep || exit 0
.
Creating Extended Listing: simpleserial-aes-CW308_NEORV32.lss
riscv32-unknown-elf-objdump -h -S -z simpleserial-aes-CW308_NEORV32.elf > simpleserial-aes-CW308_NE
.
Creating Symbol Table: simpleserial-aes-CW308_NEORV32.sym
riscv32-unknown-elf-nm -n simpleserial-aes-CW308_NEORV32.elf > simpleserial-aes-CW308_NEORV32.sym
Size after:
   text    data     bss     dec     hex filename
   5784     272    5696   11752    2de8 simpleserial-aes-CW308_NEORV32.elf
+--------------------------------------------------------
+ Default target does full rebuild each time.
+ Specify buildtarget == allquick == to avoid full rebuild
+--------------------------------------------------------
+--------------------------------------------------------
+ Built for platform iCE40 Target with neorv softcore with:
+ CRYPTO_TARGET = TINYAES128C
+ CRYPTO_OPTIONS = AES128C
+--------------------------------------------------------
```

What can you do?

- Change the code.
- Change the core parameters.
- Add custom core instructions.
- Totally open source!

FPGA ECC Attack

About ECC FPGA Attacks



You know it's ECC because I used this figure.

$$k * G$$

If attacker knows this == very bad.

How bad is it to know this?

Steal your games.

https://www.bbc.com/news/technology-12116051

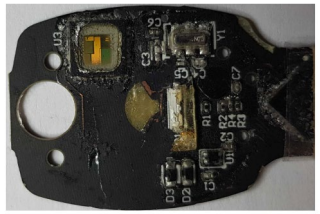


Figure 3: *Google Titan Security Key* PCB, with NXP A7005a die visible after wet chemical attack of its package

prove cumbersome. We had to find a workaround to study the implementation in a more convenient setting.

A700x. Furtherr
ily is Common
certification fou
We went thr
line a
are ba
public
exhau
chips.
We
port B
the cl
ard 3.
refere
Titan.
Ope
that al
ard *ap*
of low
level
with t

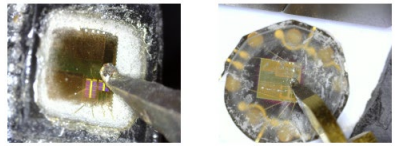


Figure 4: EM Probe Positions on *Titan* (left) and *Rhea* (right)

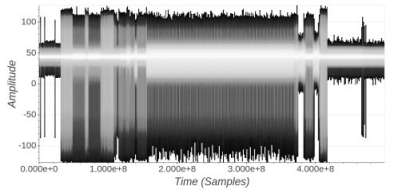
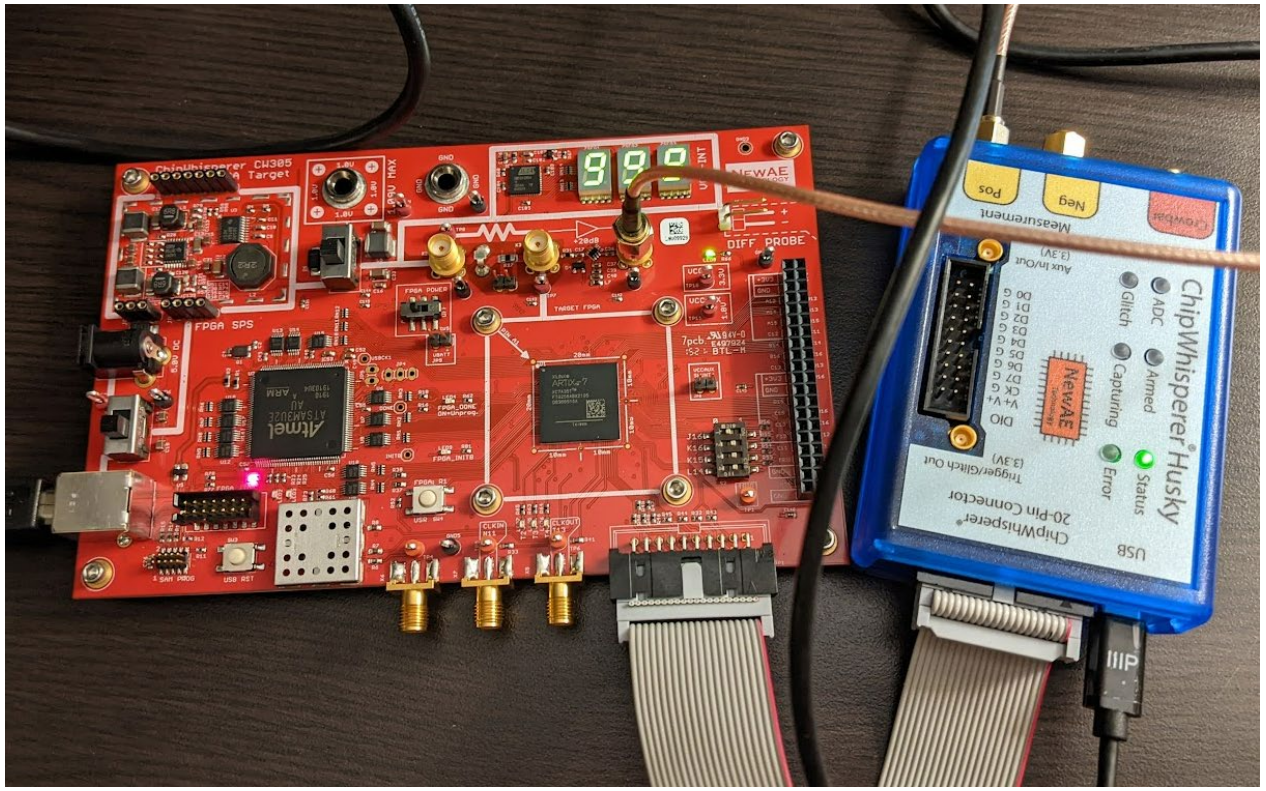


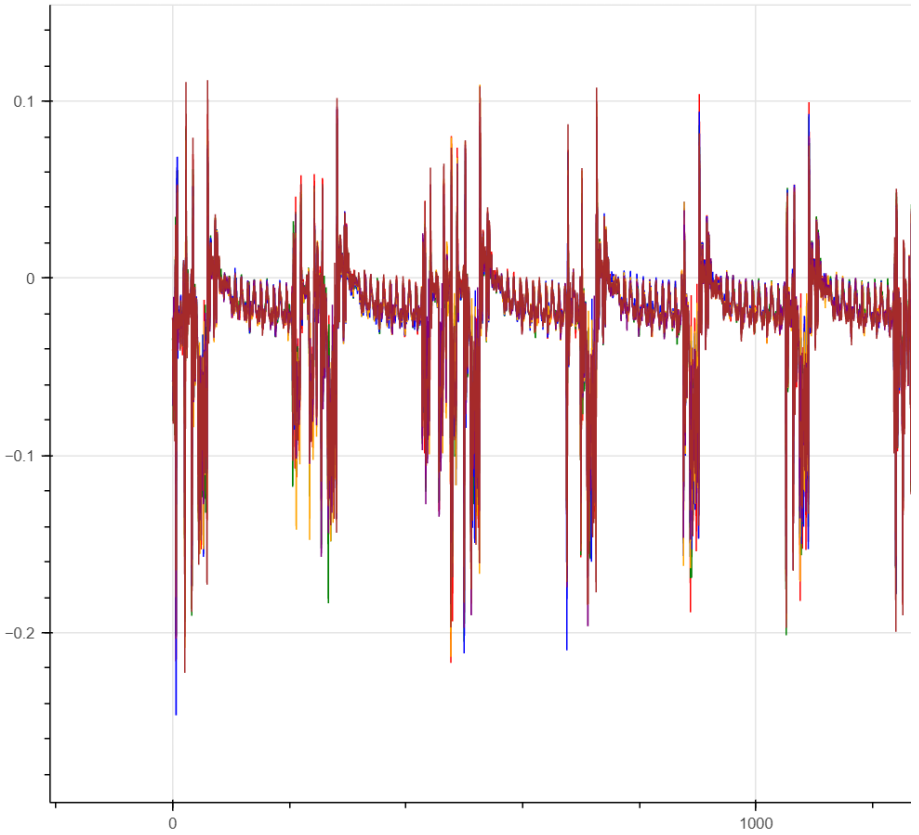
Figure 5: *Titan* ECDSA Signature EM Trace

Steal your auth tokens.

https://www.usenix.org/conference/usenixsecurity21/presentation/roche

High-Level Setup





https://github.com/newaetech/chipwhisperer-jupyter/blob/master/demos/CW305_ECC

Following Along

Ark of the ECC

An open-source ECDSA power analysis attack on a FPGA based Curve P-256 implementation

Jean-Pierre Thibault[1], Colin O'Flynn[1,2], and Alex Dewar[1]

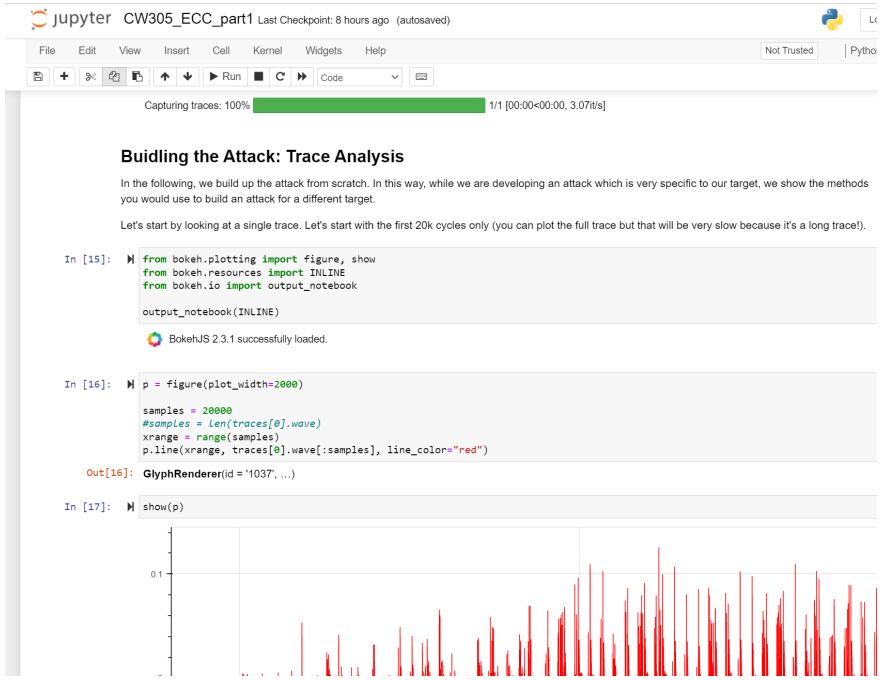
[1] NewAE Technology Inc, Canada
[2] Dalhousie University, Canada
{jpthibault,coflynn,adewar}@newae.com

Abstract. Power analysis attacks on ECC have been presented since almost the very beginning of DPA itself, even before the standardization of AES. Given that power analysis attacks against AES are well known and have a large body of practical artifacts to demonstrate attacks on both software and hardware implementations, it is surprising that these artifacts are generally lacking for ECC. In this work we begin to remedy this by providing a complete open-source ECDSA attack artifact, based on a high-quality hardware ECDSA core from the CrypTech project. We demonstrate an effective power analysis attack against an FPGA implementation of this core. As many recent secure boot solutions are using ECDSA, efforts into building open-source artifacts to evaluate attacks on ECDSA are highly relevant to ongoing academic and industrial research programs. To demonstrate the value of this evaluation platform, we implement several countermeasures and show that evaluating leakage on hardware is critical to understand the effectiveness of a countermeasure.

Keywords: power analysis · ECDSA · FPGA evaluation

1 Introduction

Side-channel power analysis attacks against cryptographic implementations are well-known in practice, starting with their seminal introduction in 1999 [16]. Since then, a considerable amount of work has been focused on symmetric algorithms, and in particular AES. Power analysis against AES has been demonstrated in real-life examples of software and hardware [21,22,28,18,25,8,32] attacks, and a reader can refer to widely available material such as published books [20], training courses, community driven tutorials such as part of the



jupyter CW305_ECC_part1 Last Checkpoint: 8 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help — Not Trusted

Run — Code

Capturing traces: 100% 1/1 [00:00<00:00, 3.07it/s]

Buidling the Attack: Trace Analysis

In the following, we build up the attack from scratch. In this way, while we are developing an attack which is very specific to our target, we show the methods you would use to build an attack for a different target.

Let's start by looking at a single trace. Let's start with the first 20k cycles only (you can plot the full trace but that will be very slow because it's a long trace!).

In [15]:
```
from bokeh.plotting import figure, show
from bokeh.resources import INLINE
from bokeh.io import output_notebook

output_notebook(INLINE)
```

BokehJS 2.3.1 successfully loaded.

In [16]:
```
p = figure(plot_width=2000)

samples = 20000
#samples = len(traces[0].wave)
xrange = range(samples)
p.line(xrange, traces[0].wave[:samples], line_color="red")
```

Out[16]: **GlyphRenderer**(id = '1037', ...)

In [17]:
```
show(p)
```

0.1

Detailed write-up:
https://eprint.iacr.org/2021/1520.pdf

Full notebooks:
https://github.com/newaetech/chipwhisperer-jupyter/blob/master/demos/CW305_ECC

More Stuff

Follow me on Twitter:

@colinoflynn

Send me an e-mail:

coflynn@newae.com

Occasional blog posts:

www.oflynn.com

Updates on new tools & tutorials:

www.newae.com (subscribe to newsletter)

www.chipwhisperer.com