# Glitching and Side-Channel Analysis for All

Colin O'Flynn – NewAE Technology Inc.

RECON 2015 – Montreal, QC.

# Overview

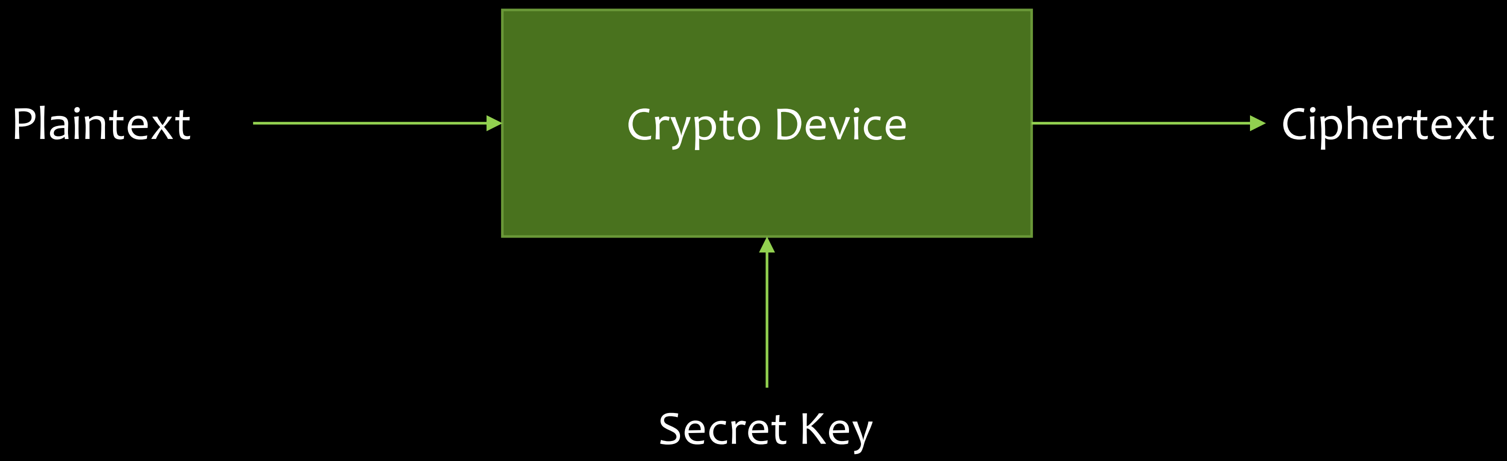
- W.t.f is side-channel power analysis (again)
- Example: IEEE 802.15.4 Node
- Example: AES-256 Bootloader
- W.t.f. is Glitching
- Simple power glitching

# About Me

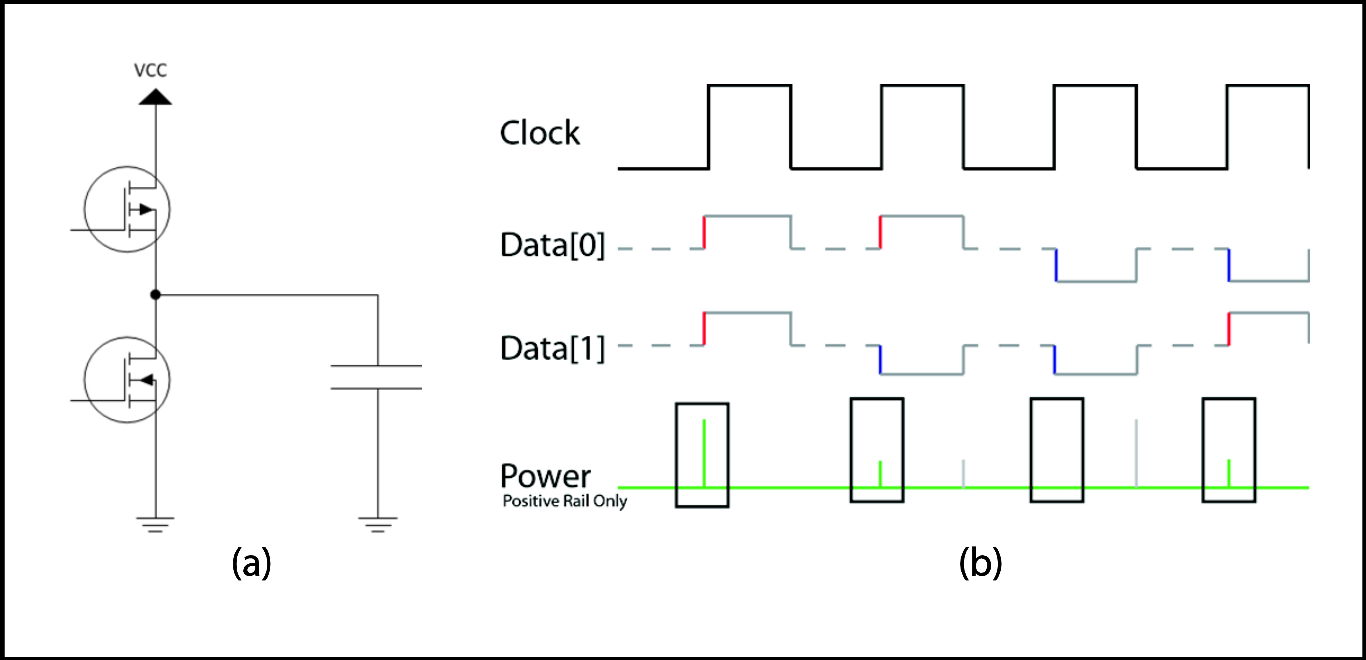
- PhD at Dalhousie University in Halifax, Canada (Ongoing)
- Designed open-source hardware security project (ChipWhisperer)
- Commercialization through NewAE Technology Inc.
- Previously talked at Blackhat US/EU/AD, RECON, ESC

# Side Channel Power Analysis

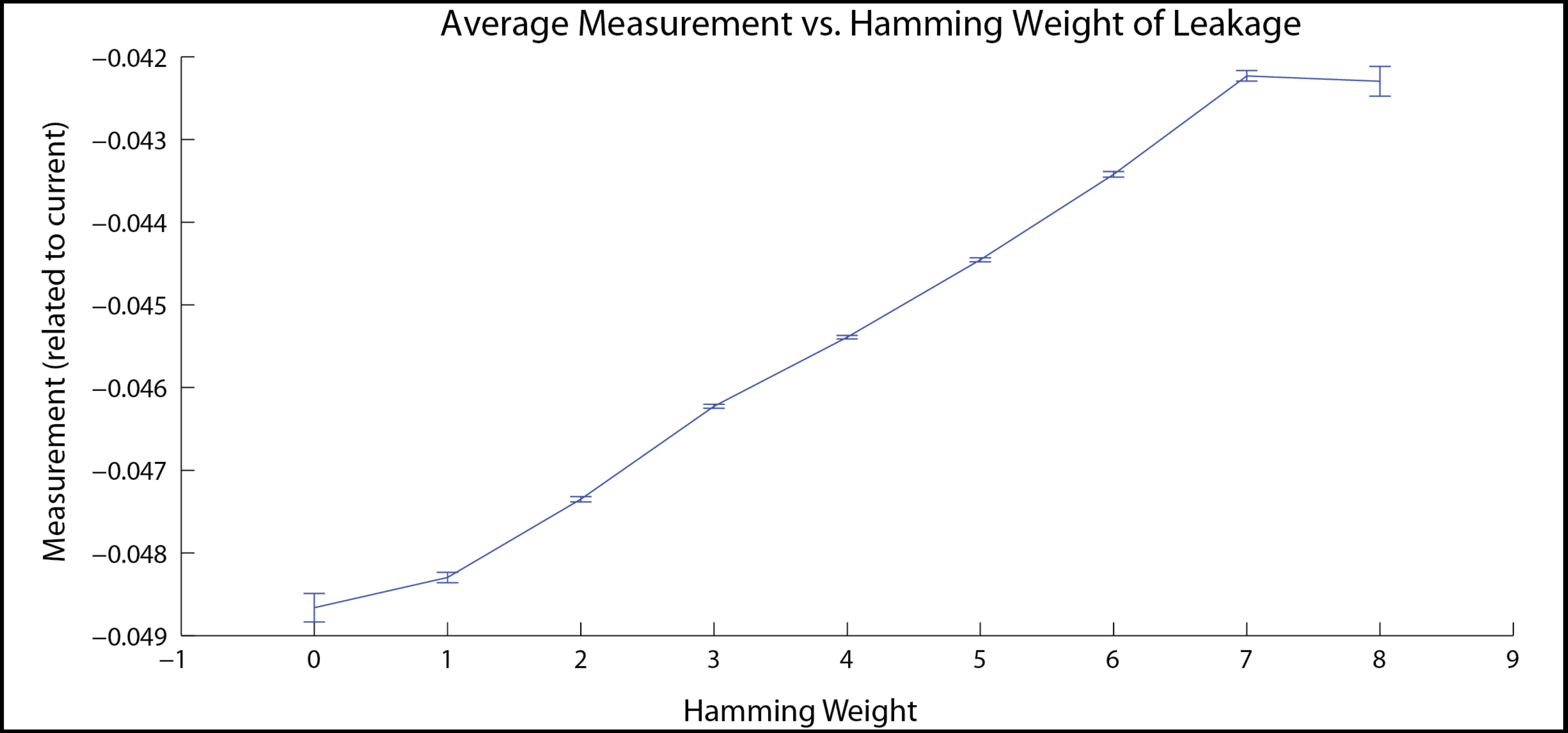
# Side Channel Analysis



# Super-Fast Side Channel

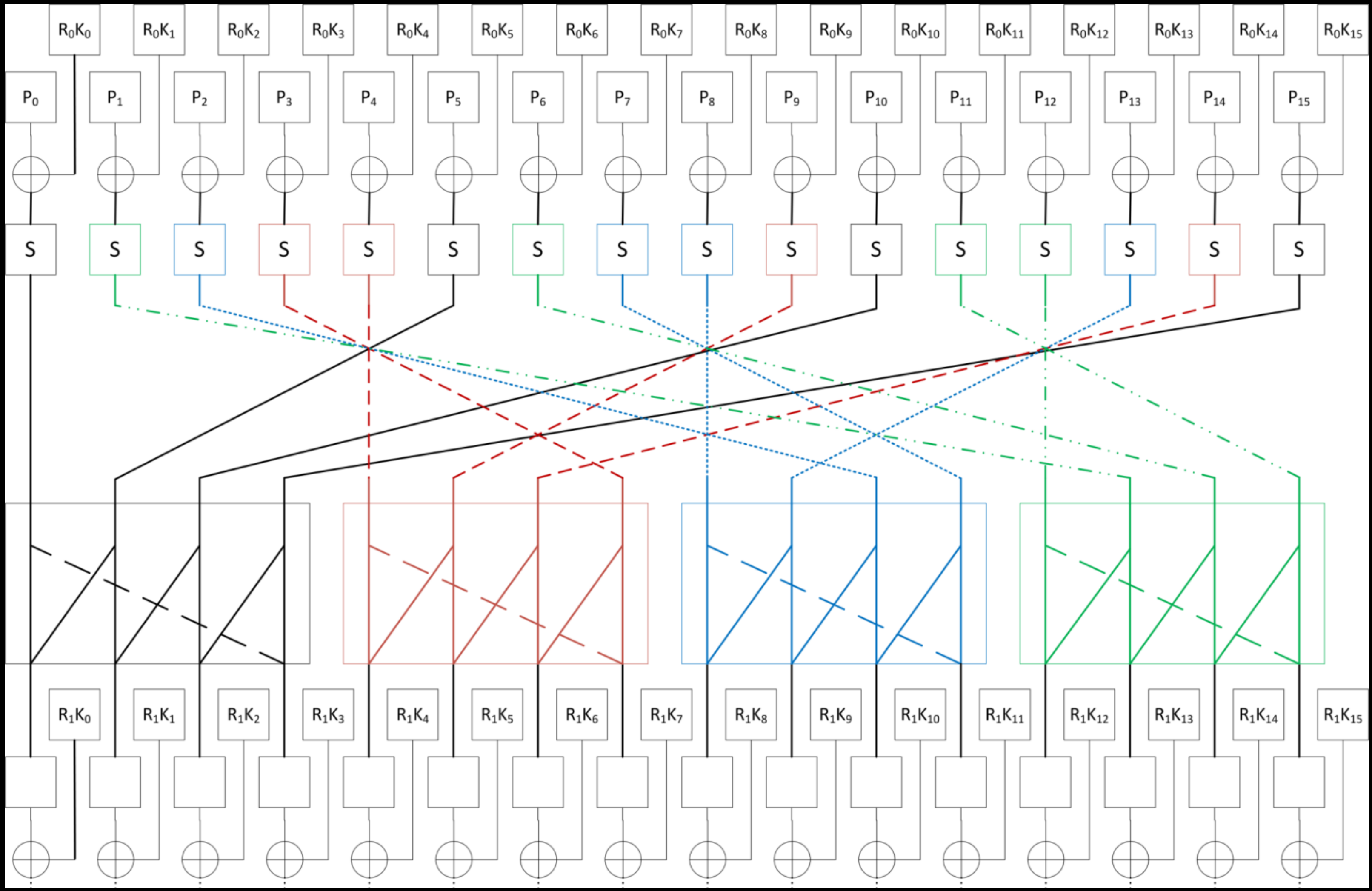


# Real-Life



Average Measurement vs. Hamming Weight of Leakage
Measurement (related to current)
−0.042
−0.043
−0.044
−0.045
−0.046
−0.047
−0.048
−0.049
−1
0
1
2
3
4
5
6
7
8
9
Hamming Weight

# Breaking Apart



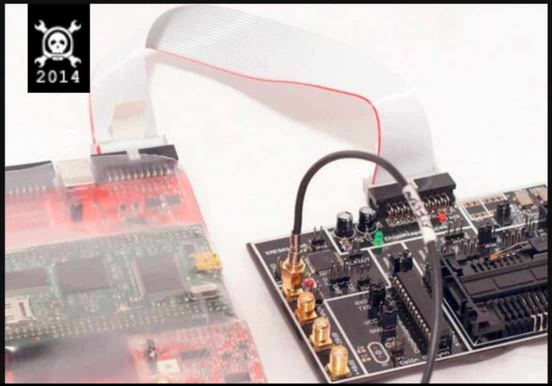
# Hardware Example

# Hackaday Prize 2014

## ChipWhisperer®: Security Research

ChipWhisperer laughs at your AES-256 implementation. But it laughs with you, not at you.

coflynn




**DESCRIPTION**

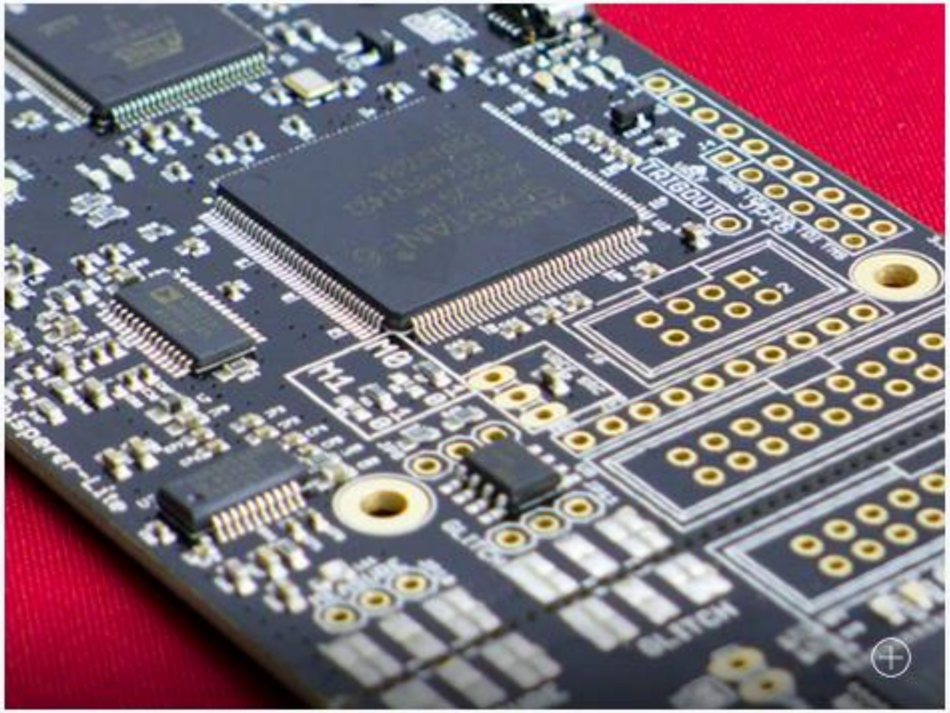
ChipWhisperer is the first open-source toolchain for embedded hardware security research including side-channel power analysis and glitching. The innovative synchronous capture technology is unmatched by other tools, even from commercial vendors. Similar commercial equipment is too expensive ($30k+), and being closed-source limits usefulness for academics. Instead this project bridges the gap between academic research and in-the-trenches engineering. Several peer-reviewed publications describe the design, matched with hours of hands-on tutorials for getting started.

The objective of ChipWhisperer is nothing short of revolutionizing the entire embedded security industry. Every designer who uses encryption in their design should be able to perform a

# Cheap Hardware… First Ver

# ChipWhisperer-Lite Kickstarter

## ChipWhisperer-Lite: A New Era of Hardware Security Research



Embedded security - is it an oxymoron? Learn the truth through a series of hands-on labs targeting computer and electrical engineers.

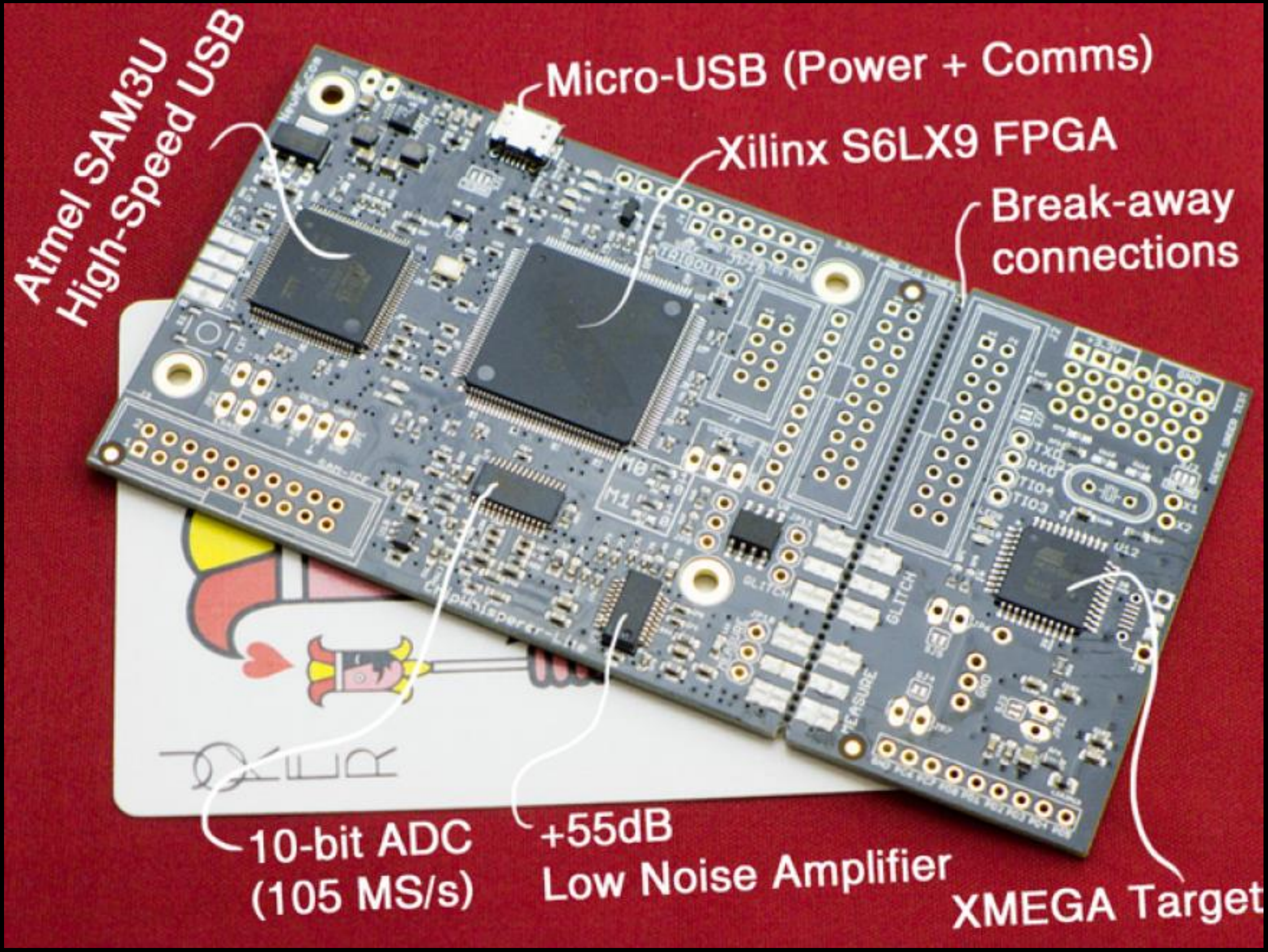
Add link

Created by

Colin O'Flynn

**331 backers** pledged $88,535 to help bring this project to life.

# Cheaper Hardware



# Open-Source Software

**chipwhisperer** GIT | HTTP | GIT Read-only | git@git.assembla.com:chipwhisperer.git

Branch master / LICENSE.txt

LICENSE.txt

```
Certain files in this project may have specific licenses or copyright
restrictions, as this project uses multiple open-source projects.
Files in this project without any specific license can be assumed
to follow the following general clause:

ChipWhisperer is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>;.
```

# Example of Power Analysis

<demo here>

# IEEE 802.15.4 Nodes

# IEEE 802.15.4

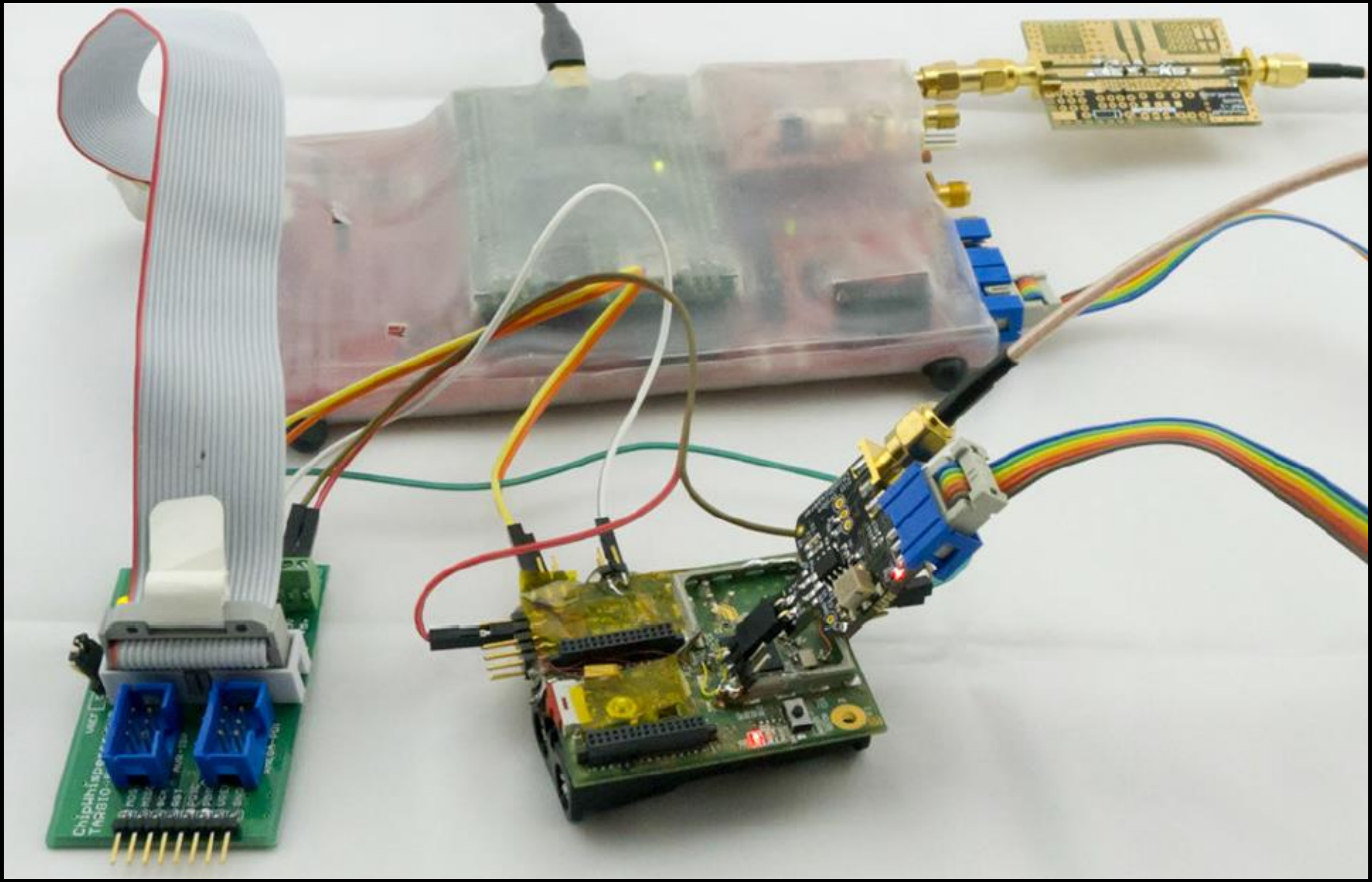
# Example #1: 802.15.4

ZigBee (ZigBee IP, ZigBee Pro, RF4CE, etc.)
WirelessHART
MiWi
ISA100.11a
6LoWPAN
Nest Weave
JenNet
Thread
Atmel Lightweight Mesh
IEEE 802.15.5
DigiMesh

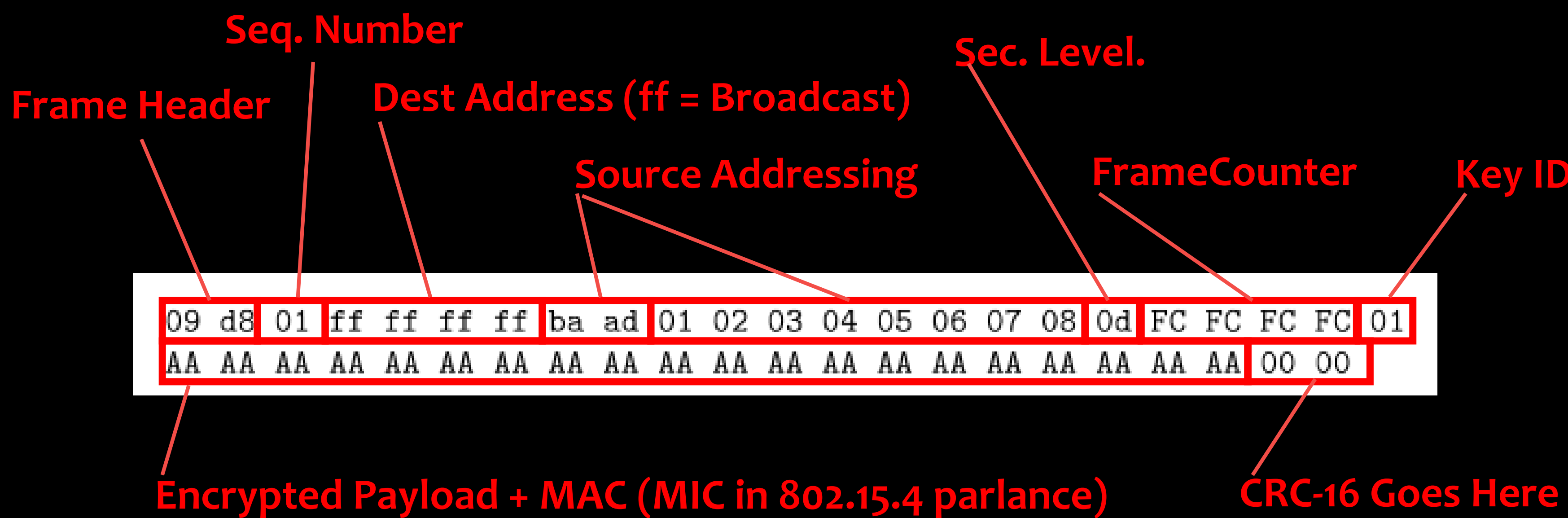
802.15.4 Node

http://eprint.iacr.org/2015/529

# Hardware Setup



# 802.15.4 Frame Format

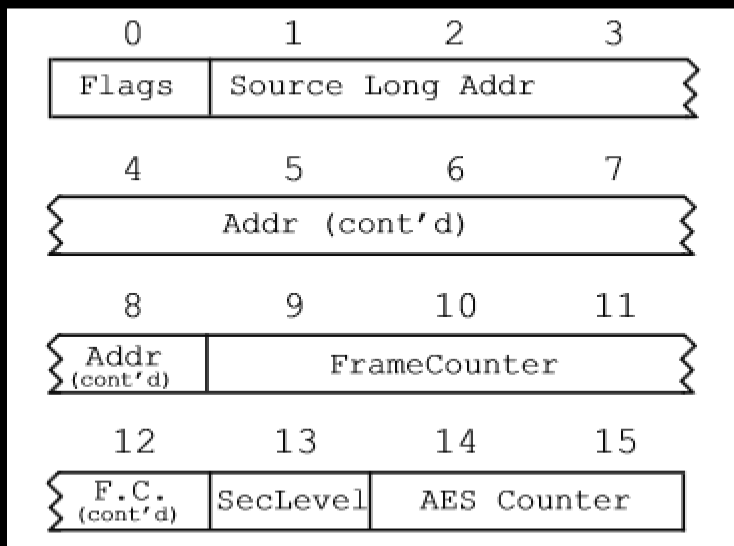


# 802.15.4 Decoding

IEEE 802.15.4 Wireless Stack: Frame Decryption Procedure:

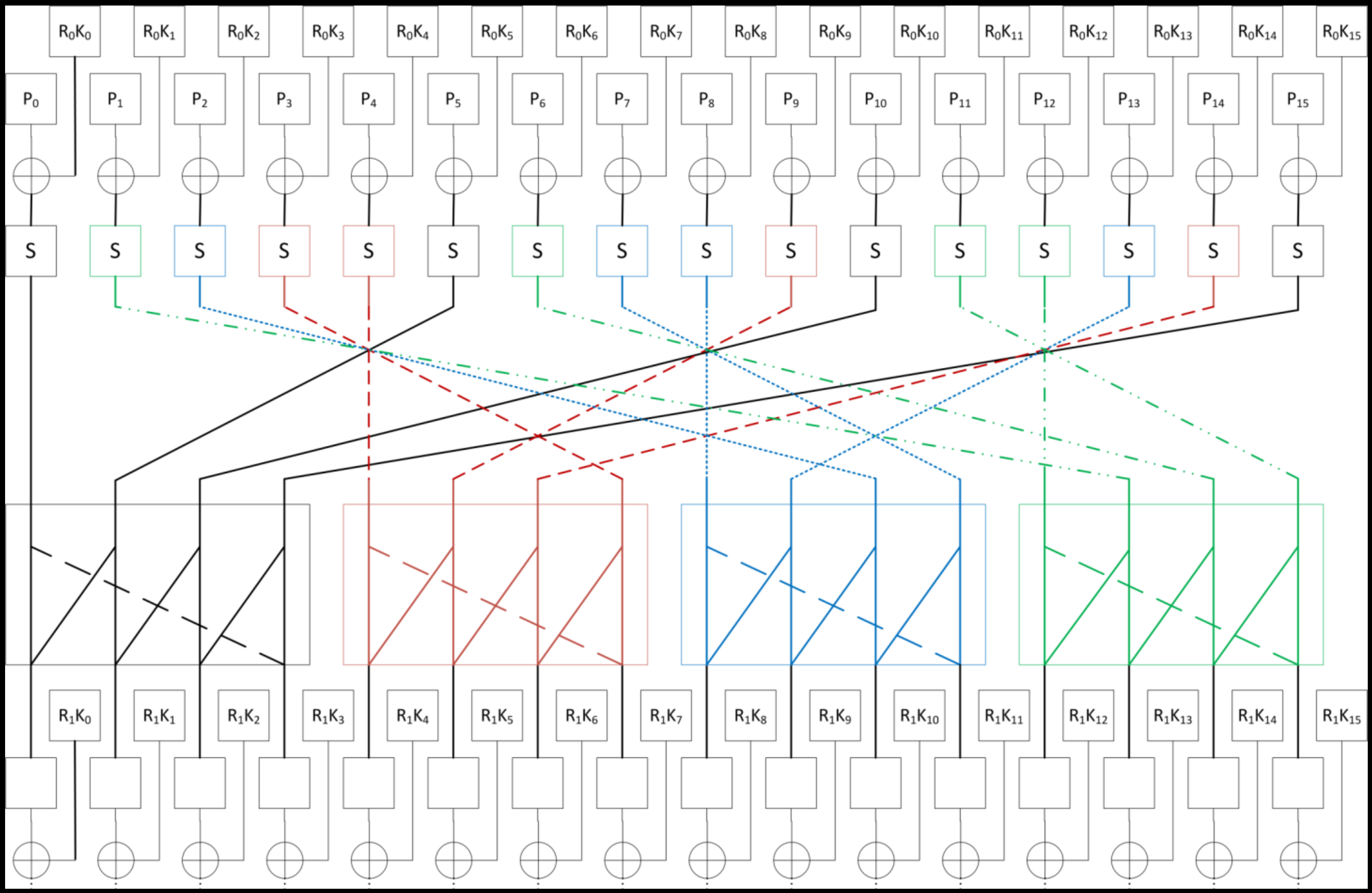
1. Validate headers and security options.
2. Check that the received frame counter is numerically greater than the last stored frame count.
3. Look up the secret key based on message address and/or key index.
4. Decrypt the payload (and MAC if present).
5. Validate the MAC (if present).
6. Store the frame counter.

# Example #1: 802.15.4




Input to AES Block

# Many fixed bytes…



# CPA Attack Result

$$\mathbf{p}^1 = [\mathtt{c\ c\ c\ c\ c\ c\ c\ c\ c\ X\ X\ X\ X\ c\ c\ c}]$$

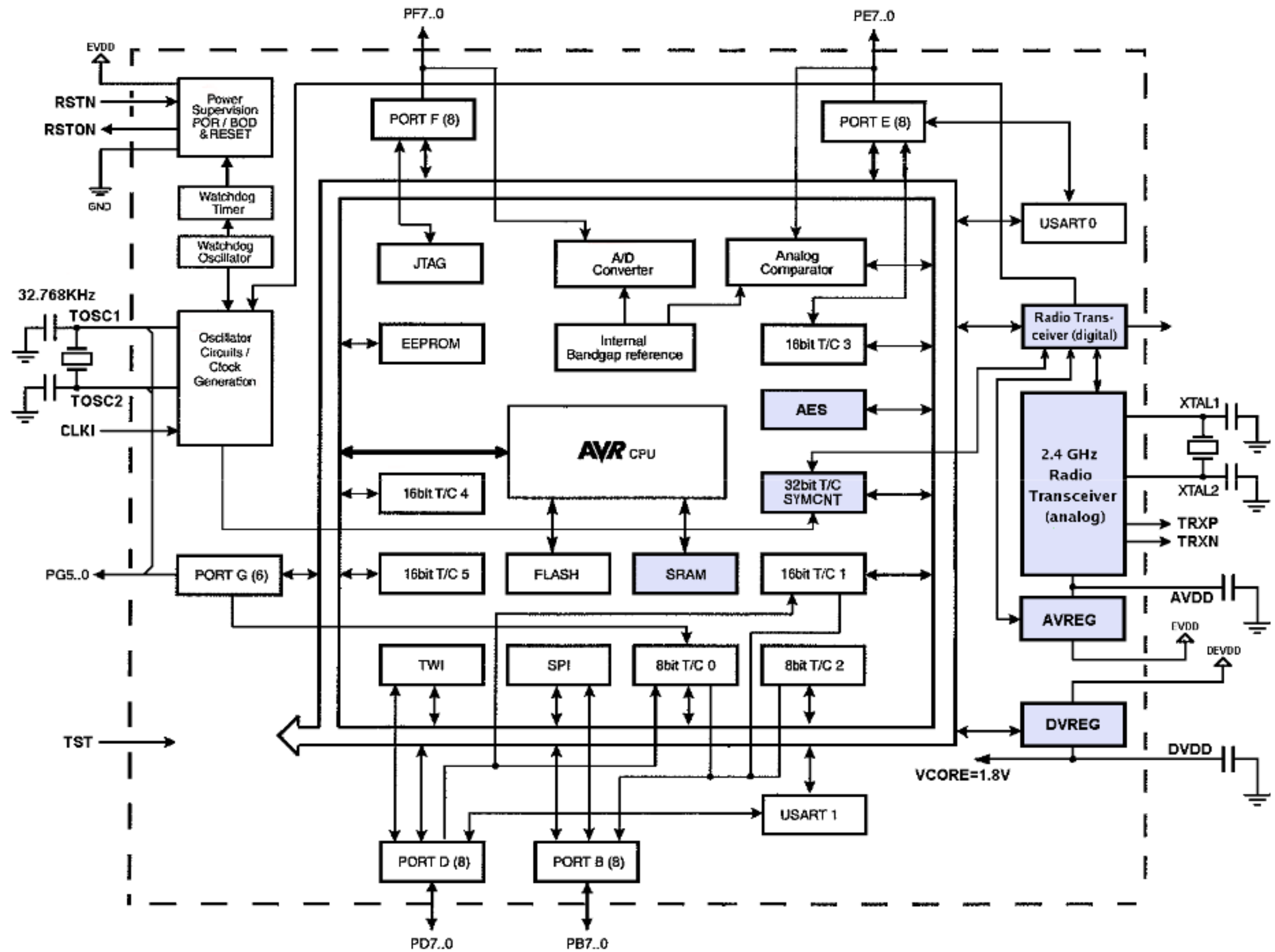
$$\mathbf{r}^1 = [\mathtt{c\ c\ c\ c\ c\ c\ c\ c\ c\ K{*}K{*}K{*}K{*}\ c\ c\ c}]$$

$$\mathbf{v}^1 = [\mathtt{c\ c\ Y{*}\ c\ c\ Y{*}\ c\ c\ c\ c\ c\ c\ Y{*}\ c\ c\ Y{*}}]$$

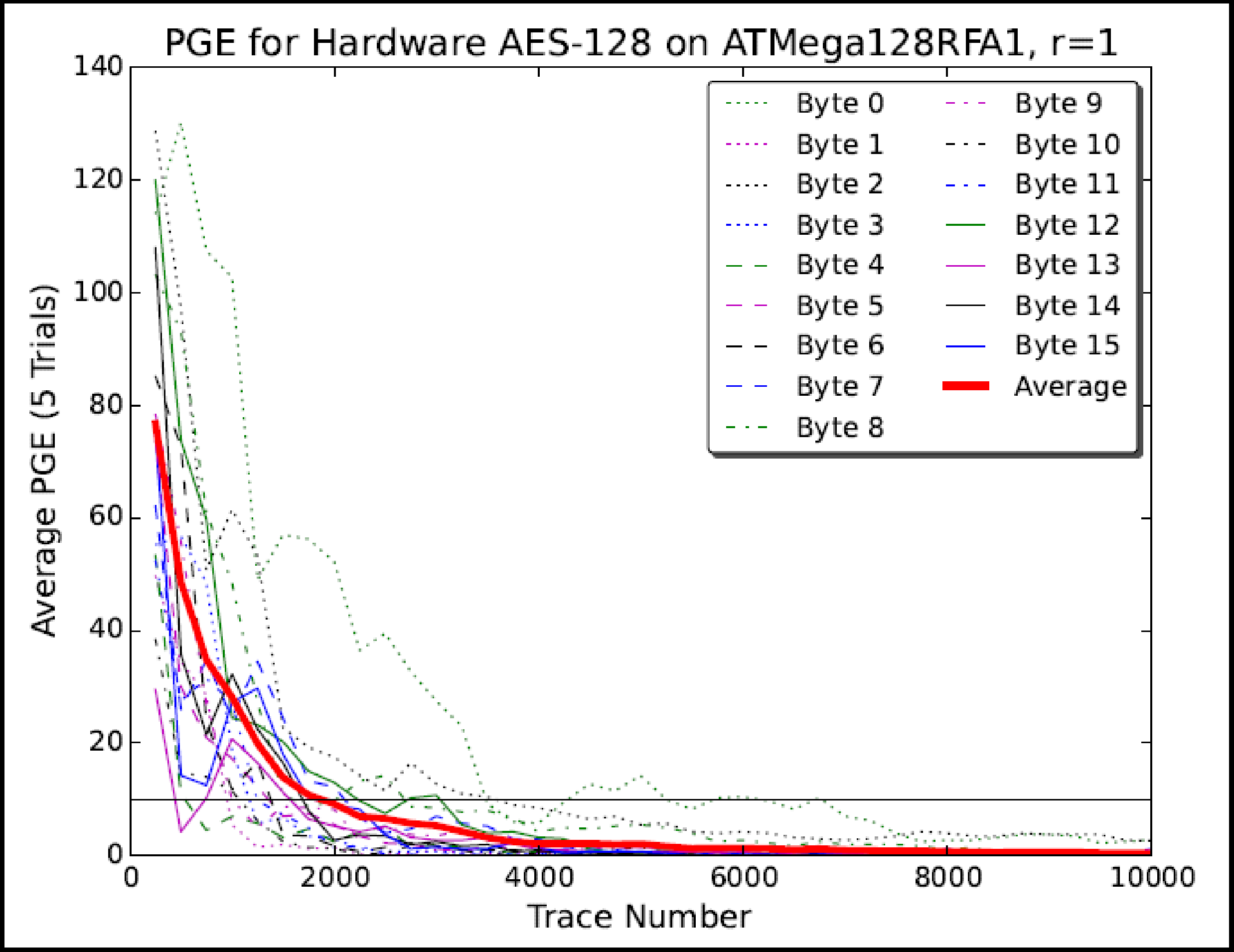
$$\mathbf{m}^1 = [\mathtt{Z{*}Z{*}Z{*}Z{*}Z{*}Z{*}Z{*}Z{*}\ c\ c\ c\ c\ Z{*}Z{*}Z{*}Z{*}}]$$

# ATMega128RFA1

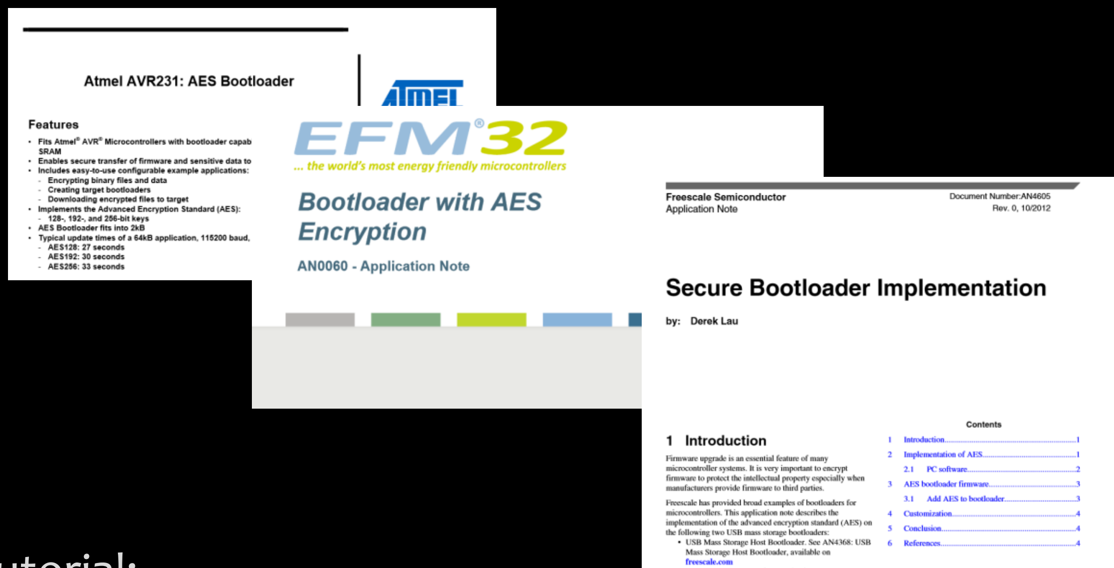
Figure 3-1 Block Diagram



# ATMegaRF AES Peripheral



# Example #2: AES-256 Bootloader



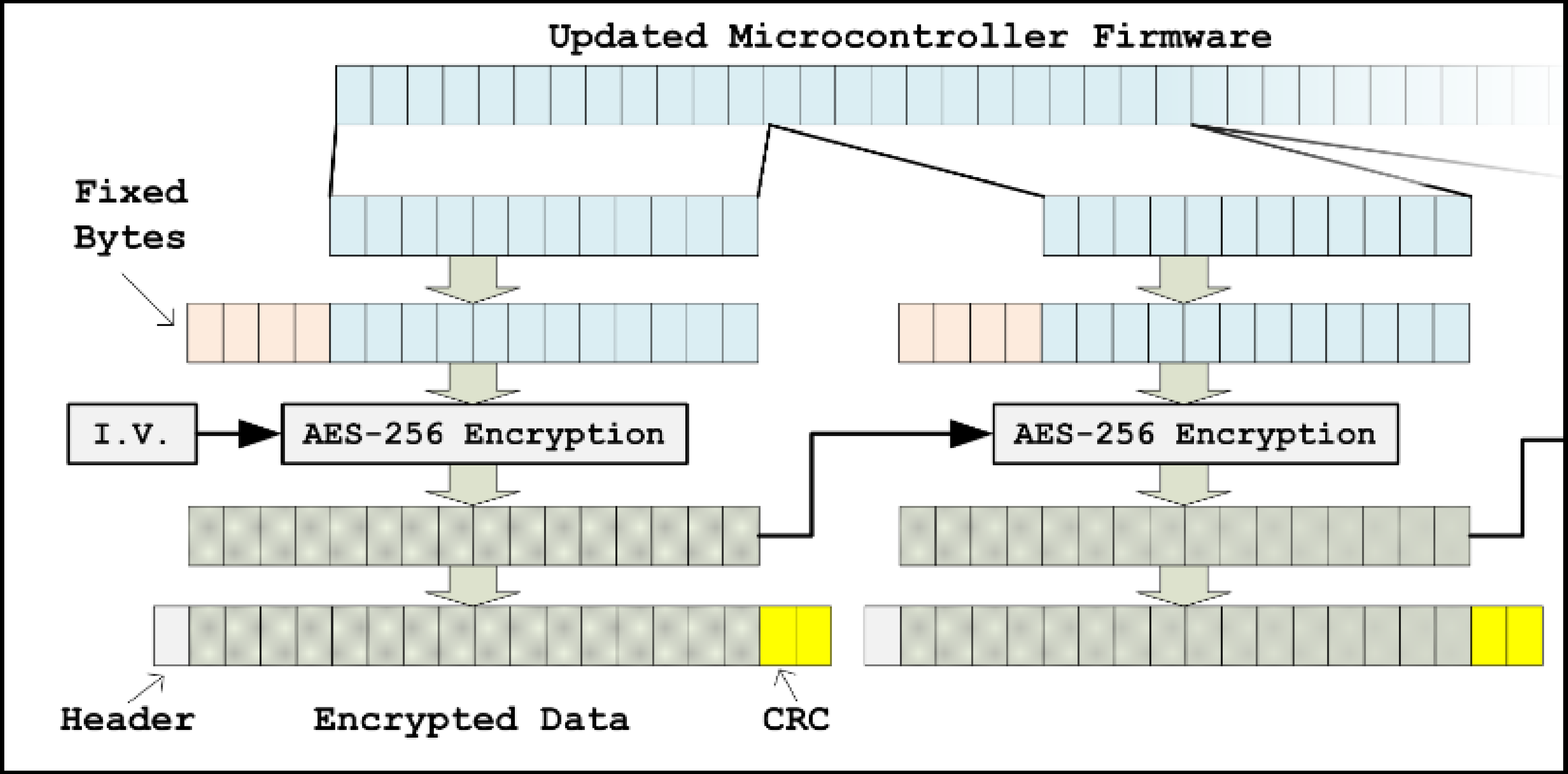

Tutorial:

http://newae.com/sidechannel/cwdocs/tutorialaes256boot.html

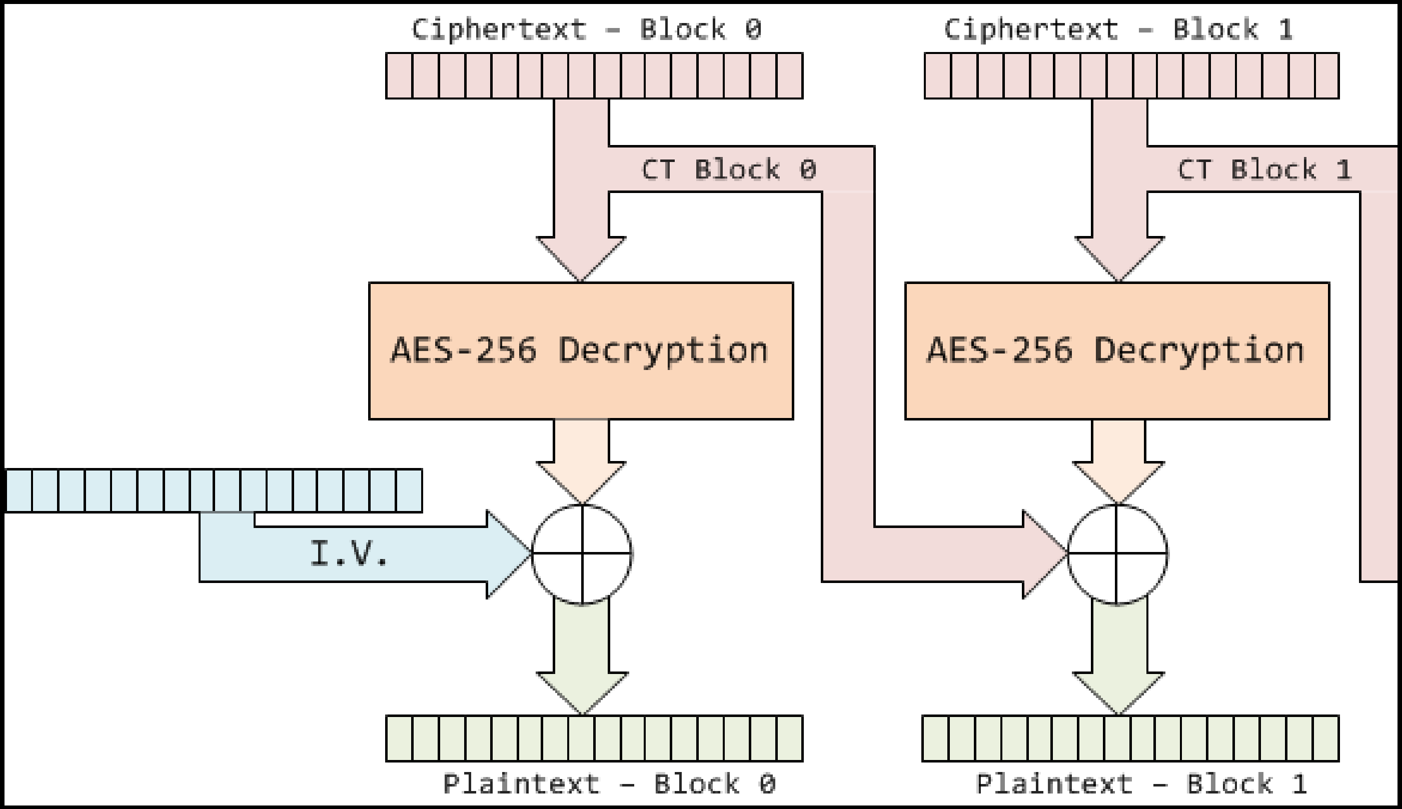
Paper (CCECE 2015):

https://eprint.iacr.org/2014/899.pdf

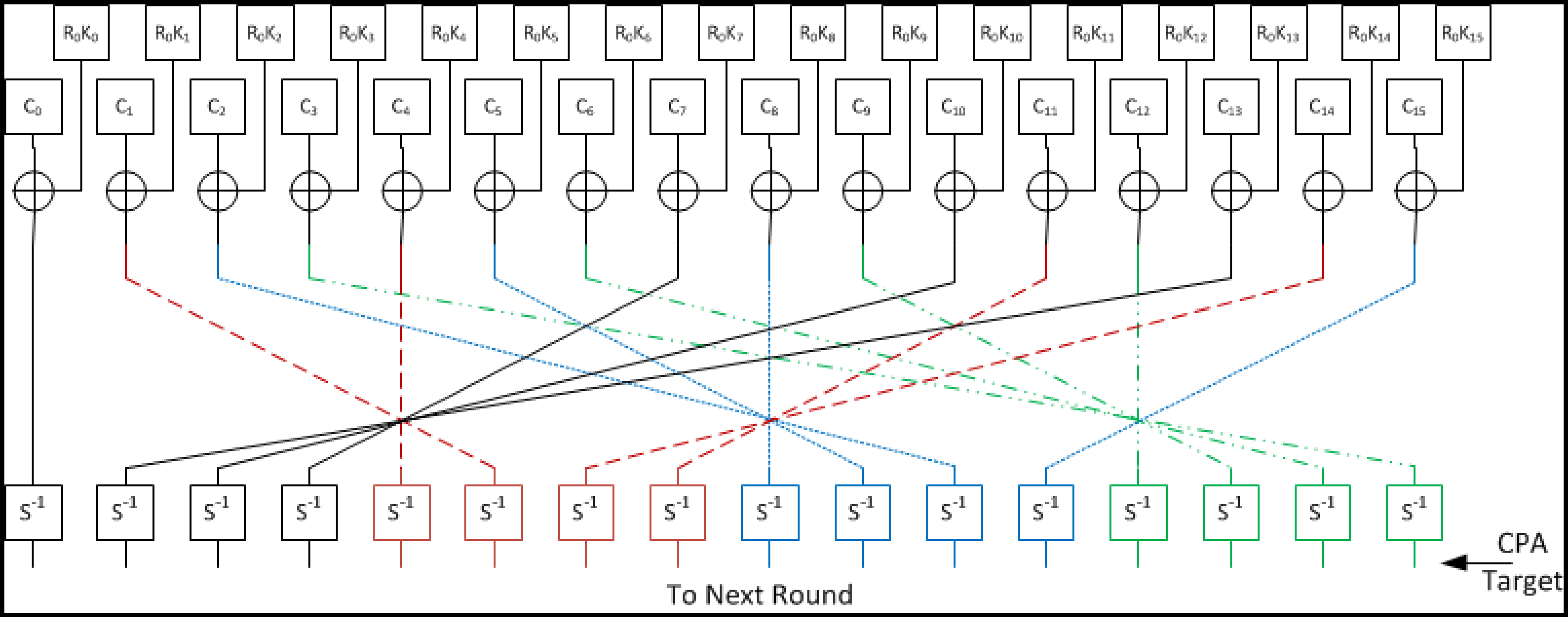
# Bootloader Protocol



# AES-256 in CBC Mode

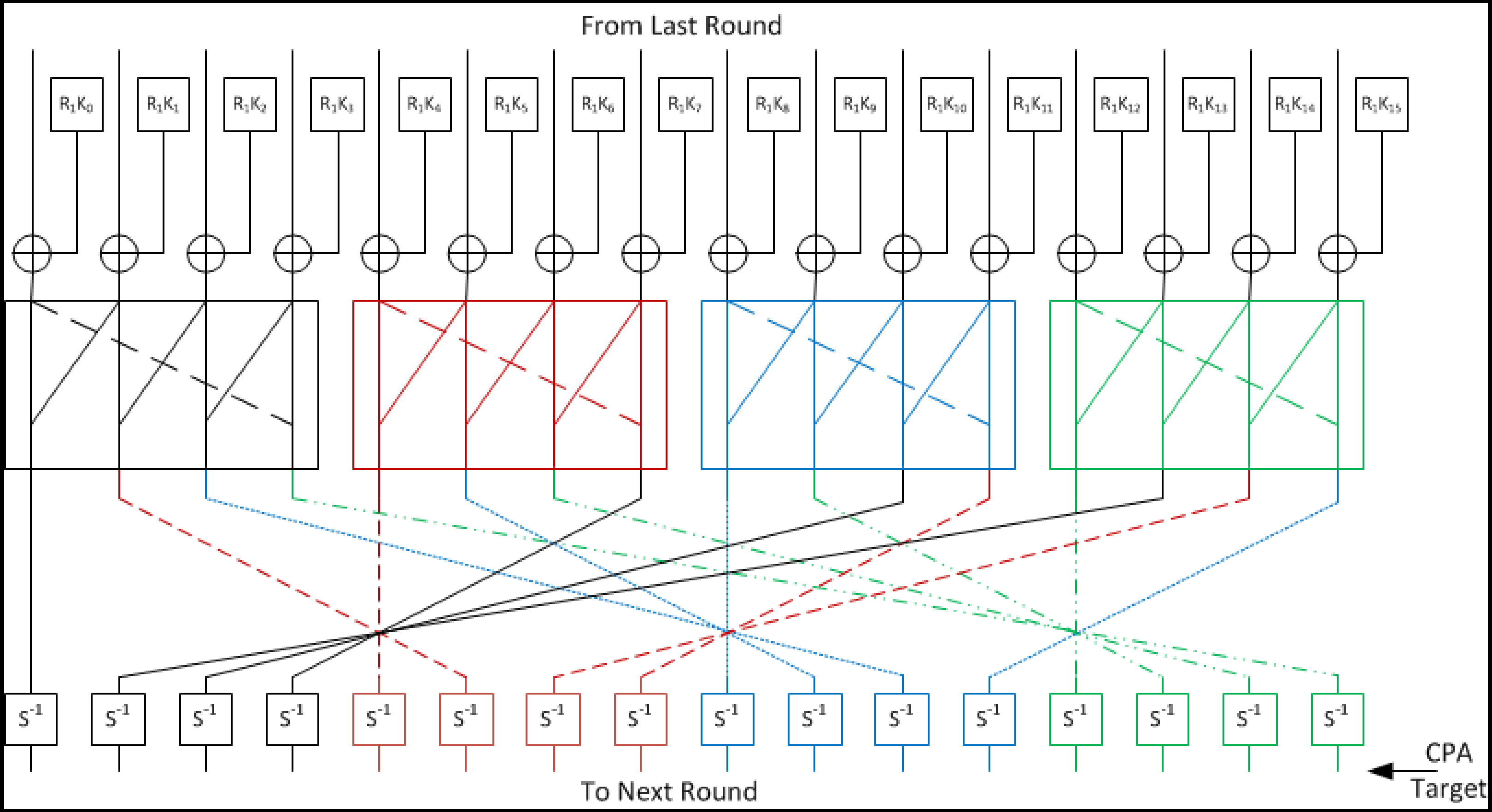


# Round 14



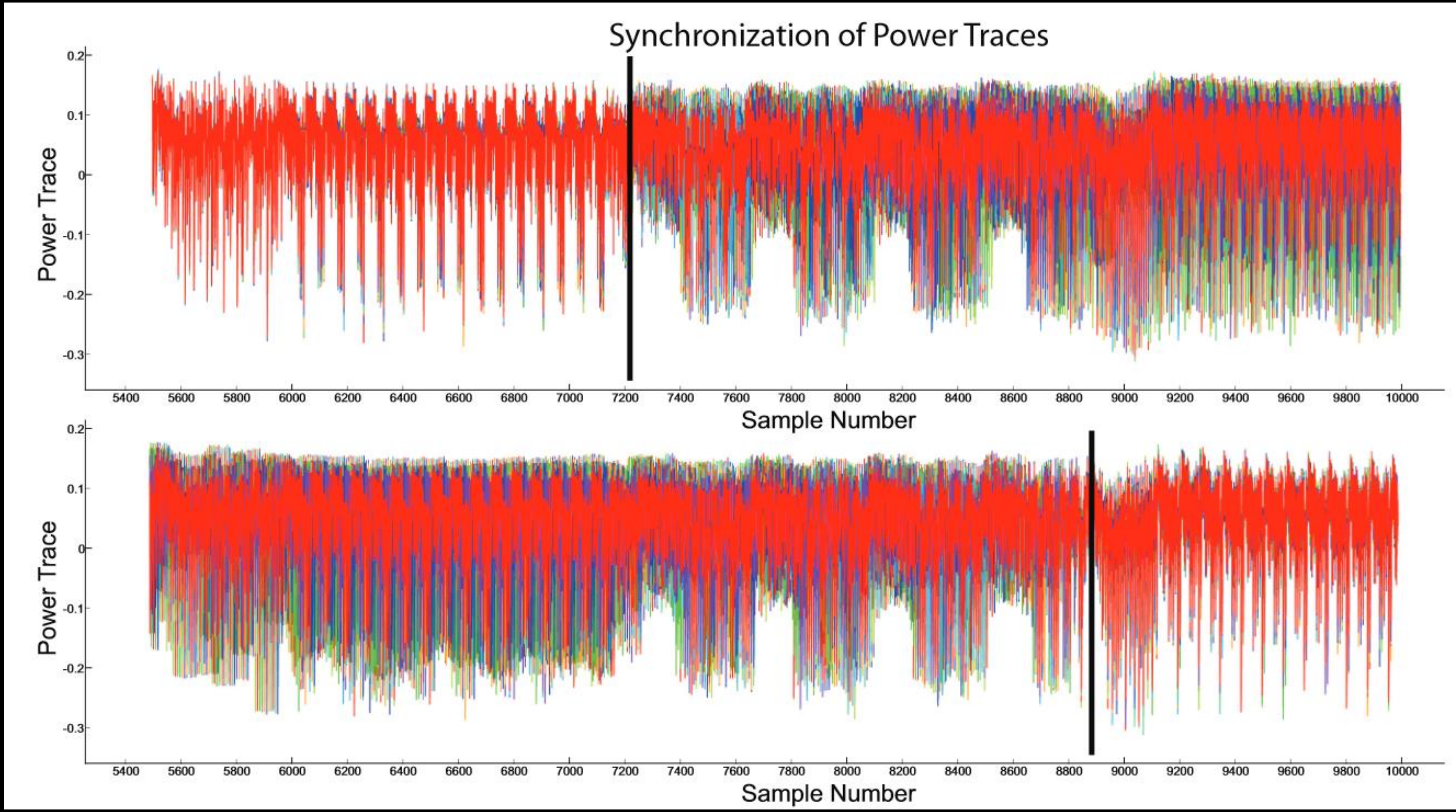
$R_0K_0$ $R_0K_1$ $R_0K_2$ $R_0K_3$ $R_0K_4$ $R_0K_5$ $R_0K_6$ $R_0K_7$ $R_0K_8$ $R_0K_9$ $R_0K_{10}$ $R_0K_{11}$ $R_0K_{12}$ $R_0K_{13}$ $R_0K_{14}$ $R_0K_{15}$
$C_0$ $C_1$ $C_2$ $C_3$ $C_4$ $C_5$ $C_6$ $C_7$ $C_8$ $C_9$ $C_{10}$ $C_{11}$ $C_{12}$ $C_{13}$ $C_{14}$ $C_{15}$
$S^{-1}$
CPA Target
To Next Round

# Round 13



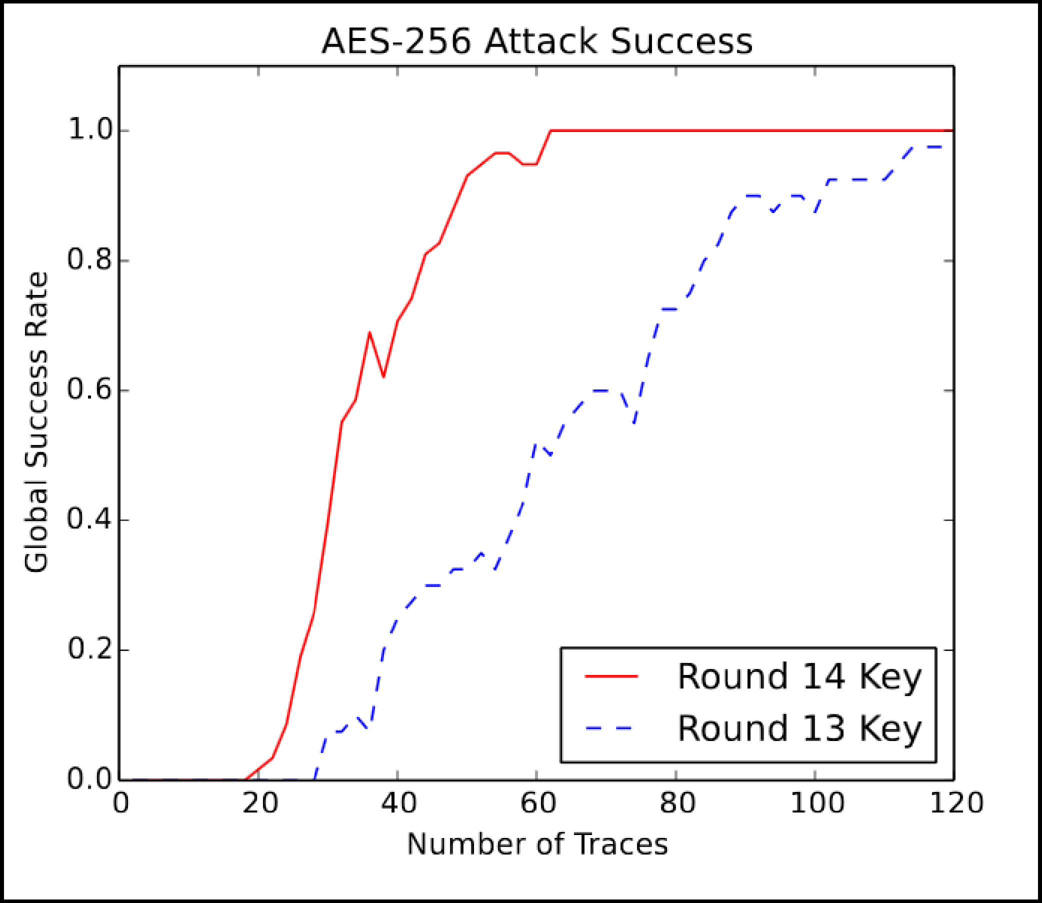
From Last Round
$R_1K_0$
$R_1K_1$
$R_1K_2$
$R_1K_3$
$R_1K_4$
$R_1K_5$
$R_1K_6$
$R_1K_7$
$R_1K_8$
$R_1K_9$
$R_1K_{10}$
$R_1K_{11}$
$R_1K_{12}$
$R_1K_{13}$
$R_1K_{14}$
$R_1K_{15}$
$S^{-1}$
To Next Round
CPA
Target

# Trace View



Synchronization of Power Traces
Power Trace
0.2
0.1
0
-0.1
-0.2
-0.3
5400
5600
5800
6000
6200
6400
6600
6800
7000
7200
7400
7600
7800
8000
8200
8400
8600
8800
9000
9200
9400
9600
9800
10000
Sample Number
Power Trace
Sample Number

# Success Rate



# Getting Started in Side Channel Power

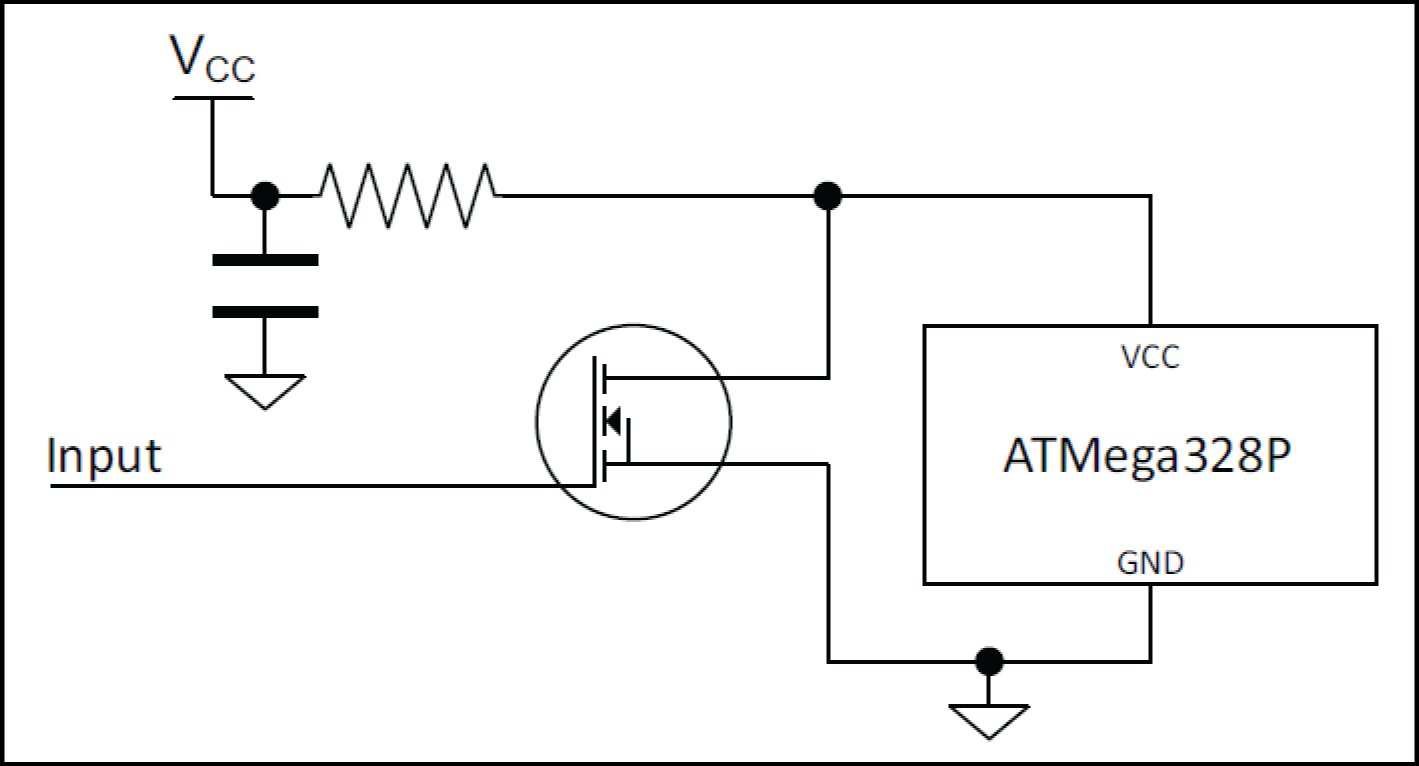
- Build/buy a *simple* target device:
  - AVR dev-board
  - Arduino Uno
  - PIC
- Get a scope with USB API
  - Picoscope
  - Most bench scopes
  - Be wary of cheap off-brand scopes, sometimes USB interface is poor
- Experiment!

# Glitching

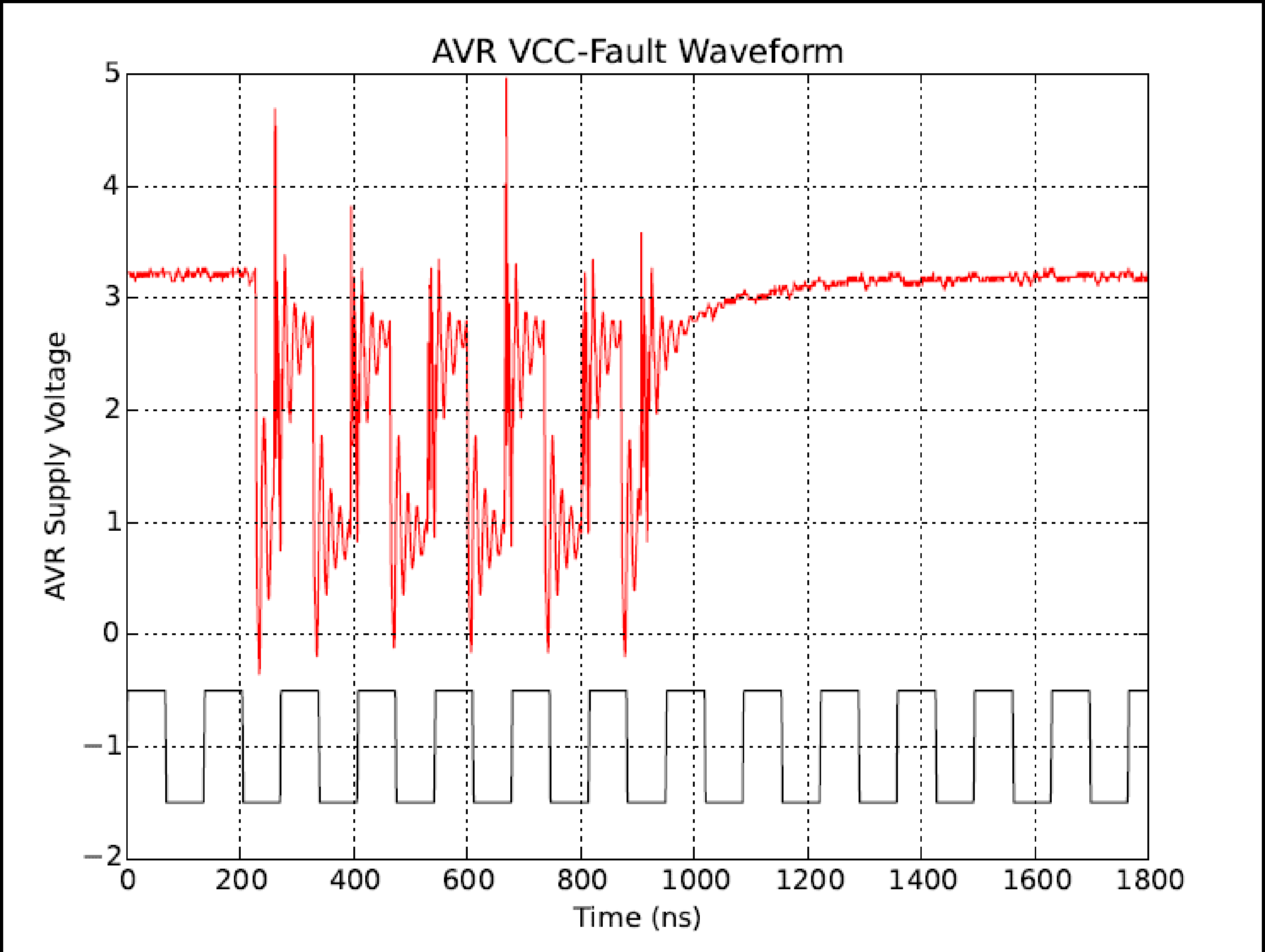
# Glitching Target

```
int i,j,count;

while(1){
    count = 0;

    for (j = 0; j < 5000; j++){
        for (i = 0; i < 5000; i++){
            count++;
        }
    }

    printf("%d %d %d\n", count, i, j);
}
```

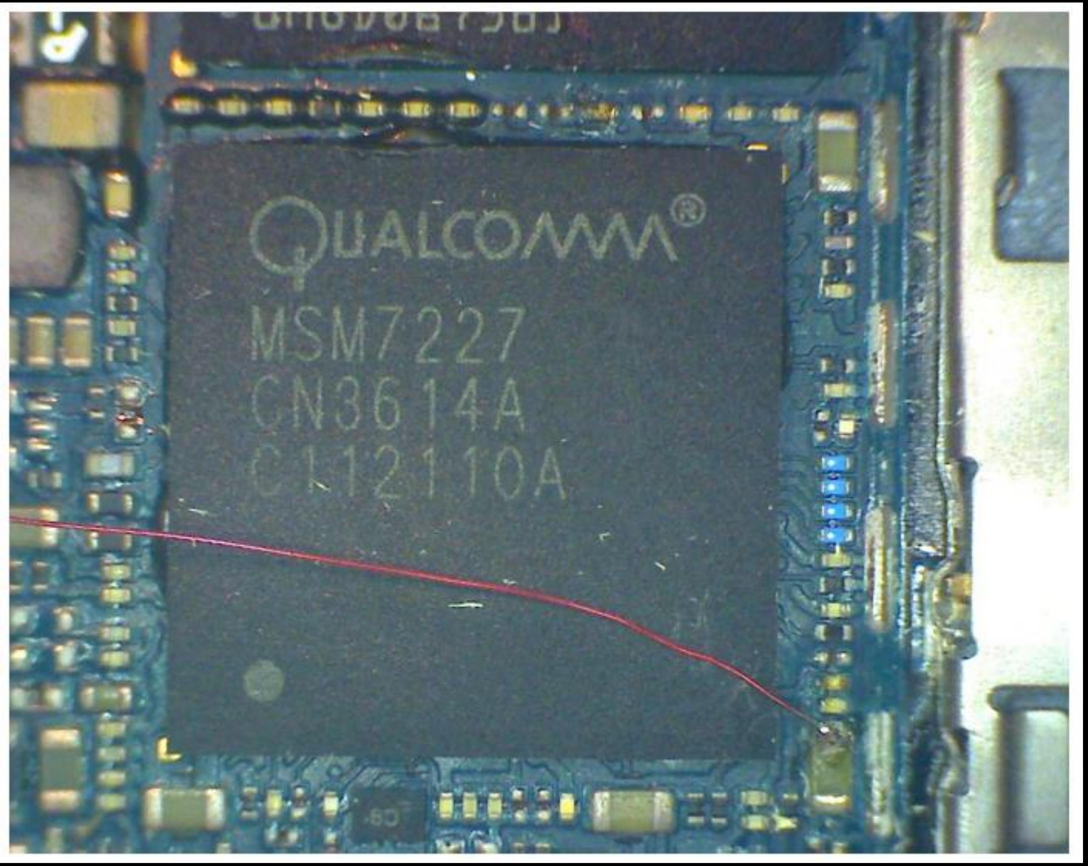
# Easy Glitching



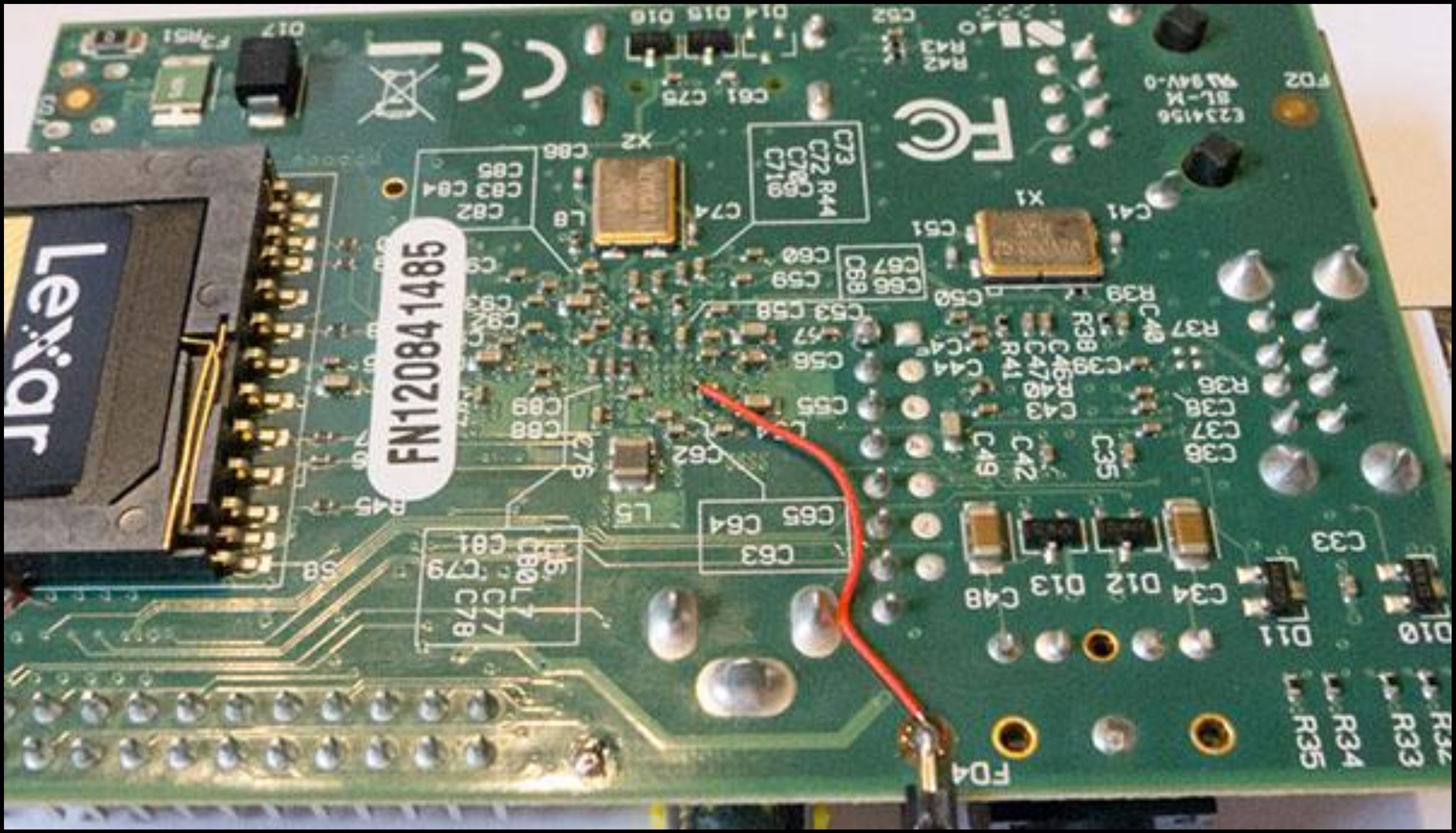
# High-Precision Glitches



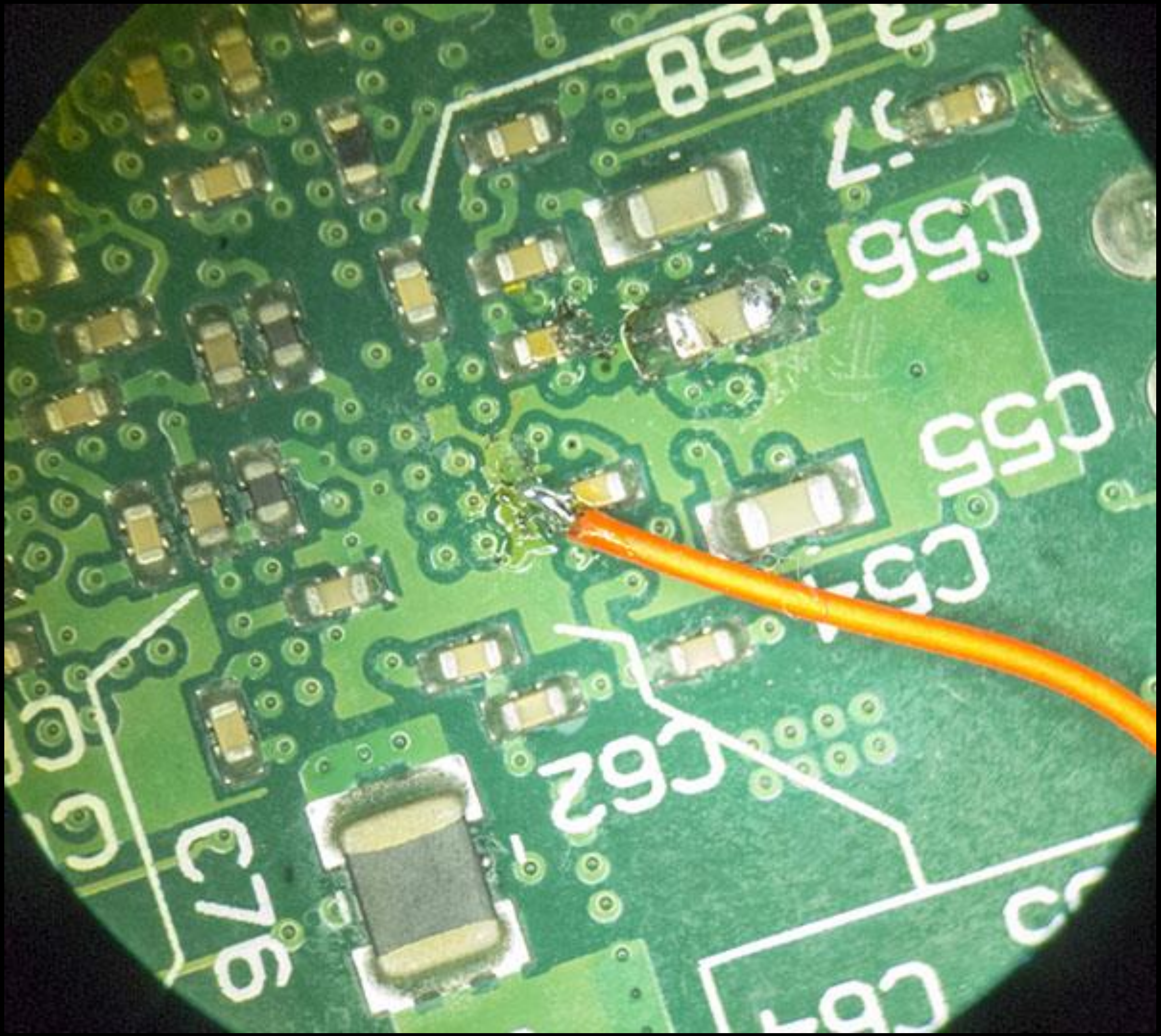
# Easy Glitching



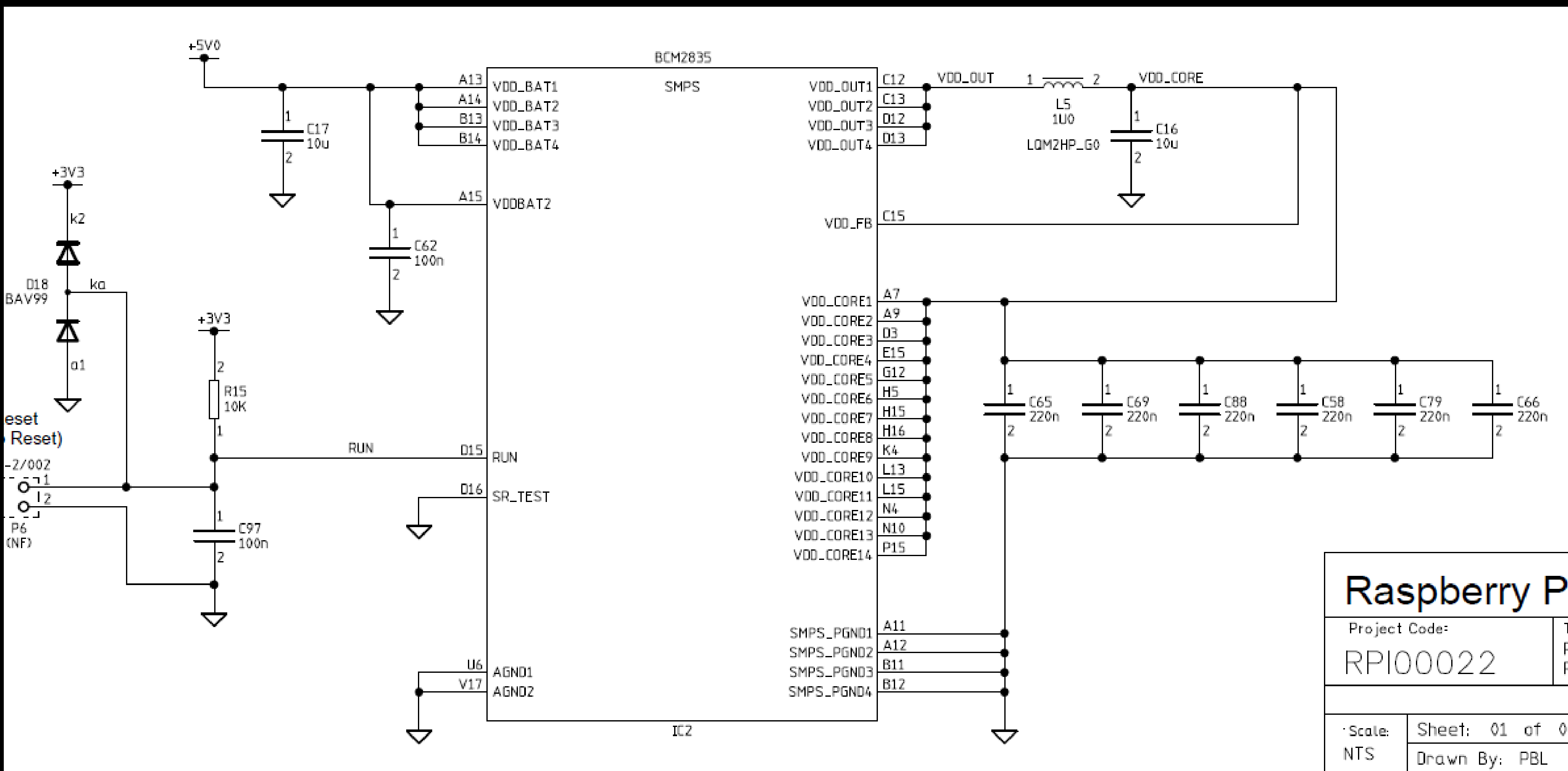
# Raspberry Pi Example



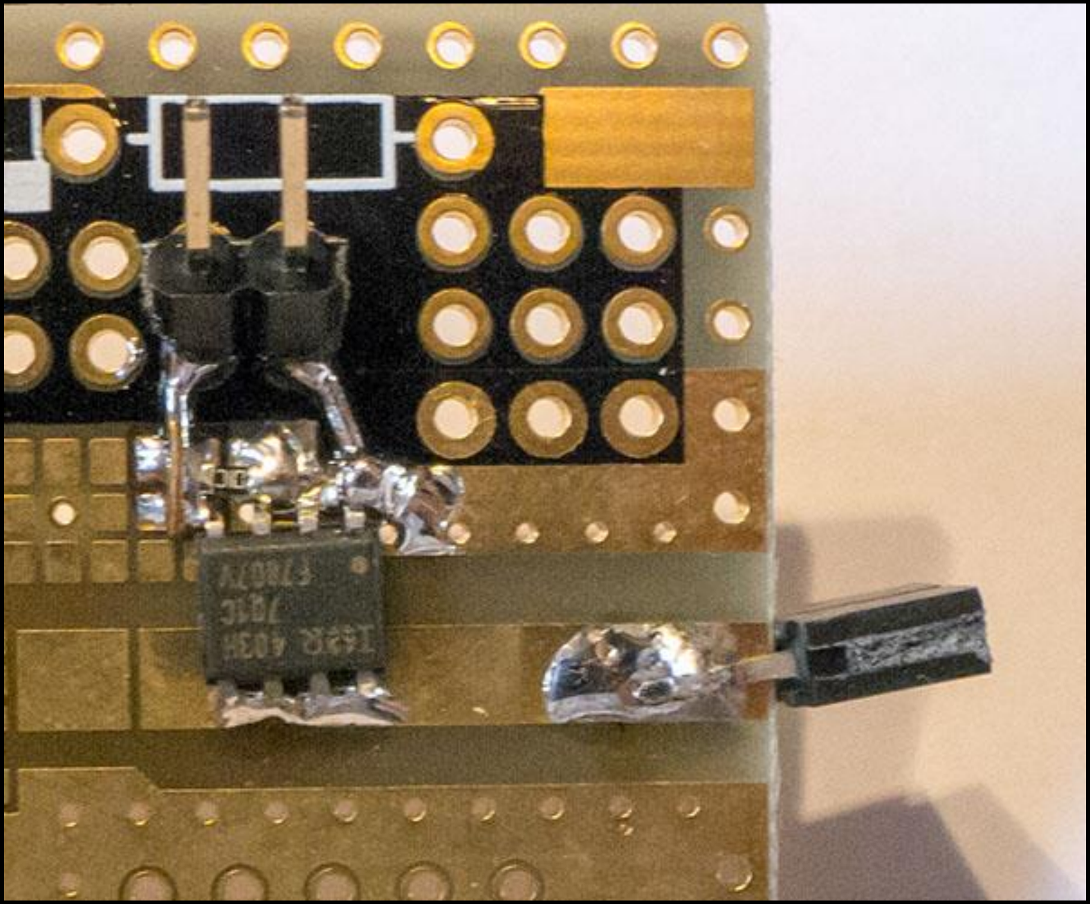
# Raspberry Pi Example



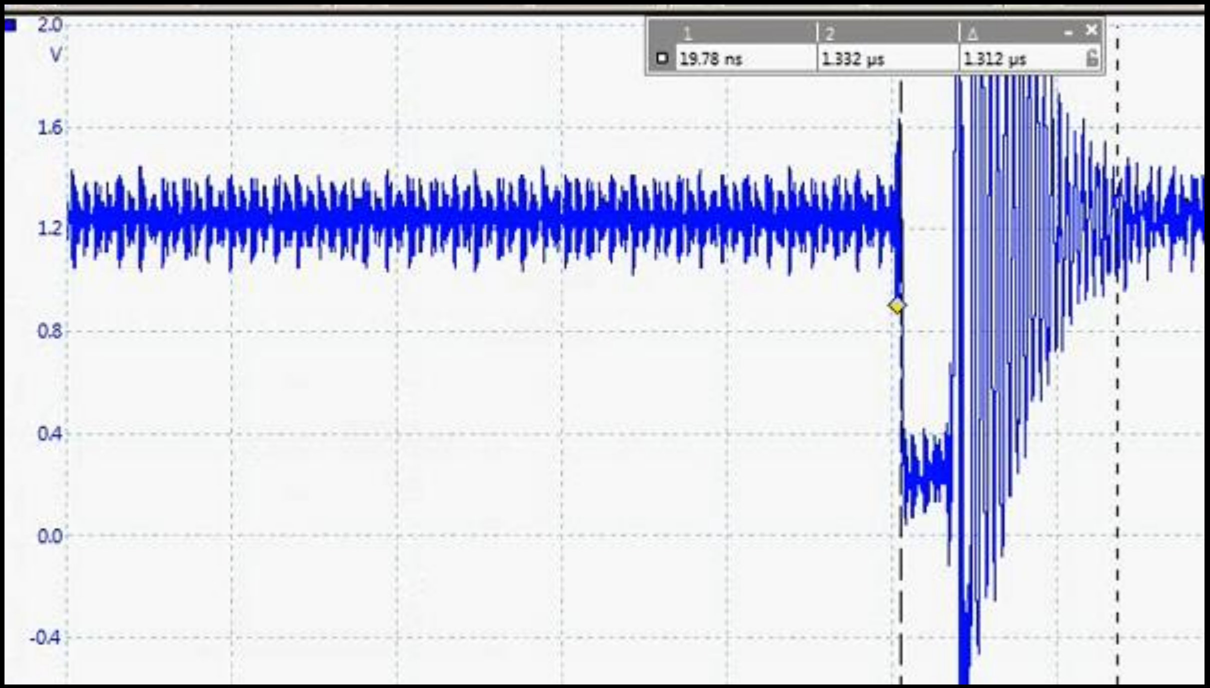
# Raspberry Pi Example



# Glitch Tool



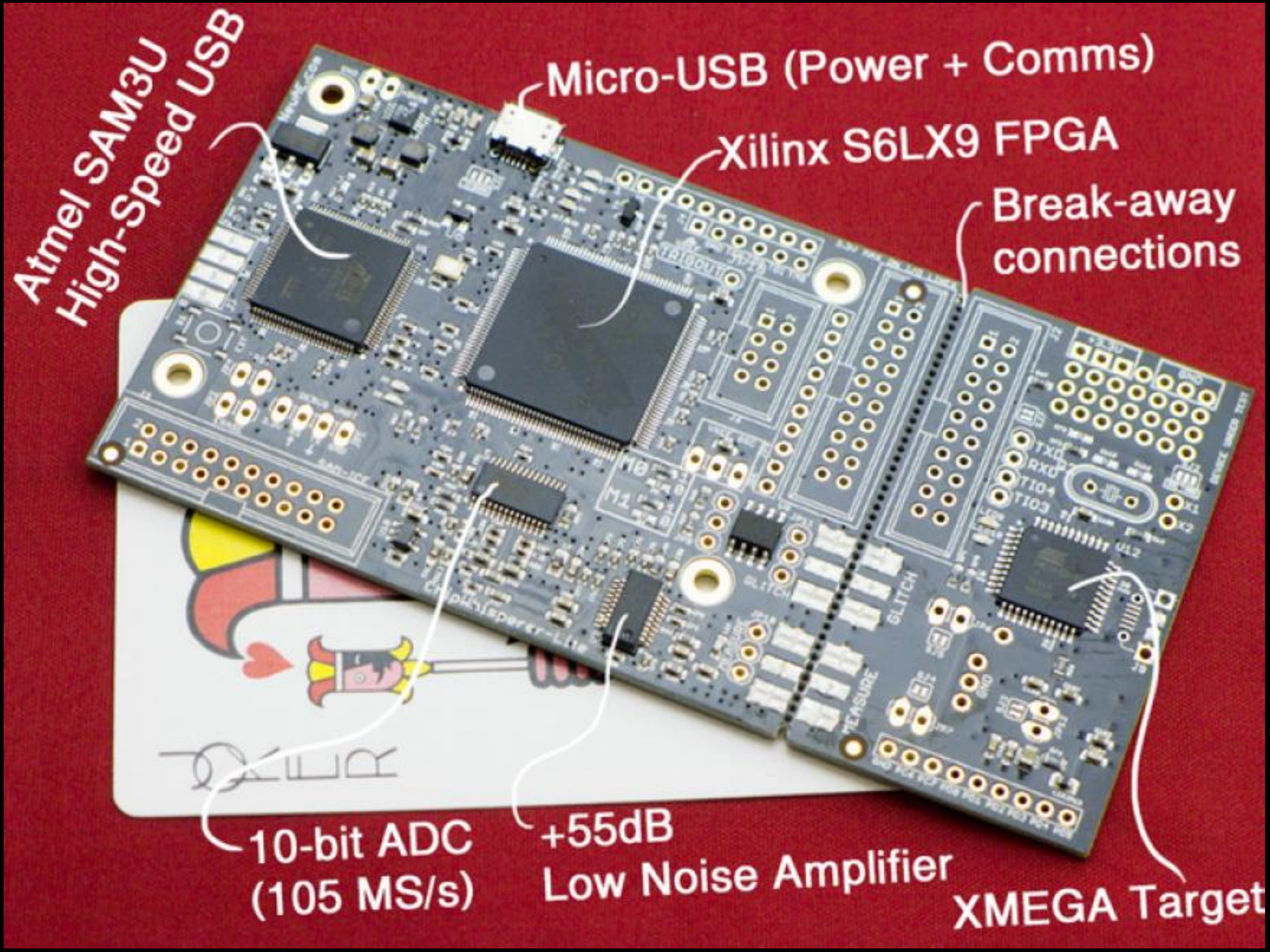
# Glitch Waveform (Raspberry Pi)



# Getting Started in Glitching

- Load simple code onto target
- Determine/guess sensitive power rail
- Test glitch parameters → ideally with profiling code

# Glitching in CW-Lite



# It's fun!

Try Power Analysis and Glitching today!

ChipWhisperer Project: www.chipwhisperer.com

NewAE Technology Inc.: www.newae.com

Personal:

@colinoflynn

coflynn@newae.com

http://www.oflynn.com