Glitch in the Matrix: Exploiting Bitcoin Hardware Wallets

Sergei Volokitin

What is a Hardware Wallet?

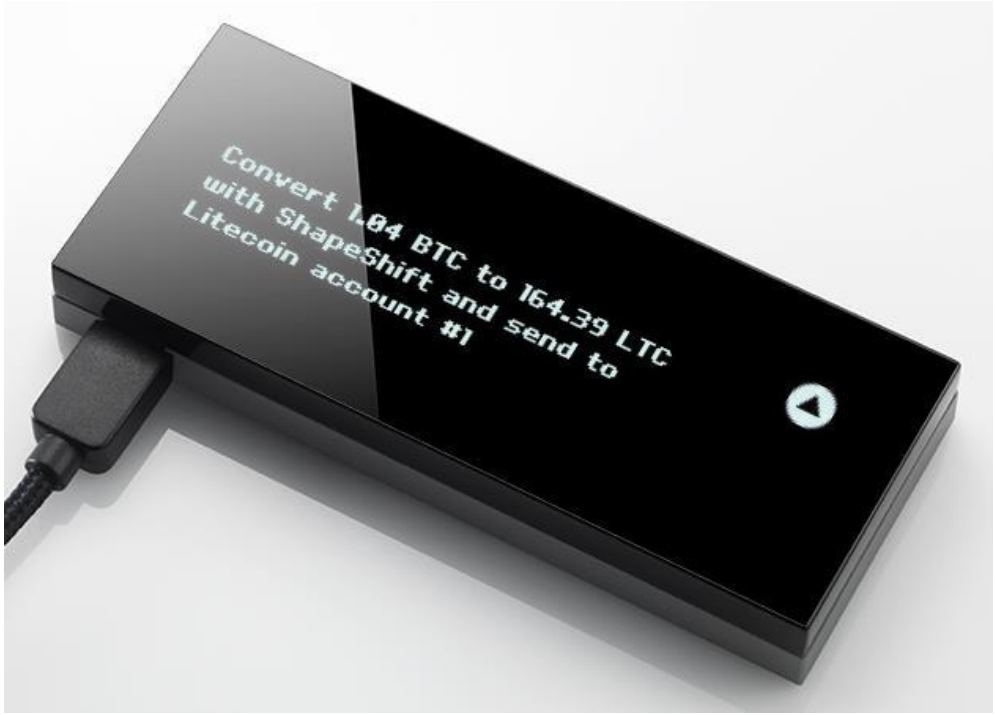
- Connects to smartphone, PC
- Stores and operates with private keys
- Mainly used for cryptocurrency keys

https://www.ledgerwallet.com/products/ledger-nano-s

https://www.keepkey.com/wp-content/uploads/2014/08/12121301/shapeshift-large.jpg

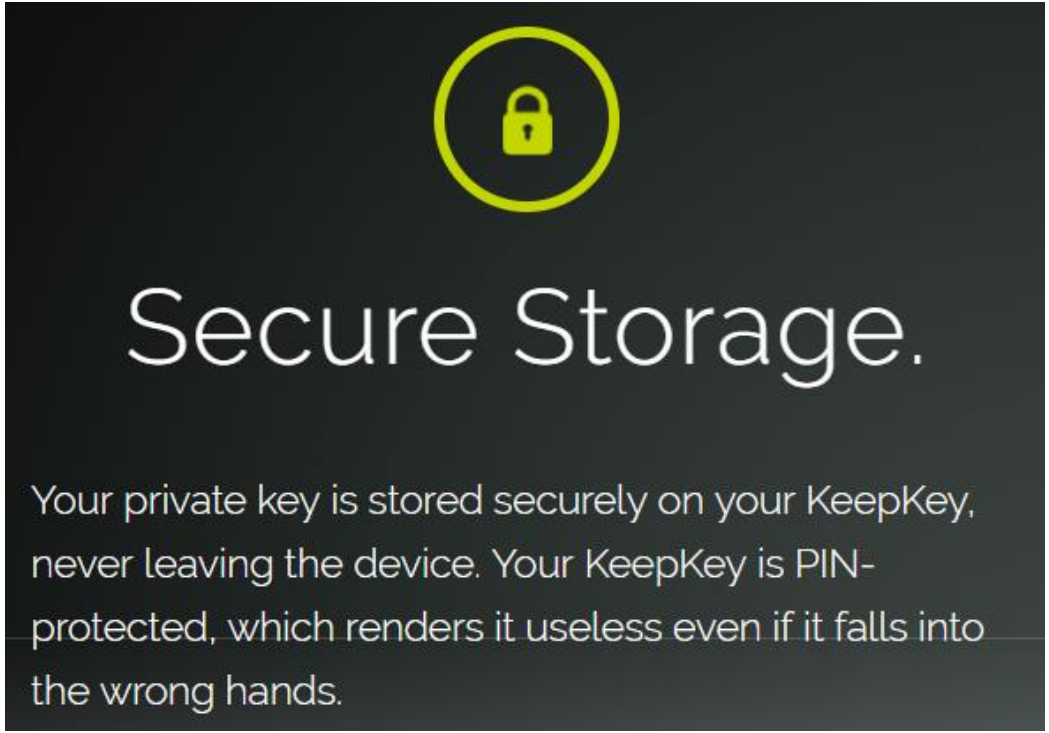
https://trezor.io/start/

KeepKey



https://www.keepkey.com/wp-content/uploads/2014/08/12121301/shapeshift-large.jpg

Why KeepKey?



https://www.keepkey.com/

KeepKey

KeepKey

KeepKey - The Simple Cryptocurrency Hardware Wallet

376 customer reviews

| 156 answered questions

Available from these sellers.

Color: **Black and Anodized Aluminum**

$129.00

used from $120.00

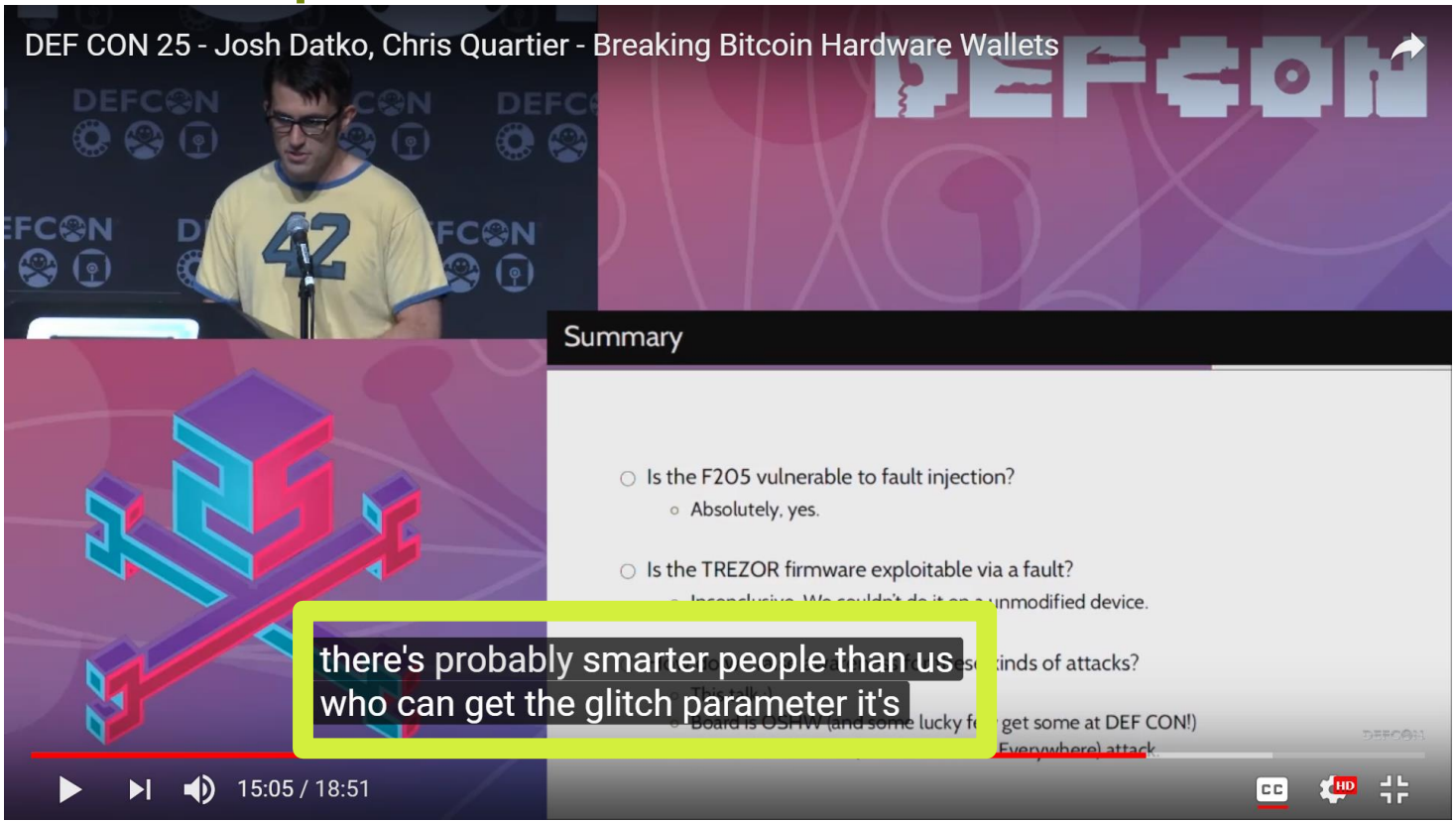
- Bank-Grade Security that is Simple: KeepKey is the most secure bitcoin wallet available. It makes best-practice bitcoin security easy so that even your grandmother can protect her bitcoin wealth.
- Backup and Recovery: During initialization, you are given the

https://www.amazon.com/KeepKey-Simple-Cryptocurrency-Hardware-Wallet/dp/B0143M2A5S

DEFCON presentation on TREZOR clone

- Is the TREZOR firmware exploitable via a fault?
  - Inconclusive. We couldn't do it on a unmodified device.

DEFCON presentation on TREZOR clone



Features

- **STM32**
- Flash on the chip
- **Large attack surface** (22 input commands without auth)
- Built-in 4 digit PIN security lock
- **Open Source (bootloader and firmware)**
- Built-in onboarding (seed generation and recovery)
- USB connectivity
- Super secure boot with **three signatures and five keys!**

Using HW wallet

Hardware architecture

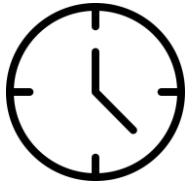
- STM32F205
- Internal 1MB of flash
- There is secure boot

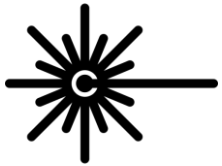
```
static const FlashSector flash_sector_map[] =
{
    { 0,  0x08000000, BSTRP_FLASH_SECT_LEN, FLASH_BOOTSTRAP },
    { 1,  0x08004000, STOR_FLASH_SECT_LEN,  FLASH_STORAGE1  },
    { 2,  0x08008000, STOR_FLASH_SECT_LEN,  FLASH_STORAGE2  },
    { 3,  0x0800C000, STOR_FLASH_SECT_LEN,  FLASH_STORAGE3  },
    { 4,  0x08010000, UNUSED_FLASH_SECT0_LEN, FLASH_UNUSED0 },
    { 5,  0x08020000, BLDR_FLASH_SECT_LEN,  FLASH_BOOTLOADER },
    { 6,  0x08040000, BLDR_FLASH_SECT_LEN,  FLASH_BOOTLOADER },
    { 7,  0x08060000, APP_FLASH_SECT_LEN,   FLASH_APP },
    { 8,  0x08080000, APP_FLASH_SECT_LEN,   FLASH_APP },
    { 9,  0x080A0000, APP_FLASH_SECT_LEN,   FLASH_APP },
    { 10, 0x080C0000, APP_FLASH_SECT_LEN,   FLASH_APP },
    { 11, 0x080E0000, APP_FLASH_SECT_LEN,   FLASH_APP },
    { -1, 0,          0,        FLASH_INVALID}
};
```

Why hardware attack

- Popular open source project
- SW is tested and patched over time
- General purpose MCU is used to keep the secrets

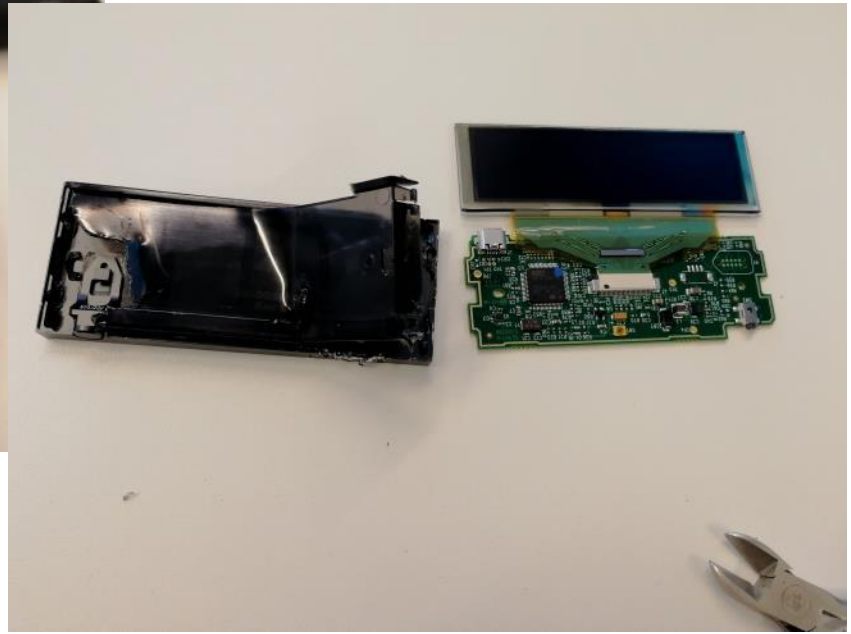
What is FI and how can it help?



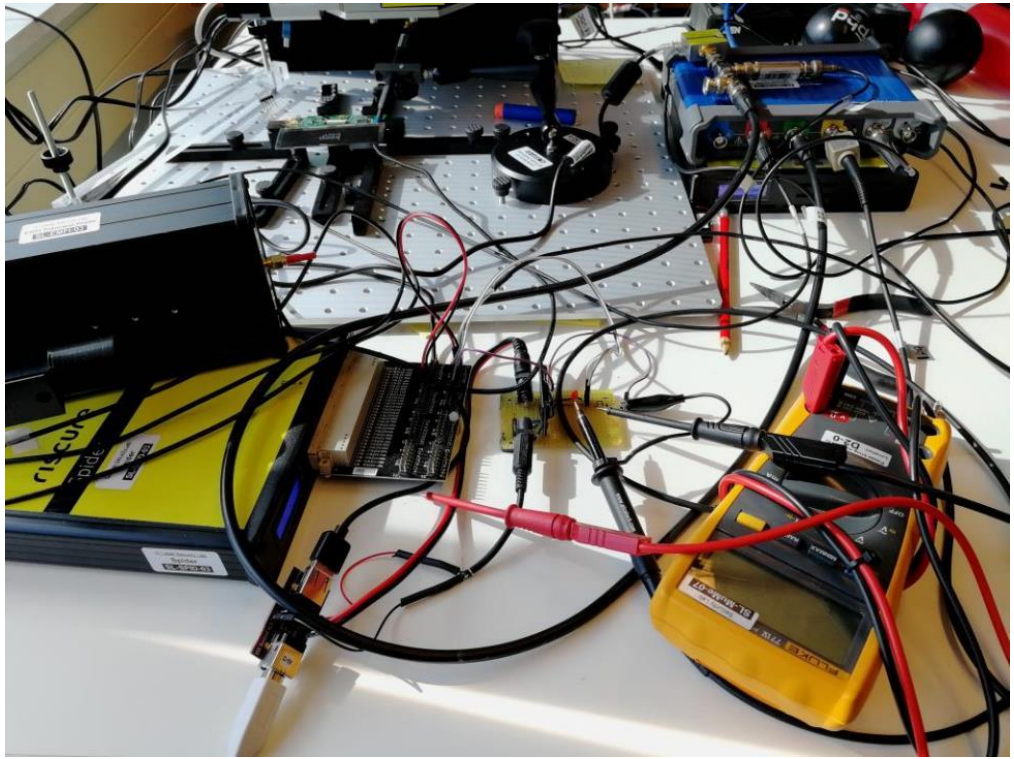




- Corrupt data (0x00, 0xFF, 0x??)
- Corrupt instructions
- Skip instructions
- …

Cracking the case



Can we glitch it?



Characterization

- Simple command to be sent to the device
- Ping command receives a message and sends it back
- Test if we can successfully glitch the hardware

```
void fsm_msgPing(Ping *msg)
{
    …
   if(msg->has_message)
    {
        resp->has_message = true;
        memcpy(&(resp->message), &(msg->message), sizeof(resp->message));
    }
    msg_write(MessageType_MessageType_Success, resp);
    go_home();
}
```

Characterization

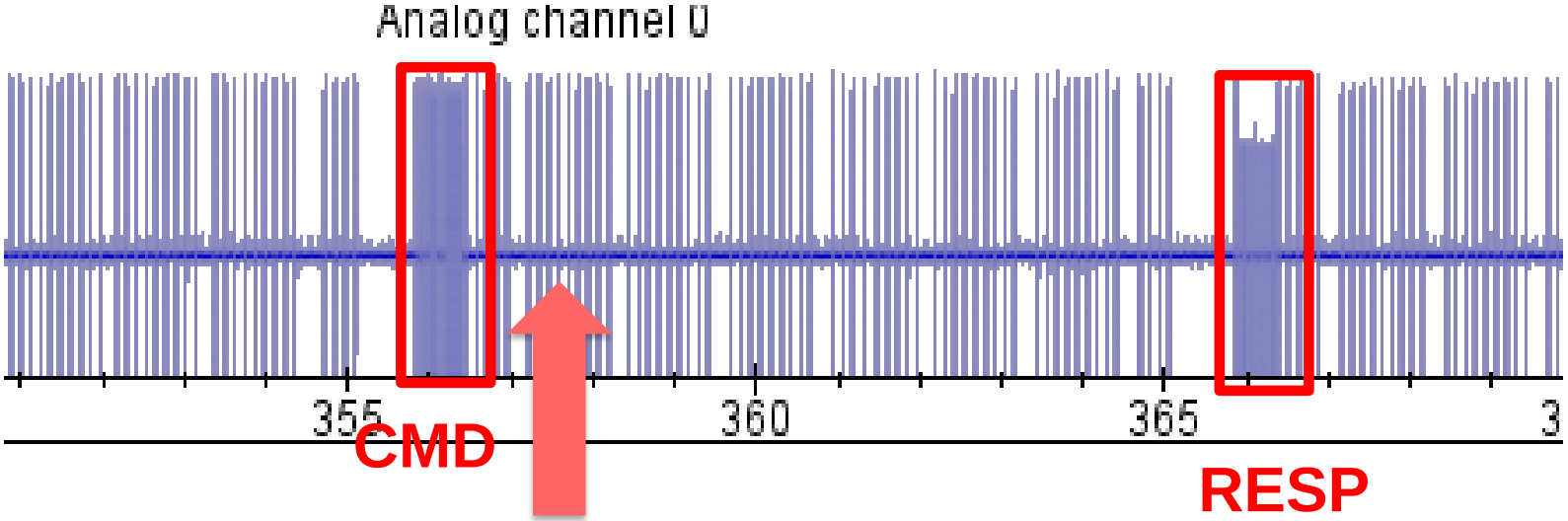
- Simple command to be sent to the device
- Ping command receives a message and sends it back
- Test if we can successfully glitch the hardware

```
void fsm_msgPing(Ping *msg)
{
    …
   if(msg->has_message)
    {
        resp->has_message = true;
        memcpy(&(resp->message), &(msg->message), sizeof(resp->message));
    }
    msg_write(MessageType_MessageType_Success, resp);
    go_home();
}
```

GLITCH!

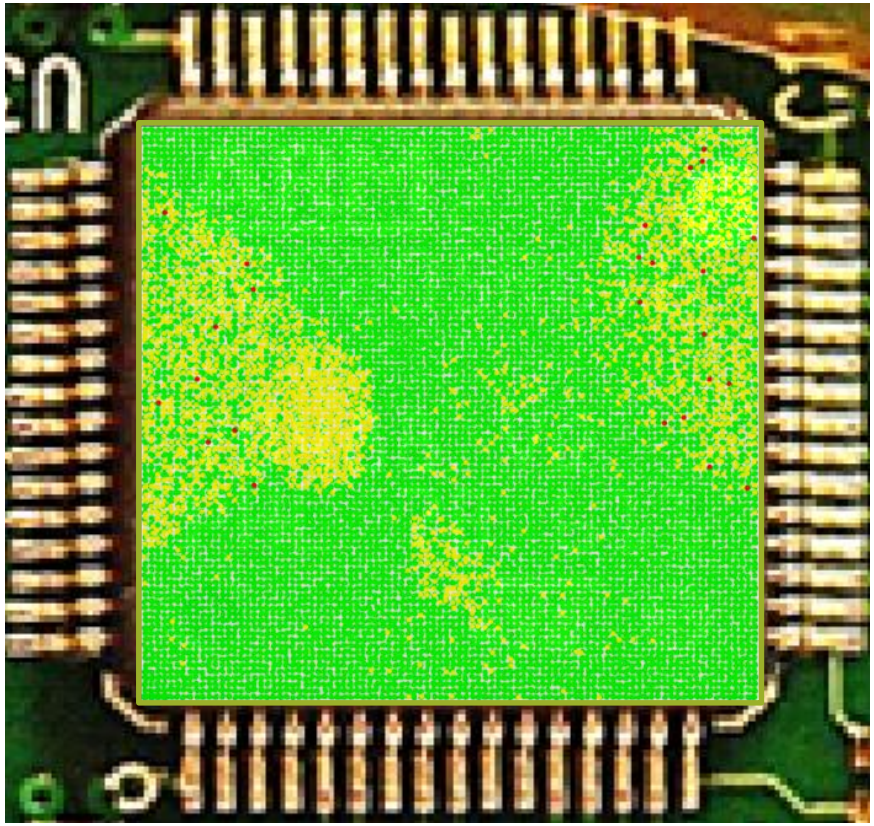
No code execution, no easy trigger

- The power comes from USB and quite noisy
- No modifications to the device were made
- When a command is sent a similar pattern is observed

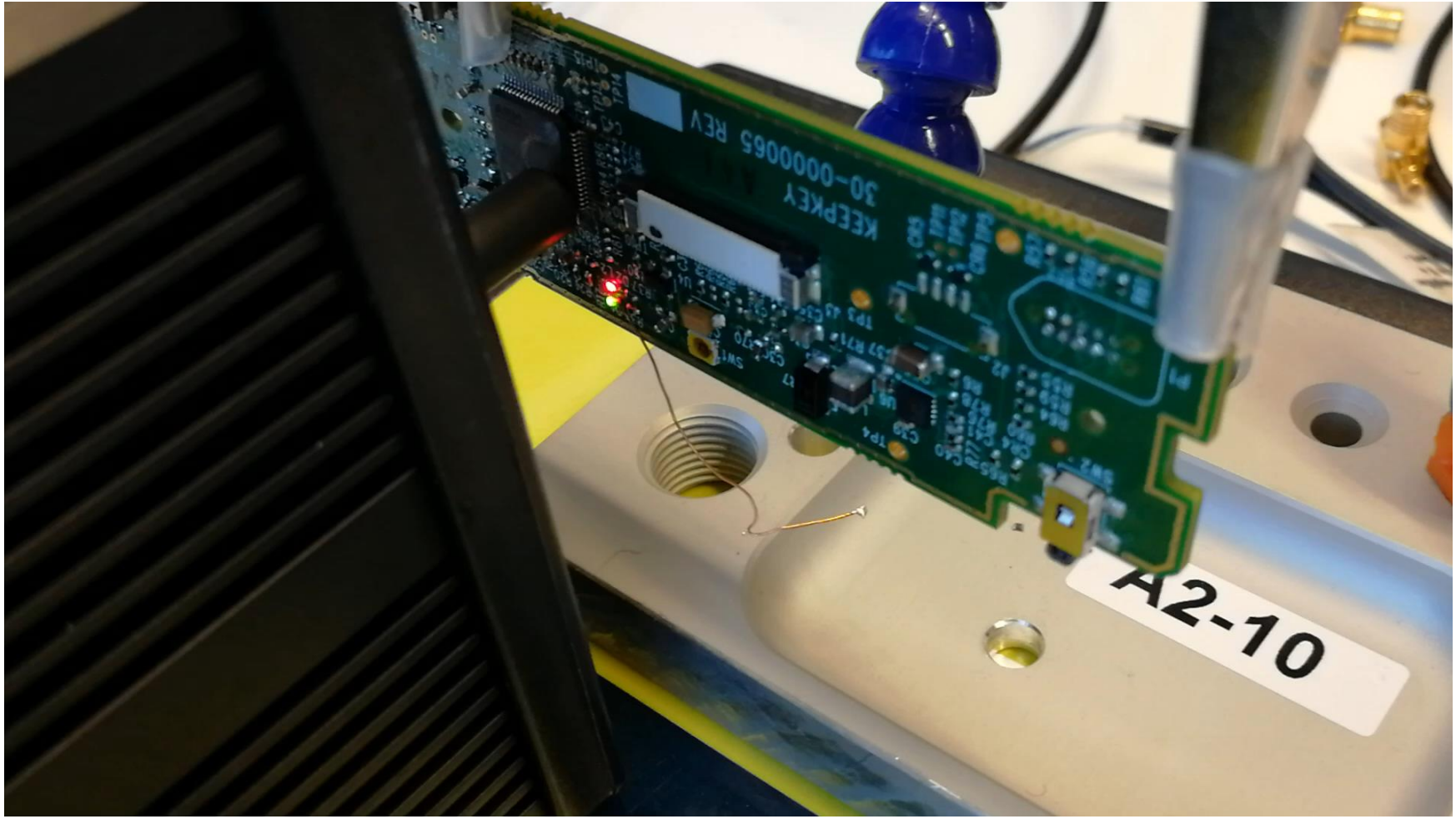


GLITCH!

Characterization results



DEMO


A2-10

Characterization results

Ping command response
HelloWorld
\01
H
H\00\00\00\00World
Hworld\10
Hel
Hel\08oWorld
Hell\00World
HelloW
HelloWo
Hell

Ping command response
HelloWorl\00
HelloWorlW
lelloWorld
\00elloWorld
He
He\00\00\00\00orld
##
HelloWorlD
@elloWorld
HElloWorld
Settings applied

Glitching the screen output

Glitching the screen output

Glitching the screen output


```
/// Non-maskable interrupt handler
void nmi_handler(void) {
   // Look for the clock instability interrupt. This is a security measure
   // that helps prevent clock glitching.

   if ((RCC_CIR & RCC_CIR_CSSF) != 0) {
      layout_warning_static("Clock instability detected. Reboot Device!");
      system_halt();
   } }
```

More glitches

Is there an exploitable glitch?

Getting full access to the device

```
void fsm_msgChangePin(ChangePin *msg)
{
    bool removal = msg->has_remove && msg->remove;
    bool confirmed = false;

    if(removal)
    {
        if(storage_has_pin())
        {
            confirmed = confirm(ButtonRequestType_ButtonRequest_Rem
                                "Remove PIN", "Do you want to remov
        }
        else
    ...

    if(!pin_protect("Enter Current PIN"))
    {
        go_home();
        return;
    }
```

GLITCH!

Getting full access to the device

```
void fsm_msgChangePin(ChangePin *msg)
{
    bool removal = msg->has_remove && msg->remove;
    bool confirmed = false;

    if(removal)
    {
        if(storage_has_pin())
        {
            confirmed = confirm(ButtonRequestType_ButtonRequest_Rem
                                "Remove PIN", "Do you want to remov
        }
        else
    ...

    if(!pin_protect("Enter Current PIN"))
    {
        go_home();
        return;
    }
```

FAIL!

Getting full access to the device #2

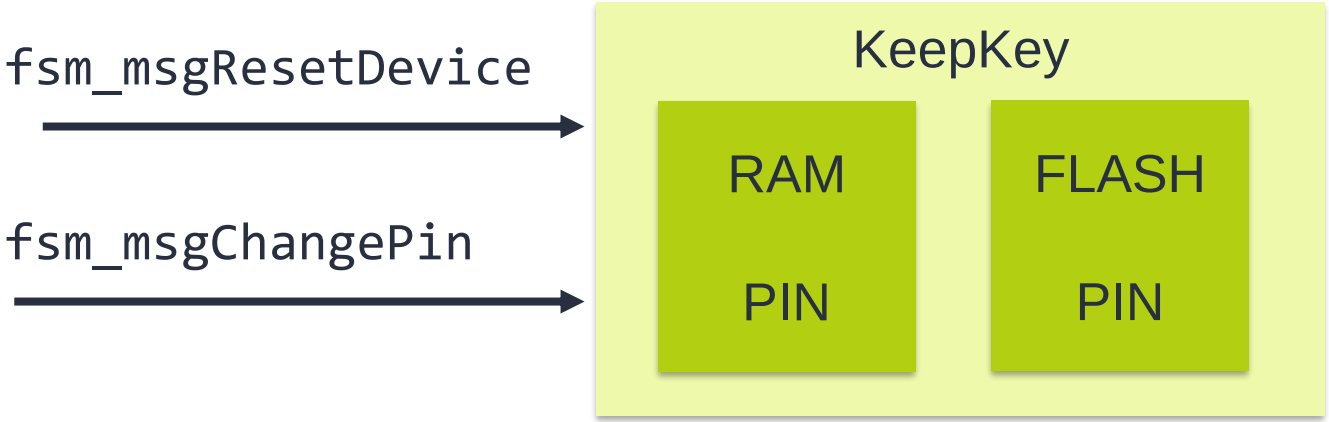
```
void fsm_msgResetDevice(ResetDevice *msg)
{
    if(storage_is_initialized())
    {
        fsm_sendFailure(FailureType_Failure_UnexpectedMessage,
                        "Device is already initialized. Use Wipe first.");
        return;
    }

    reset_init(
        msg->has_display_random && msg->display_random,
        msg->has_strength ? msg->strength : 128,
        msg->has_passphrase_protection && msg->passphrase_protection,
        msg->has_pin_protection && msg->pin_protection,
        msg->has_language ? msg->language : 0,
        msg->has_label ? msg->label : 0
    );
}
```

GLITCH!

SW Design leading to exploitable FI

- The glitch of the if-statement is possible but does not change the flash



- `fsm_msgResetDevice` command once glitched only changes PIN in RAM
- `fsm_msgChangePin` compares against PIN in RAM and saves a new one to FLASH

Getting full access to the device #2

The attack:

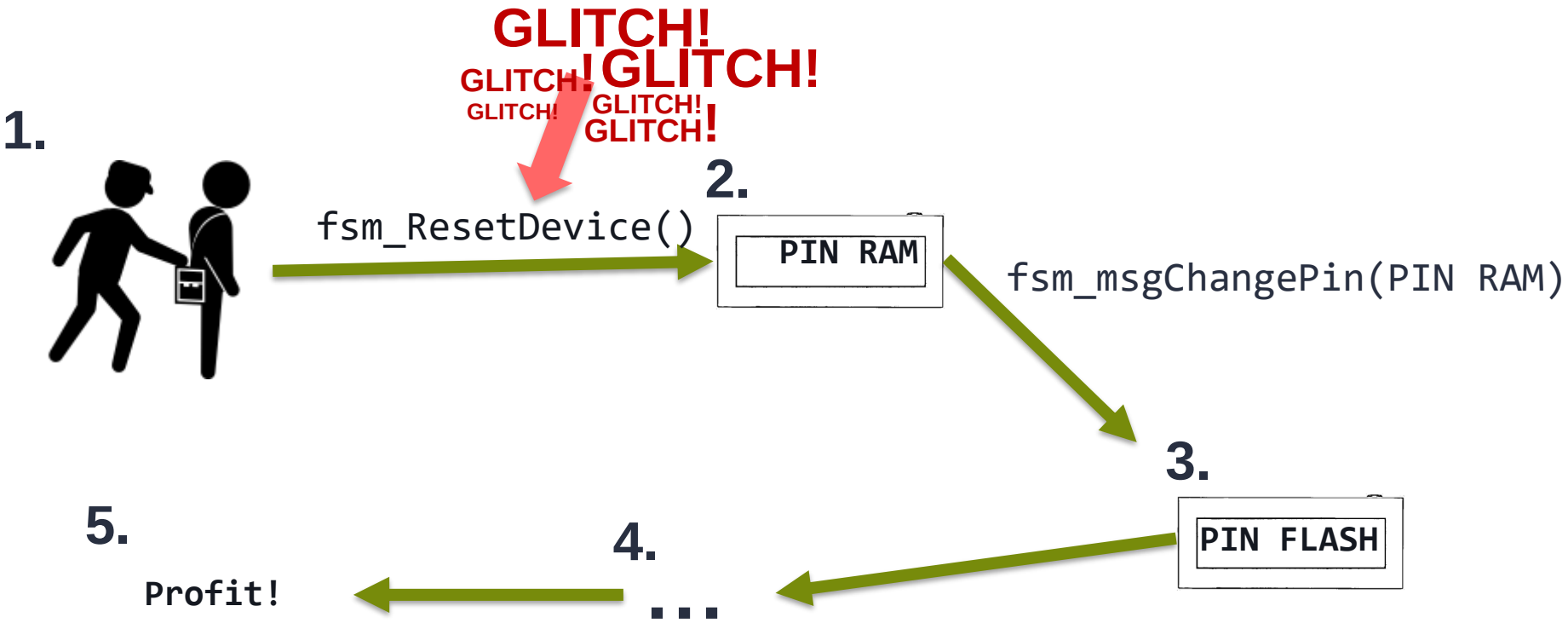
1. ~~Steal~~ Find a device
2. Glitch the check of the lifecycle check
3. Set a new PIN on the device, keep the seed
4. Unlock the device using the new pin

...

5. Get full access to the device's coins

GLITCH!

Getting full access to the device #2



Results

Success rate **~1.2%**

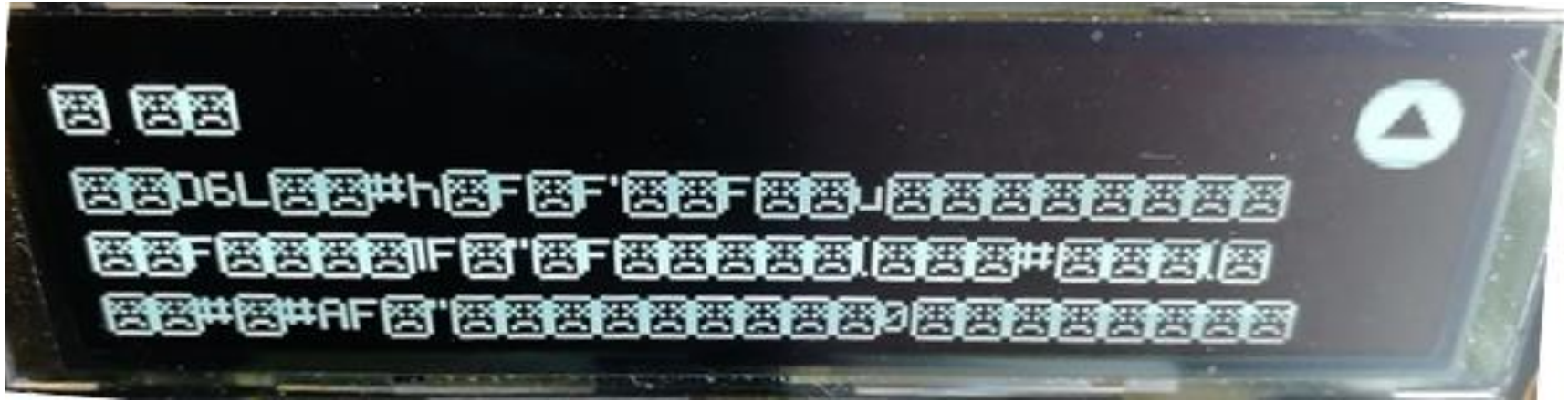
Attempt rate **0.3** att/sec

On average it takes **5 minutes** to glitch the PIN

Conclusions

- Non secure hardware is easily glitchable
- Simple FI counter measures are not sufficient against EMFI
- Large set of commands avaliable to anauthorized user are difficult to secure

Questions?



Sergei Volokitin
Security Analyst
sergei@riscure.com

riscure

Challenge your security

Riscure B.V.
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90

www.riscure.com

Riscure North America
550 Kearny St., Suite 330
San Francisco, CA 94108 USA
Phone: +1 650 646 99 79

inforequest@riscure.com

Riscure China
Room 2030-31, No. 989, Changle Road, Shanghai 200031
China
Phone: +86 21 5117 5435

inforcn@riscure.com

riscure

Challenge your security