

# Breaking Bitcoin Hardware Wallets

Glitches cause stitches!

---

Josh Datko

Chris Quartier

Kirill Belyayev

Updated: 2017/07/07



# Link Drop!

All updated references, notes, links, can be found here:

<https://www.cryptotronix.com/breakingbitcoin>

# The bug that started it all

---

```
1 bool storage_is_pin_correct(const char *pin)
2 {
3     return strcmp(shadow_config.storage.pin,
4                   pin) == 0;
5 }
```

---

On the STM32F205, when the first pin character is wrong it returns in 100ns. When the fourth was wrong, it returned in about 1100ns.

## ? Broken Window Theory for Bugs

| If this was there, what else could we find?

# Initial Attack Plan

1. Send `change_pin` via Python.
2. Watch the return over USB-measure when the PIN failed.
3. Profit?!

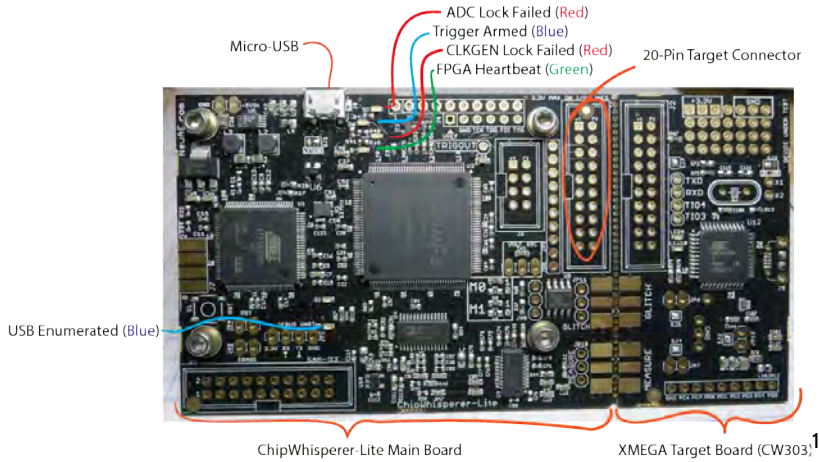


## Back off timer

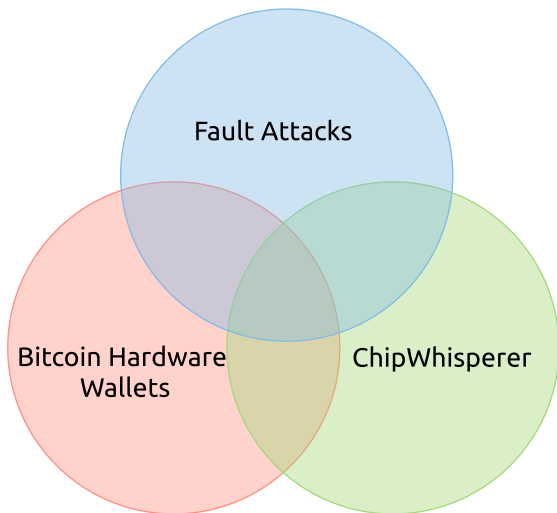
Prevents retries with a busy wait loop.



# ChipWhisperer



# This talk



# One slide intro to Fault Attacks

## Definition

An attack that applies an external stress on electronic system, which generates a security failure<sup>2</sup>.

### Two Parts:

#### 1. Fault Injection

- Vcc glitching
- Clock glitching

#### 2. Fault Exploitation

- Nicolas Bacca suggested glitching flash ops<sup>3</sup>, we wanted to bypass the PIN as it was closer to ChipWhisperer examples.

# Our Motivation

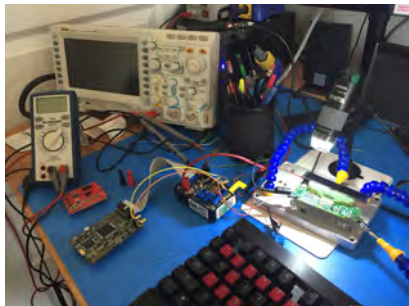
---

What happens when you apply the ChipWhisperer to the STM32F205 (F205)?

- Is the F205 vulnerable to fault injection?
- Is the TREZOR firmware exploitable via a fault?
- How do we raise awareness for these kinds of attacks?



# We just press the glitch button right?



- Turns out, you can't just shake the wallet and have BTC fall out.
- Requires some RE to determine voltages, test points, how to modify the firmware, etc...
- HW Wallets went OOS :(



## How to slow down attacks

| Exhaust the supply chain

# The Fail Train Cometh

- Clock glitching kinda worked? It made Windows USB very sad :(
- Rebooting unsigned firmware is teh suck (buttons to press).
- Timing analysis was working, but power analysis with CW was not.
- Logic level conversion is proof that the singularity is far away.
- Lots of scotch.



**F-it dude, let's go bowling.**



Or why don't we just make our own TREZOR?

## And now for something completely different

---

Before we get to the new hardware, we tried two other paths.

- De-scrambling the pin via OpenCV to automate testing.
- Decapping the STM32F205

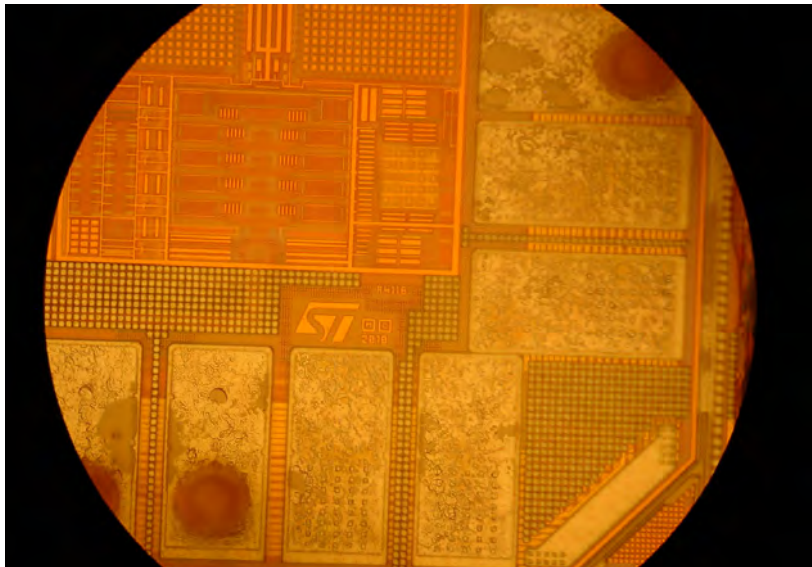
# I spy with my little eye

The screenshot displays a desktop environment with several windows:

- Figure 1 (Top Left):** A photograph of a gold-colored TREZOR hardware wallet. The screen shows the text "Please enter approval PIN" and a 3x3 grid of numbers: 1 1 8, 2 7 5, 4 6 9.
- Figure 1 (Top Middle):** A 3x3 grid of keys labeled Key 00 through Key 22. The numbers on the keys are: Row 1: 1, 3, 8; Row 2: 2, 7, 5; Row 3: 4, 6, 9.
- Figure 1 (Top Right):** A row of labels "Digit 1Digit 2Digit 3Digit 4Digit 5Digit 6Digit 7Digit 8Digit 9" followed by the digits "1 2 3 4 5 6 7 8 9".
- Figure 1 (Bottom Left):** A comparison of image and template processing for the digit '9'. It shows "Image" and "Image Area" (a yellow box around the digit) on the left, and "Template" and "Template Area" (a yellow box around the digit) on the right. Below each pair is a "Threshold" value.
- Terminal Window (Bottom Right):** A purple terminal window showing the following output:

```
Key 21 has been guessed already!  
Key 02 has been guessed already!  
Key 22 has been guessed already!  
Key 06 has been guessed already!  
Key 28 has been guessed already!  
Key 01 has been guessed already!  
Match score is too low for the key 22!  
Key 11 has been guessed already!  
Key 18 has been guessed already!  
Output:  
[[1 3 8]  
 [2 7 5]  
 [4 6 9]]  
press any key to continue...  
Guessing last digit!  
Output:  
[[1 3 8]  
 [2 7 5]  
 [4 6 9]]  
press any key to exit...  
]
```

# Decap all the things!



# We are silicon n00bs

- TBH, I just wanted to a cool silicon pic for DEF CON :)
- Decapping-as-a-Service exists though (Dangerous Prototypes)
- I asked smarter people about this:
  - Cheap images don't tell you much.
  - Some interconnects are exposed.
  - Maybe flip bits during runtime?

**? Want more pics?**

| All the decap pics are on the website.

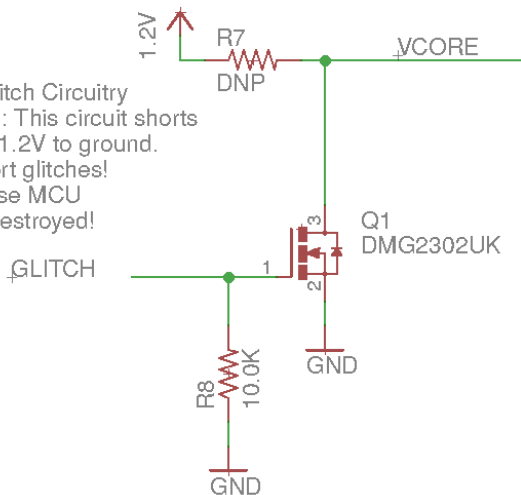
# Breaking Bitcoin Board



- Fits the ChipWhisperer UFO format
- It is *also* a TREZOR clone.
- Through-hole XTAL for more fun :)
- On board glitch hardware to attack *without* a ChipWhisperer

# Glitch on the cheap

MCU Glitch Circuitry  
Warning: This circuit shorts  
internal 1.2V to ground.  
Use short glitches!  
Otherwise MCU  
will be destroyed!

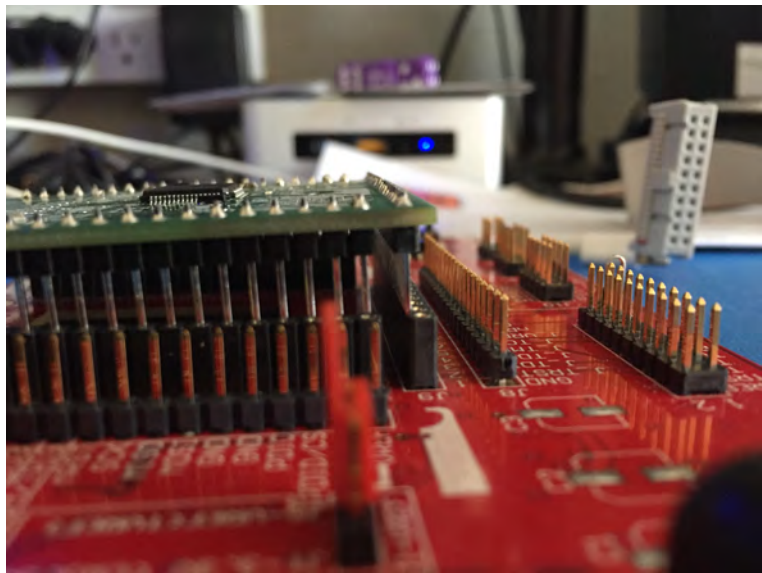




# A better setup



# There's always a Rev B



# Loop, what loop?

---

```
1 void glitch1(void)
2 {
3     //Some fake variable
4     volatile uint8_t a = 0;
5     putchar('A');
6     //Should be an infinite loop
7
8     while(a != 2){;}
9
10    uart_puts("1234");
11    while(1){;}
12 }
```

---

# Loop, what loop?

The screenshot displays a software interface with several components:

- Scope Settings:** A table with parameters and values.
- Trace Output Plot:** A graph titled "Power Trace View" showing a noisy red signal around 0.
- Debug Logging:** A list of log messages, including warnings and info.
- Terminal Window (CWCapture.pyw):** A window showing a list of hex addresses and data, with the text "A/ohello" appearing multiple times.

Parameter	Value
<b>Glitch Module</b>	
Clock Source	CLKGEN
Glitch Width (as % of period)	10
Glitch Width (fine adjust)	0
Glitch Offset (as % of period)	10
Glitch Offset (fine adjust)	0
Glitch Trigger	Manual
Single-Shot Arm	After Scope Arm
Exit Trigger Offset	0
Repeat	21
Manual Trigger / Single-Shot Arm	
Output Mode	Enable Only
Read Status	
Reset DCM	

**Trace Output Plot:** Power Trace View. The y-axis is labeled "Data" and ranges from -0.4 to 0.4. The x-axis is labeled "0". The plot shows a noisy red signal fluctuating around 0.

**Debug Logging:**

- WARNING - Timeout in OpenADC capture(), trigger FORCED
- WARNING - Timeout in OpenADC capture(), trigger FORCED
- WARNING - Timeout in OpenADC capture(), trigger FORCED
- WARNING - Timeout in OpenADC capture(), trigger FORCED
- WARNING - Timeout in OpenADC capture(), trigger FORCED
- WARNING - Timeout in OpenADC capture(), trigger FORCED
- WARNING - Timeout in OpenADC capture(), trigger FORCED
- WARNING - Timeout in OpenADC capture(), trigger FORCED
- INFO - Capture completed.
- WARNING - SAM3U Serial buffers OVERRUN - data loss has occurred.
- INFO - Capture completed.
- INFO - Capture completed.
- INFO - Capture completed.
- INFO - Capture completed.
- WARNING - Timeout in OpenADC capture(), trigger FORCED
- WARNING - Timeout in OpenADC capture(), trigger FORCED

**CWCapture.pyw Terminal:**

```
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1800000018 200 200 536936476
1500A00000000hello
A/ohello
A/ohello
A/ohello
A12345
```

# Ooof, that hurts

---

```
1 void glitch_infinite(void)
2 {
3     char str[64]; unsigned int k = 0;
4     //This also adds lots of SRAM access
5     volatile uint16_t i, j;
6     volatile uint32_t cnt;
7     while(1){
8         cnt = 0; trigger_high(); trigger_low();
9         for(i=0; i<200; i++){
10             for(j=0; j<200; j++){cnt++;}}
11         sprintf(str, "%lu %d %d %d\n",
12                 cnt, i, j, k++);
13         uart_puts(str);}}
```

---





# O Password, My Password

---

```
1 void glitch3(void)
2 {
3     char passwd[] = "touch"; char passok = 1;
4     for(cnt = 0; cnt < 5; cnt++){
5         if (inp[cnt] != passwd[cnt]){
6             passok = 0;}}
7     if (!passok){
8         uart_puts("Denied\n"); while (1);
9     } else {
10        uart_puts("Welcome\n");
11    }
12
13    led_error(1); led_error(1); led_error(1);
14 }
```



# O Password, My Password

The screenshot displays a software interface for a glitch attack. The main window is titled "ChipWhisperer™ Capture V3.3.4 - 06\_glitch.cwb".

**Scope Settings:**

Parameter	Value
Target ID2: GPIO	Disabled
Target ID3: GPIO	Disabled
Target ID4: GPIO	Disabled
NST: GPIO	High
PDID: GPIO	Default
PDIC: GPIO	Default
Target Power State	<input checked="" type="checkbox"/>
<b>Glitch Module</b>	
Clock Source	CLKGEN
Glitch Width (as % of period)	10
Glitch Width (fine adjust)	0
Glitch Offset (as % of period)	10
Glitch Offset (fine adjust)	0
Glitch Trigger	Ext. Trigger/Single-Shot
Single-Shot Arm	After Scope Arm
Ext. Trigger Offset	20
Repeat	20

**Trace Output Plot:** A waveform plot showing a glitch pulse. The y-axis is labeled "Data" and ranges from -0.4 to 0.4. The x-axis is labeled "Time" and ranges from 0 to 20. The plot shows a red trace that starts at 0, drops to approximately -0.3, then rises sharply to 0.4, and then settles around 0.2.

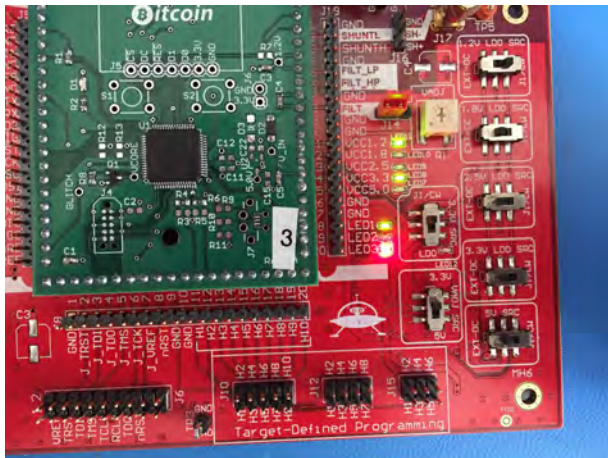
**Power Trace View:** A window titled "CWCapture.pyw" showing a terminal output of a password field being attacked. The output is as follows:

```
000000hello
Password:touj
02000000hello
Password:touj
Gonna need you to stick that password waaaaaaay up in your butt, Morty.
000000hello
Password:touj
#000001000000hello
Password:touj
0002000000hello
Password:touj
Gonna need you to stick that password waaaaaaay up in your butt, Morty.
04A0000000hello
Password:touj
+00000hello
Password:touj
Gonna need you to stick that password waaaaaaay up in your butt, Morty.
@00000hello
Password:touj
```

The terminal window also includes a "Send" button and a "Clear" button. At the bottom, it shows "TX on Send: \n" and "RX/TX: Display Mode: ASCII with hex for invalid ASCII".

**Debug Logging:** A window titled "Debug Logging" showing a list of "INFO - Capture completed." messages.

# O Password, My Password



# Ok, how'd we do

- Is the F205 vulnerable to fault injection?
  - Absolutely, yes.
- Is the TREZOR firmware exploitable via a fault?
  - Maybe? We have thoughts on how to trigger but going from example to exploit takes some work still.
  - We talked to TREZOR and KeepKey about some issues.
- How do we raise awareness for these kinds of attacks?
  - While not quite an *unlooper device*, our PCB will help you find the BORE (Break Once Run Everywhere) attack.

# Summary of Vulnerabilities

- STM32F205 is susceptible to fault attacks.
- KeepKey had a timing analysis bug on PIN verification.
- TREZOR (and all clones) did not enable Clock Security System in the MCU, allowing injection of clock faults.
- A few pieces of code that could be made to more resilient.



## Takeaway for wallet users

Don't lose physical control of your wallet.  
You really want to set PIN plus password.



## Takeaway for wallet designers

You will be glitched—can you trust your clock and VCC?

# Defenses from Fault Attacks

Write code assuming you will be glitched! (Riscure RSA 2008)<sup>4</sup>  
and The Sorcerer's Apprentice Guide to Fault Attacks.

- Don't use 0 and not 0, using Hamming distance.
- Count your functions!
- Check for complete loop completion.
- Add Random delay—makes triggering a bit harder.
- Check sensitive operations multiple times and compare results.
- Use multiple MCUs and check results?!



## Chipwhisperer vs. STM32F205

| Let's see some glitches!!!



SHOW ME WHAT U GOT

# Endnotes

<sup>1</sup>[https://wiki.newae.com/File:Cwlite\\_basic.png](https://wiki.newae.com/File:Cwlite_basic.png)

<sup>2</sup>*Encyclopedia of Cryptography and Security, 2nd Edition.*

<sup>3</sup><https://www.slideshare.net/EricLarcheveque/bitcoin-hardware-wallets-security>

<sup>4</sup>[https://cryptotronix.files.wordpress.com/2017/07/paper\\_side\\_channel\\_patterns.pdf](https://cryptotronix.files.wordpress.com/2017/07/paper_side_channel_patterns.pdf)