Defeating a Secure Element with Multiple Laser Fault Injections

Olivier Hériveaux

Plan

Introduction

Microchip ATECC circuit family
Context: Coldcard Hardware Wallet
Sample preparation
First observations

EEPROM fault model characterization

Wake Up mechanism evaluation
Construction of a powerful attack primitive

Testing the *Read* command

Multiple testing campaigns, with 1, 2 and finally 4 faults
Progressive chip internals discovery
Final working attack with 2 faults

Study in complete black box approach!

Previous, current and future work

Evaluated in 2019

Evaluated in 2020

Stay tuned...

Coldcard Wallet

Bitcoin Hardware Wallet

Evaluation of the new Mk3 version

STM32L496 Microcontroller

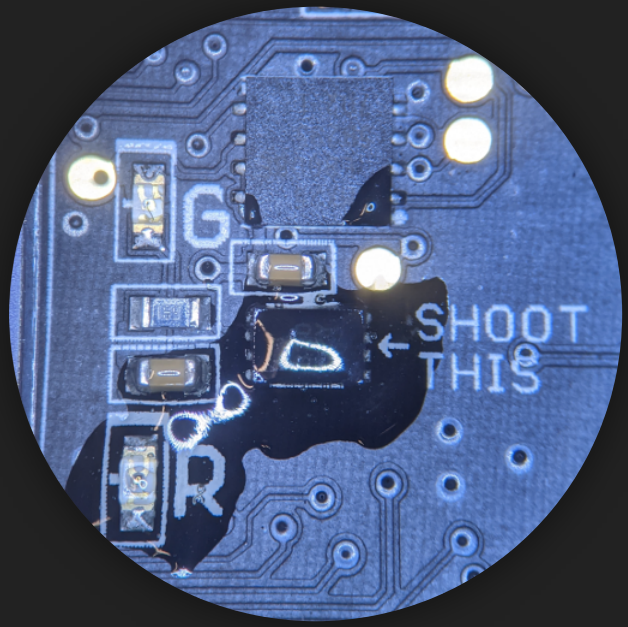
Main firmware
Encrypts/decrypts the Seed

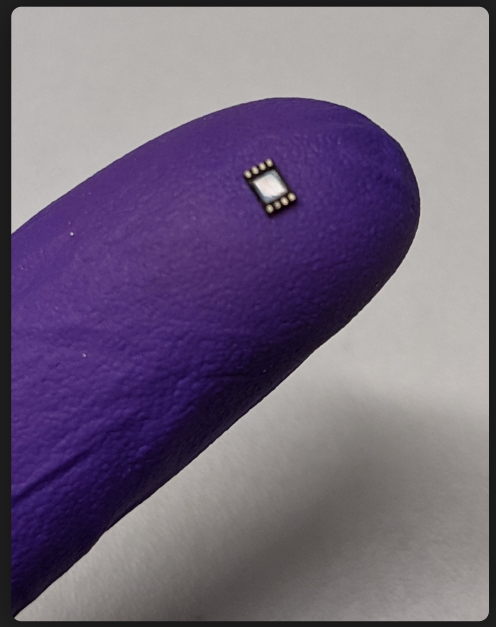
⇅

ATECC608A

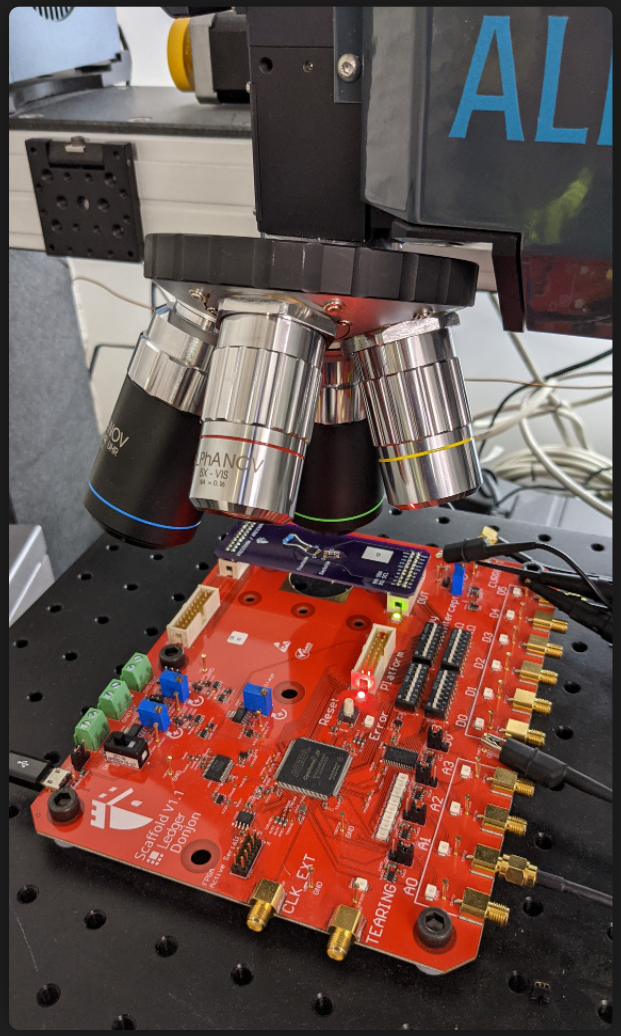
Securely stores the Seed (private key)
Unlock through authentification (PIN code)

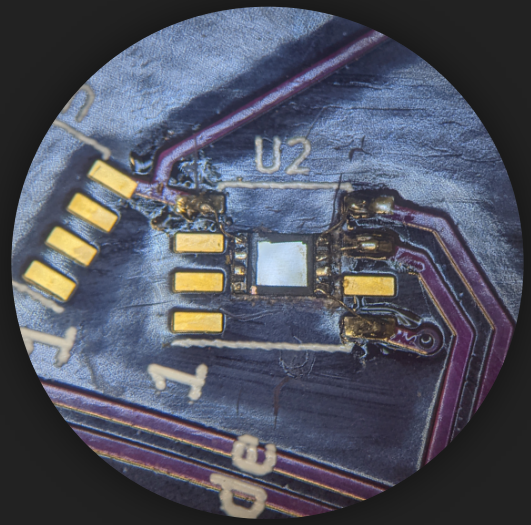
Desoldering and depackaging

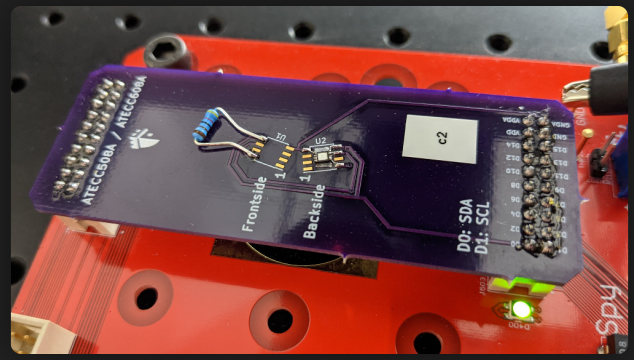




Sample setup for Laser Fault Injection







New study

Optical imaging of the die

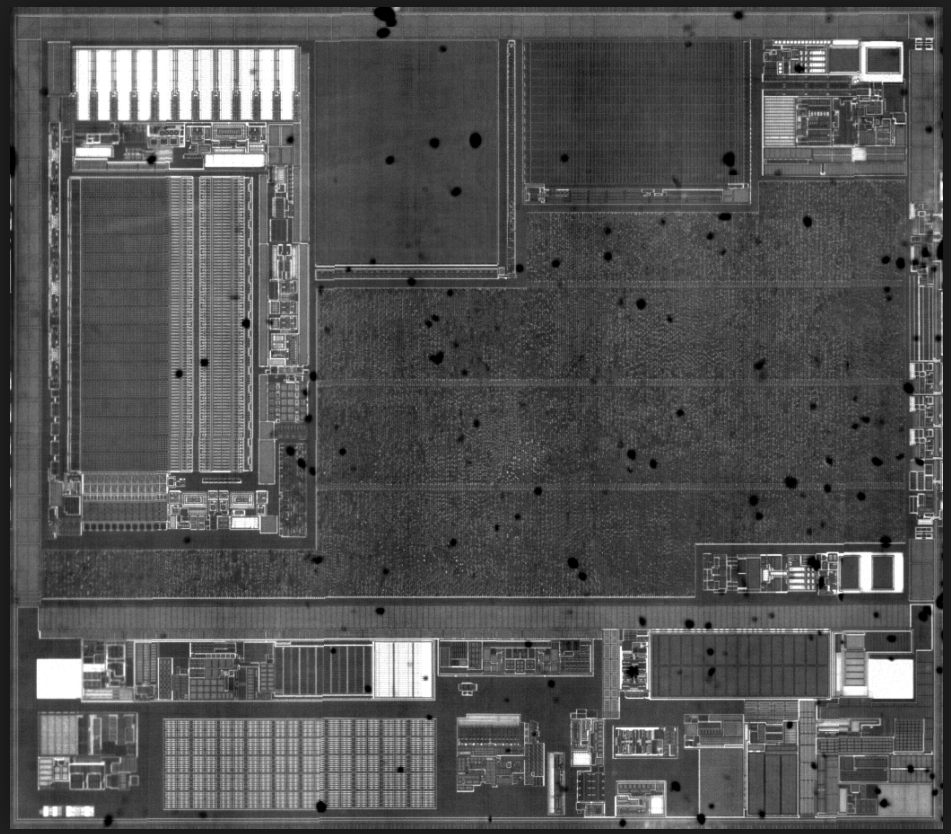
Infrared camera mounted on microscope
Circuit looks exactly the same as previous generation!

Same circuit, locked functionnalities?

Multiple commercial grades
Larger market coverage
No additional cost
Industry common practice

Same circuit, new firmware?

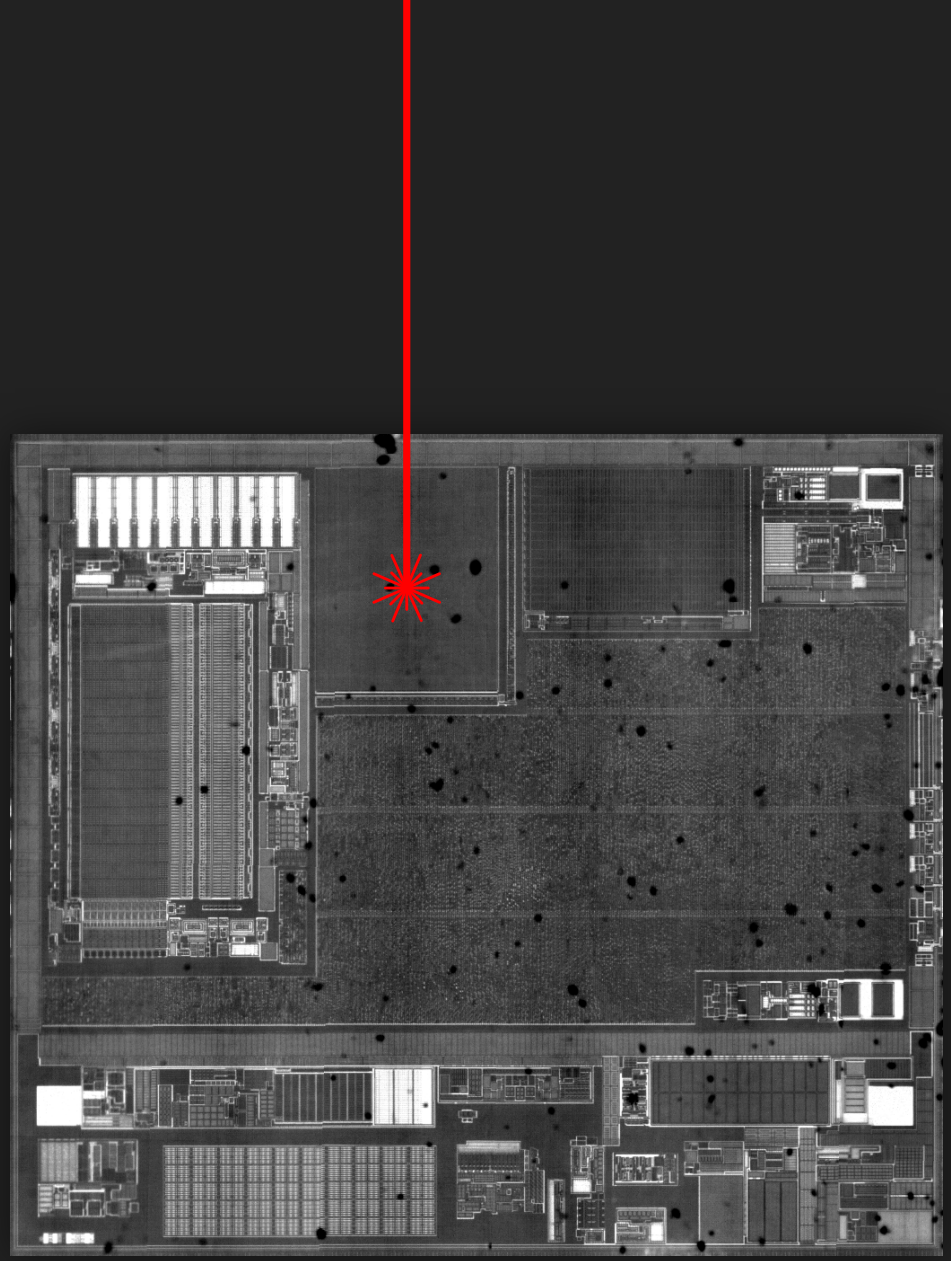
Firmware stored in ROM
Only one mask to modify, limited cost
Cannot patch hardware issues
Update limited to ROM size



New study

ATECC508A attack trial

ROM Fault Injection
CPU instruction modification
Did not work
No secret extracted with this attack path



New study

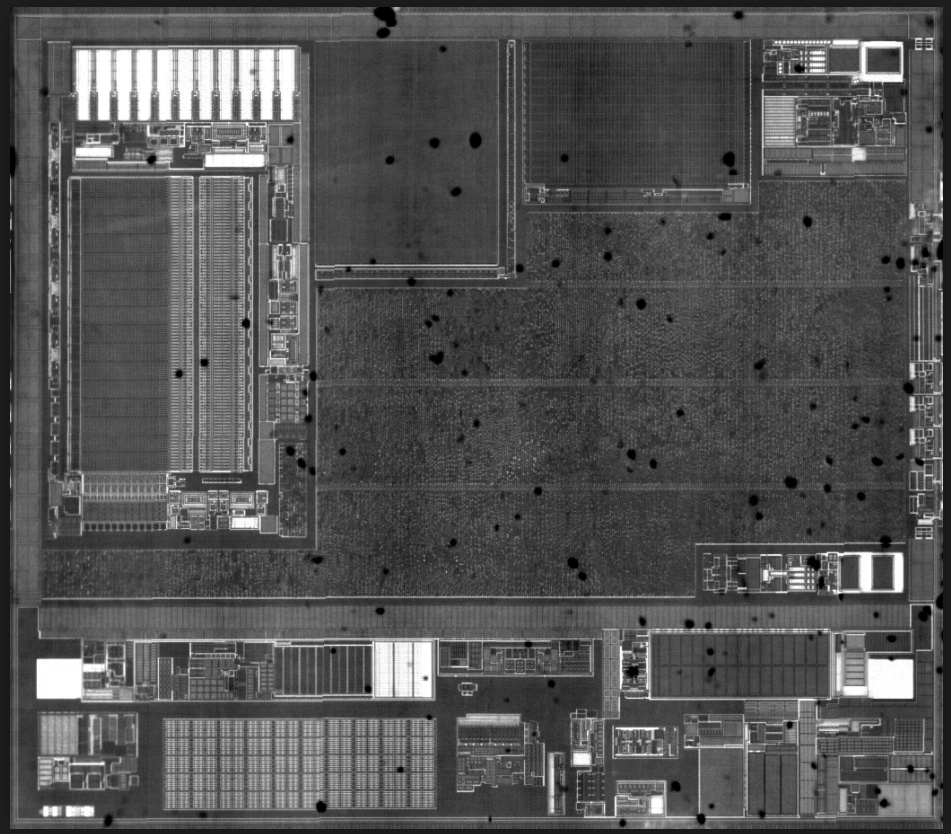
ATECC508A attack trial

ROM Fault Injection
CPU instruction modification
Did not work
No secret extracted with this attack path

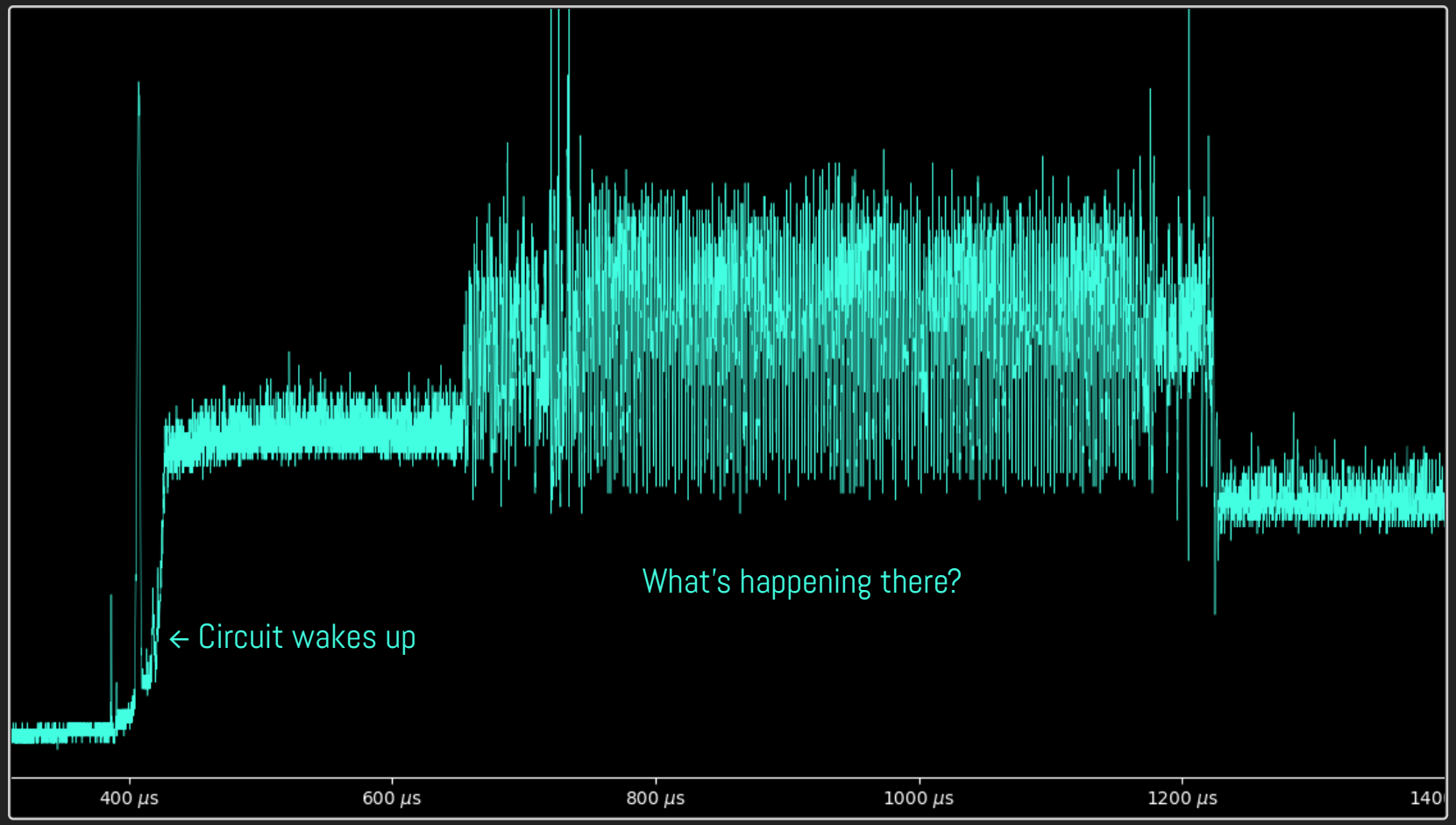
Search for a new attack path...

Wake Up mechanism investigation

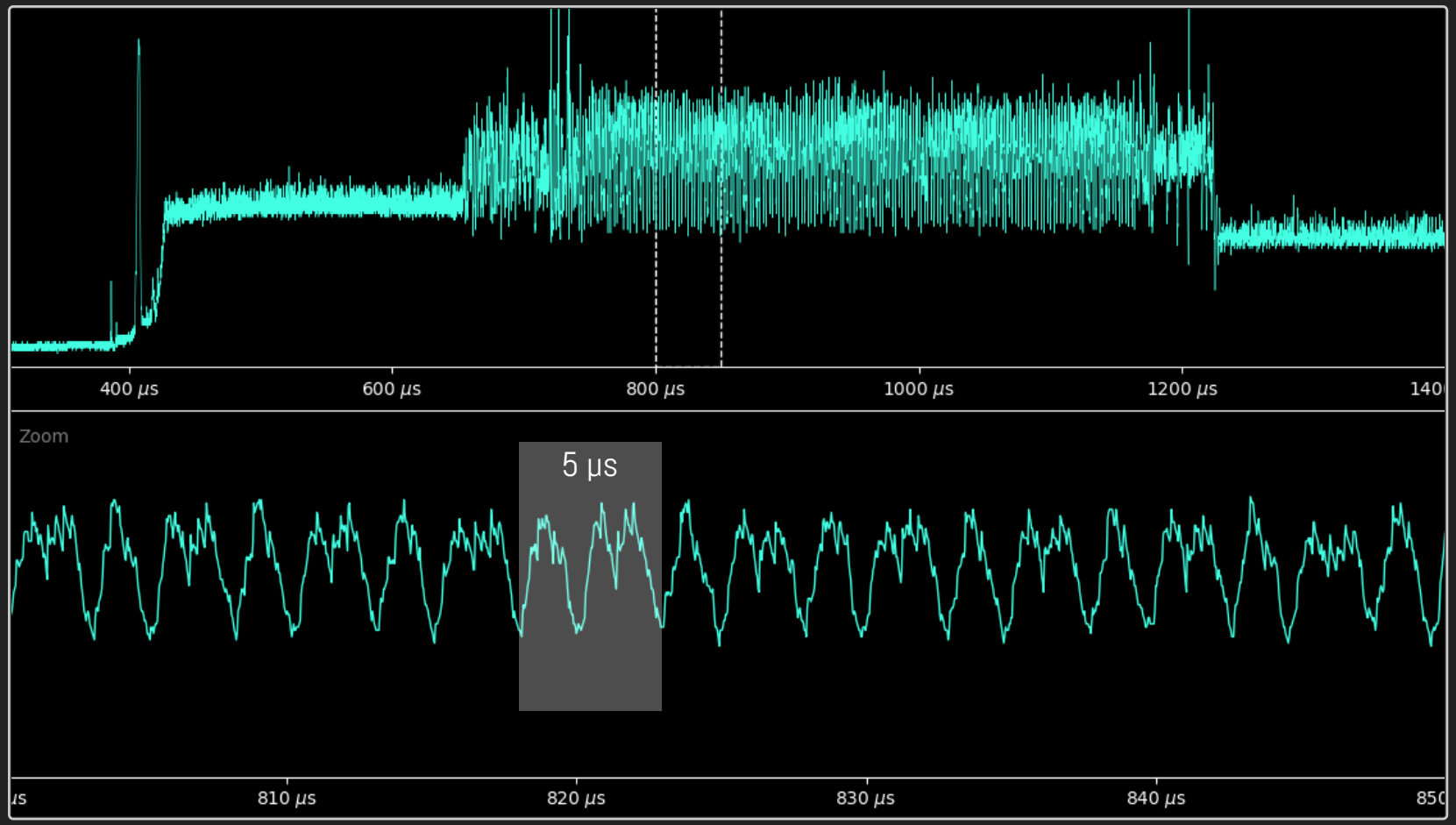
Low power functionality, exits the sleep mode
Suspicious CPU activity observed
Guess: access rights caching for later use?



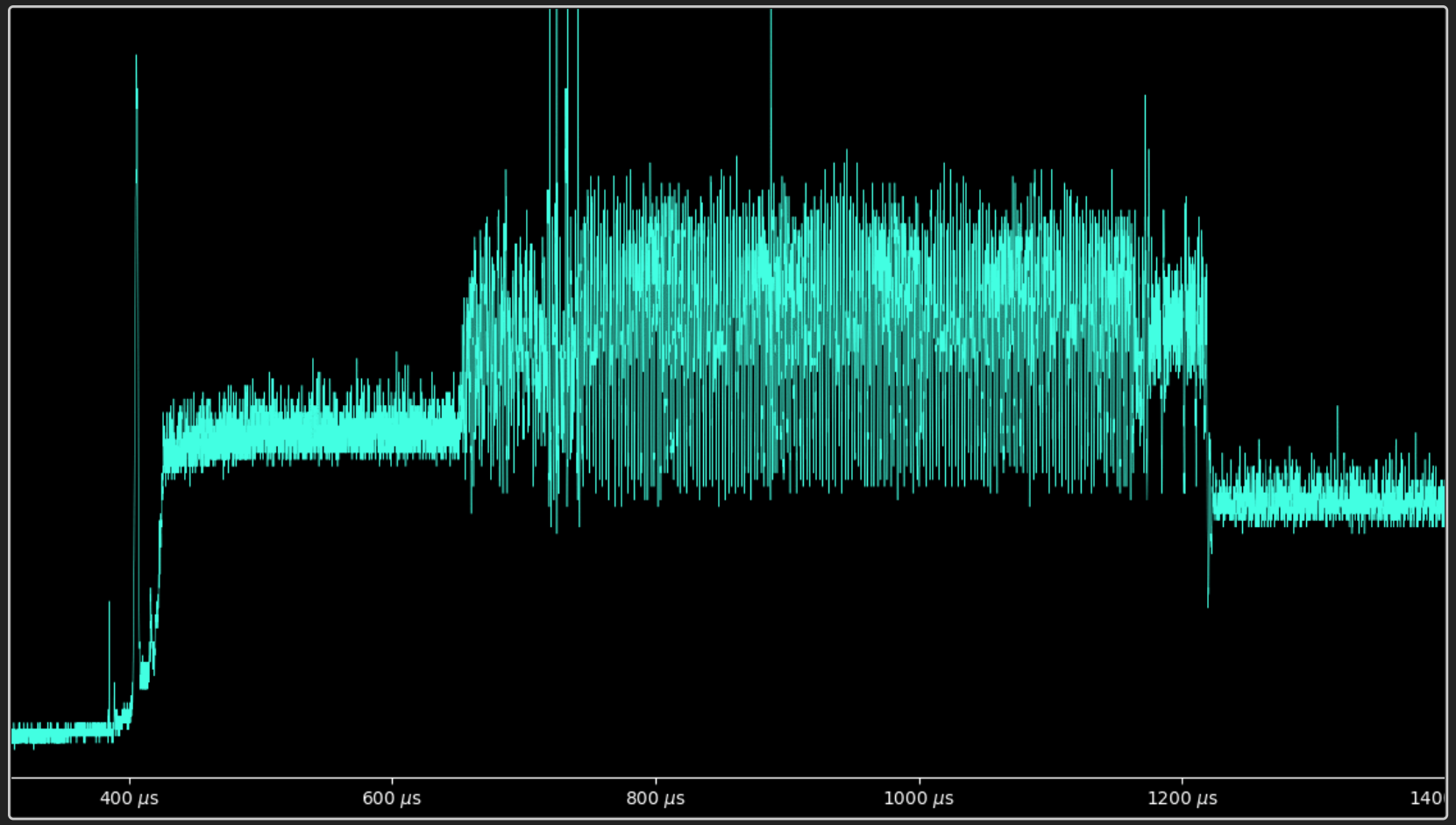
Wake Up power trace



Wake Up power trace



Fault Injection during Wake Up



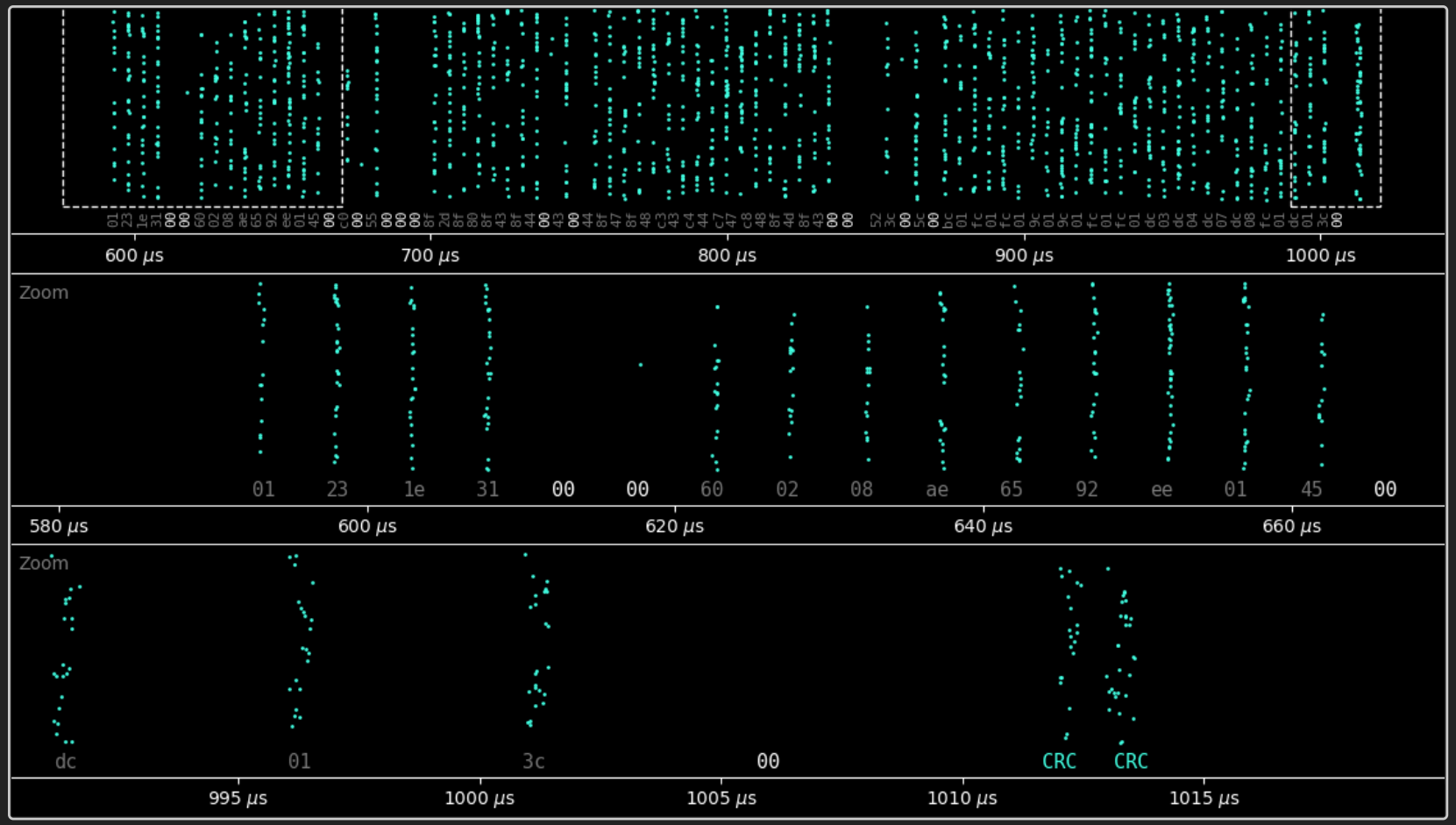
Wake Up: some faults obtained

104814 tests

SELF_TEST_ERROR for 1567 trials:

- Faults when shooting the EEPROM
- Works only for specific shooting times

EEPROM characterization



characterization

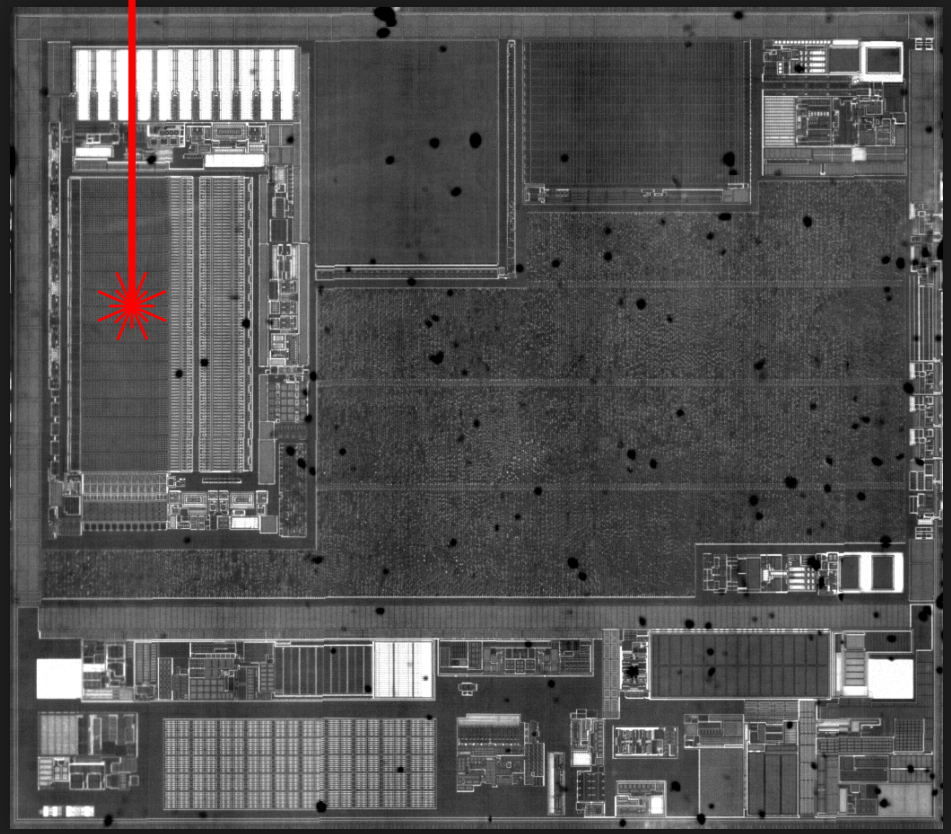
Wake Up triggers a Self-Test procedure

CRC-16 calculated over configuration data
Initial guess was wrong
Faulting self-test won't unlock access to secrets

EEPROM fault model identified

Configuration in EEPROM is not encrypted
Transient stuck at zero of fetched data
Success probability is 97%

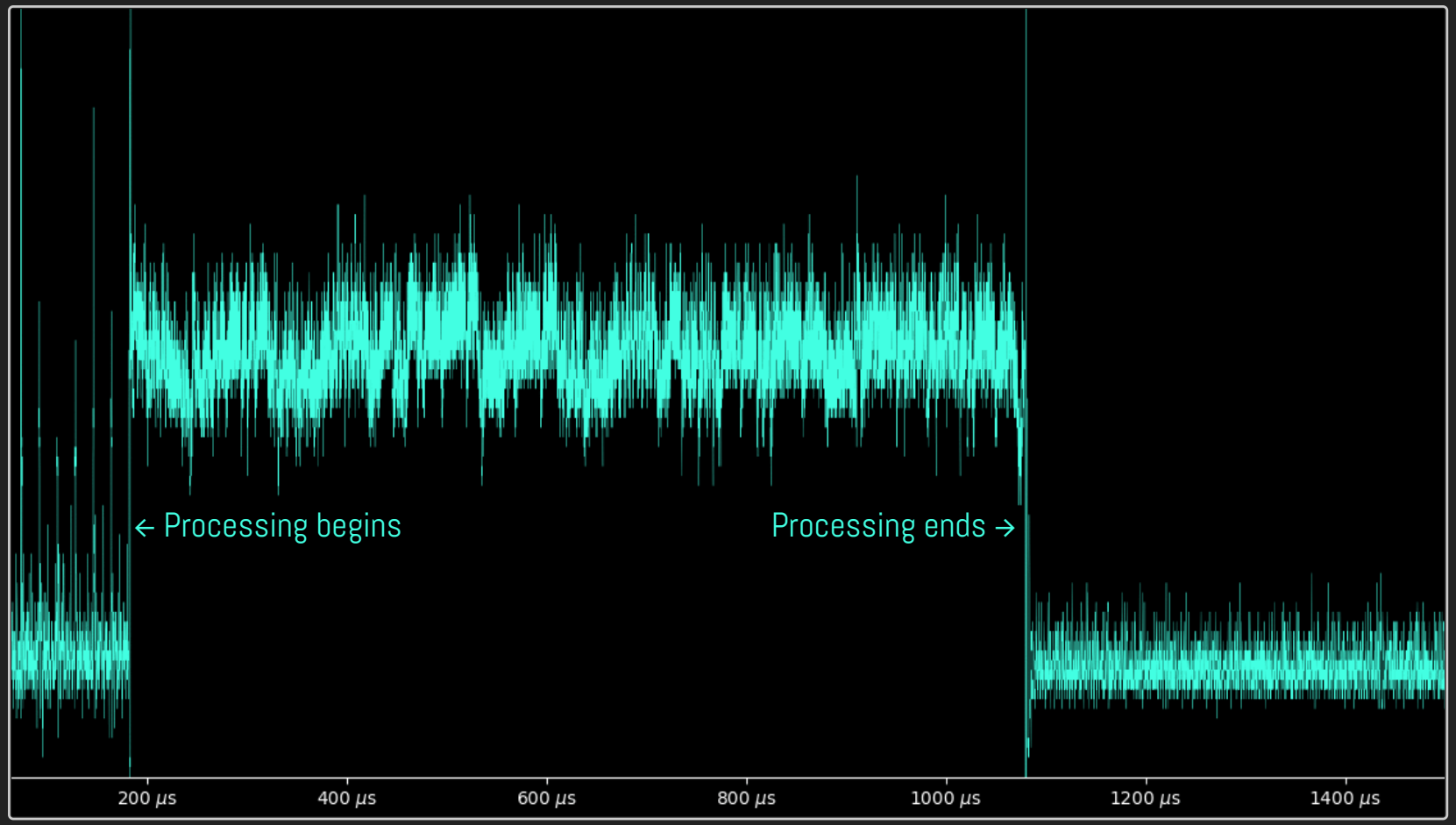
Powerful and reliable attack primitive for next attempts!



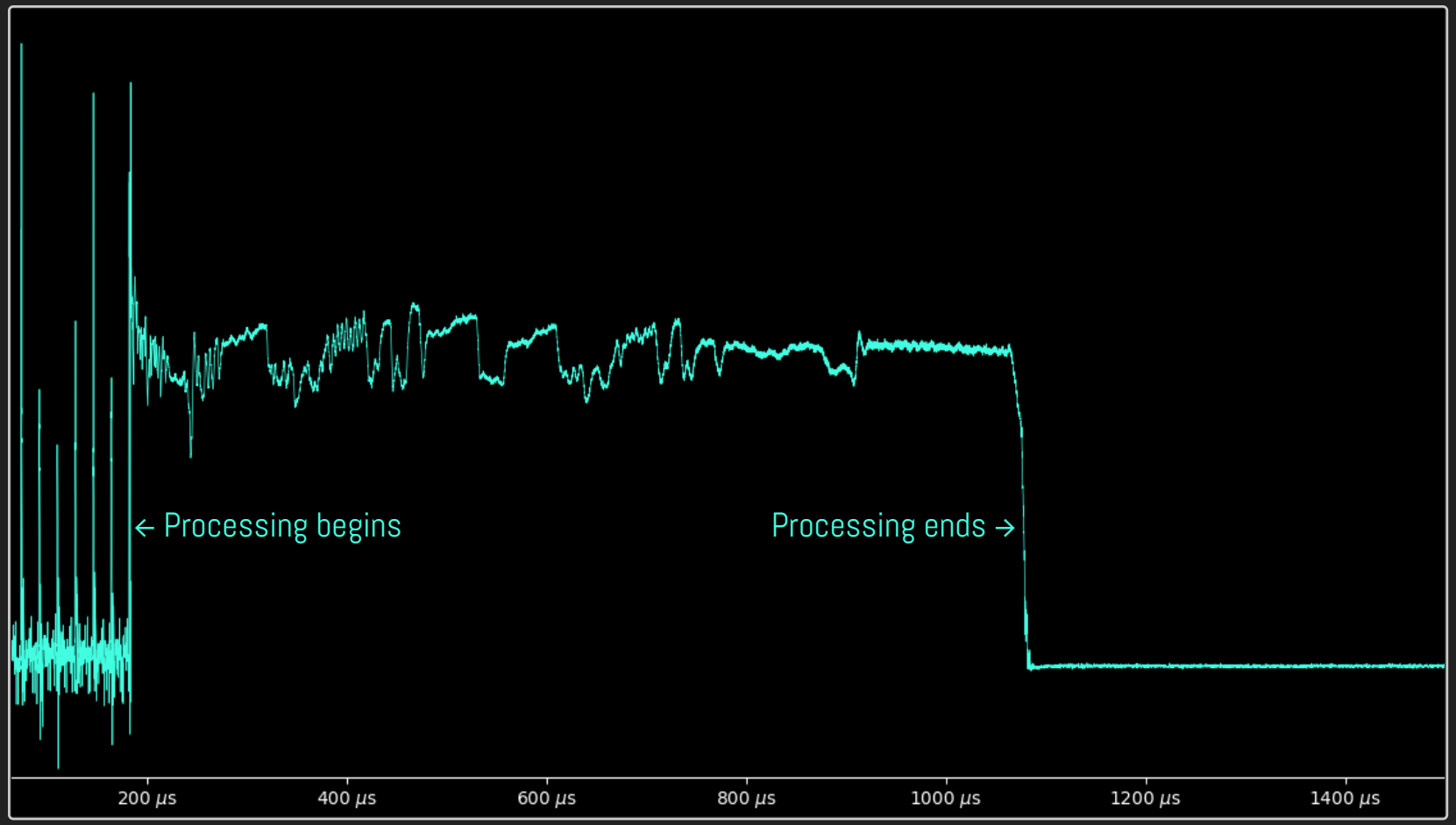
Attack plan

```
1  void read_command(int slot){
2    uint16_t config;
3    eeprom_read(get_config_address(slot), &config, 2);
4
5    if (config & IS_SECRET){
6      // Access denied
7      i2c_transmit(EXECUTION_ERROR);
8      return;
9    }
10
11   // Access granted, fetch the data from EEPROM
12   uint8_t buf[32];
13   eeprom_read(get_data_address(slot), buf, 32);
14   // Send response
15   i2c_transmit(OK, buf);
16 }
```

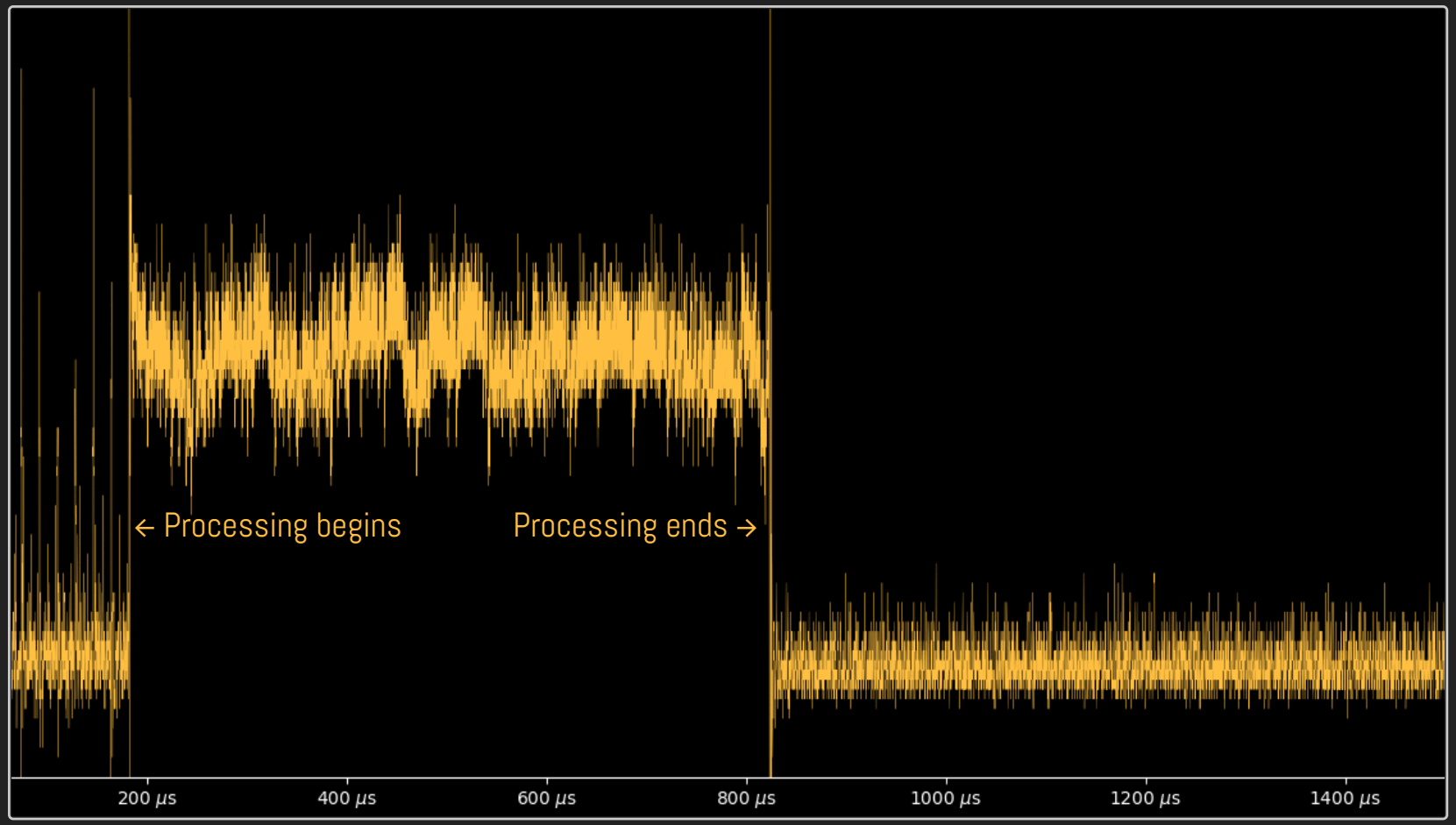
Read OK


← Processing begins
Processing ends →
200 µs
400 µs
600 µs
800 µs
1000 µs
1200 µs
1400 µs

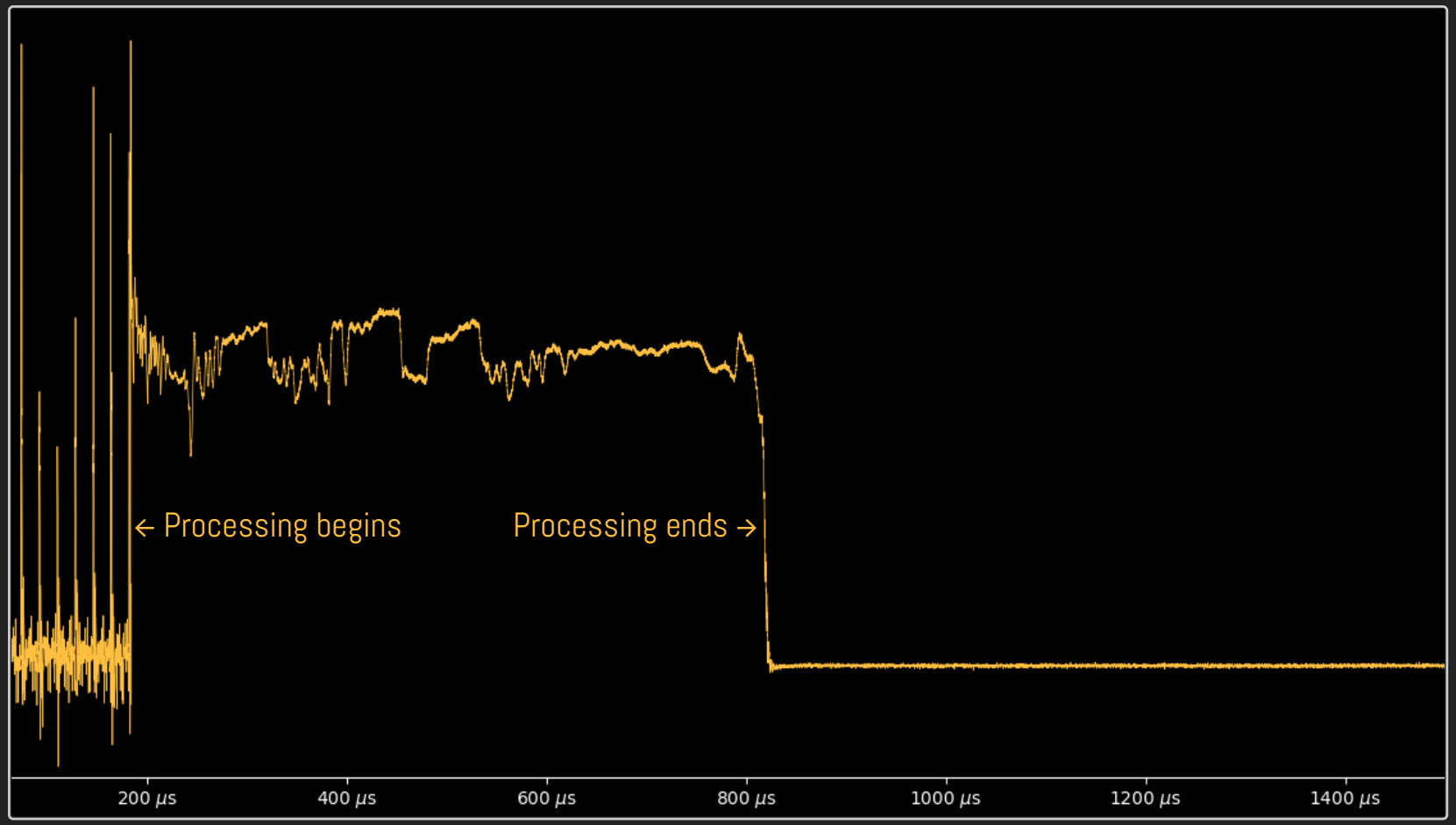
Read OK (averaged)



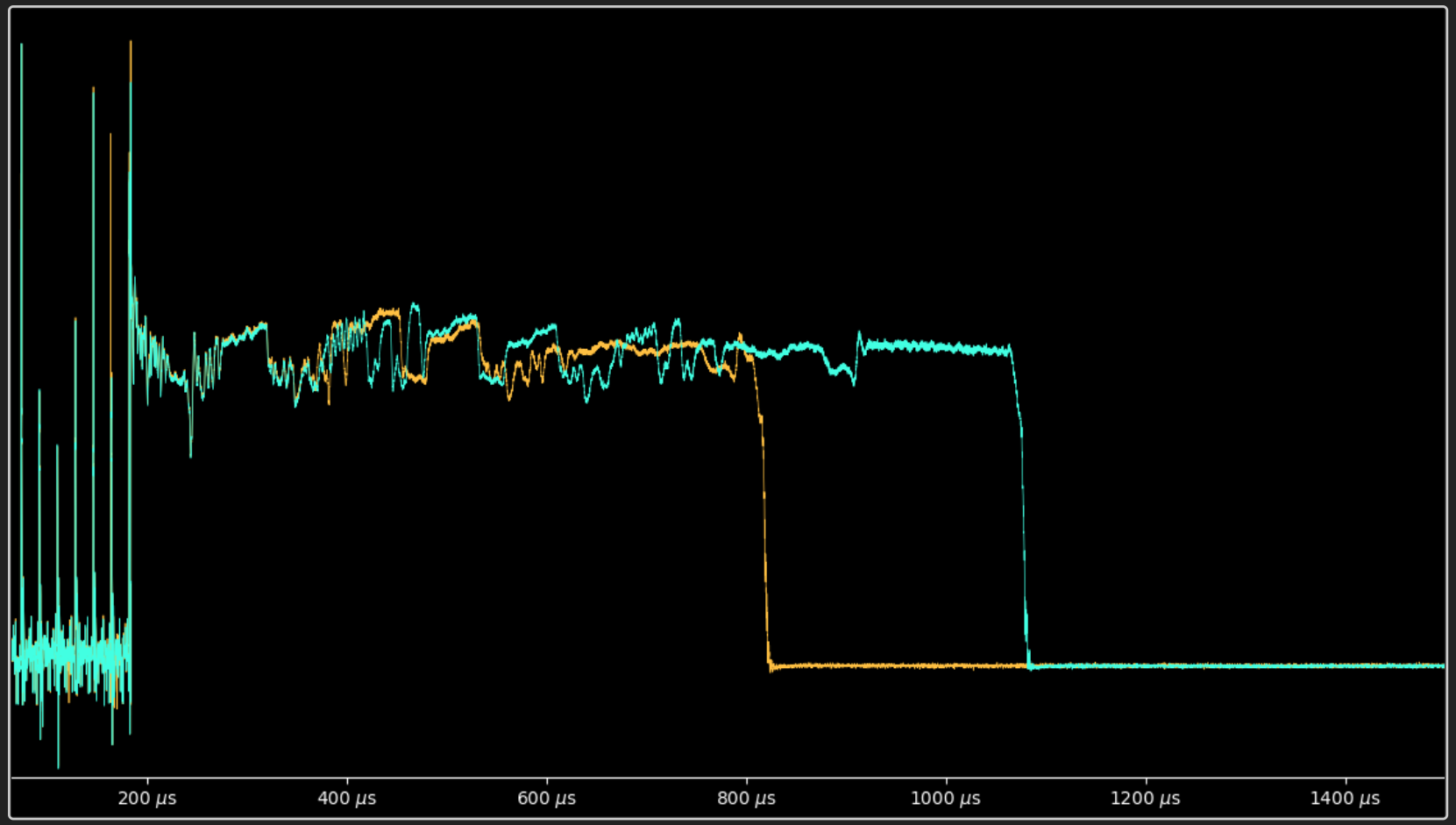
Read EXECUTION_ERROR



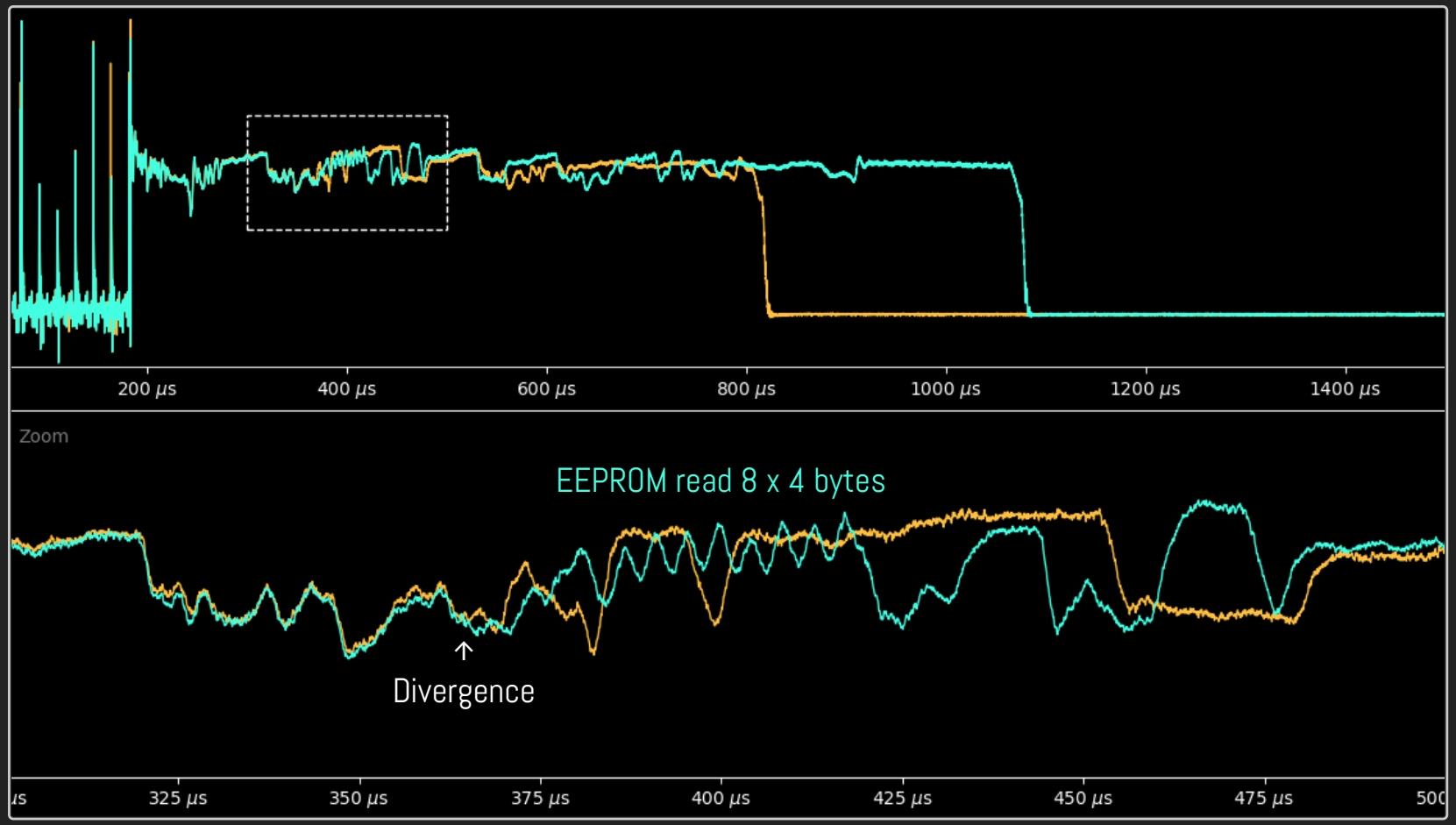
Read EXECUTION_ERROR (averaged)



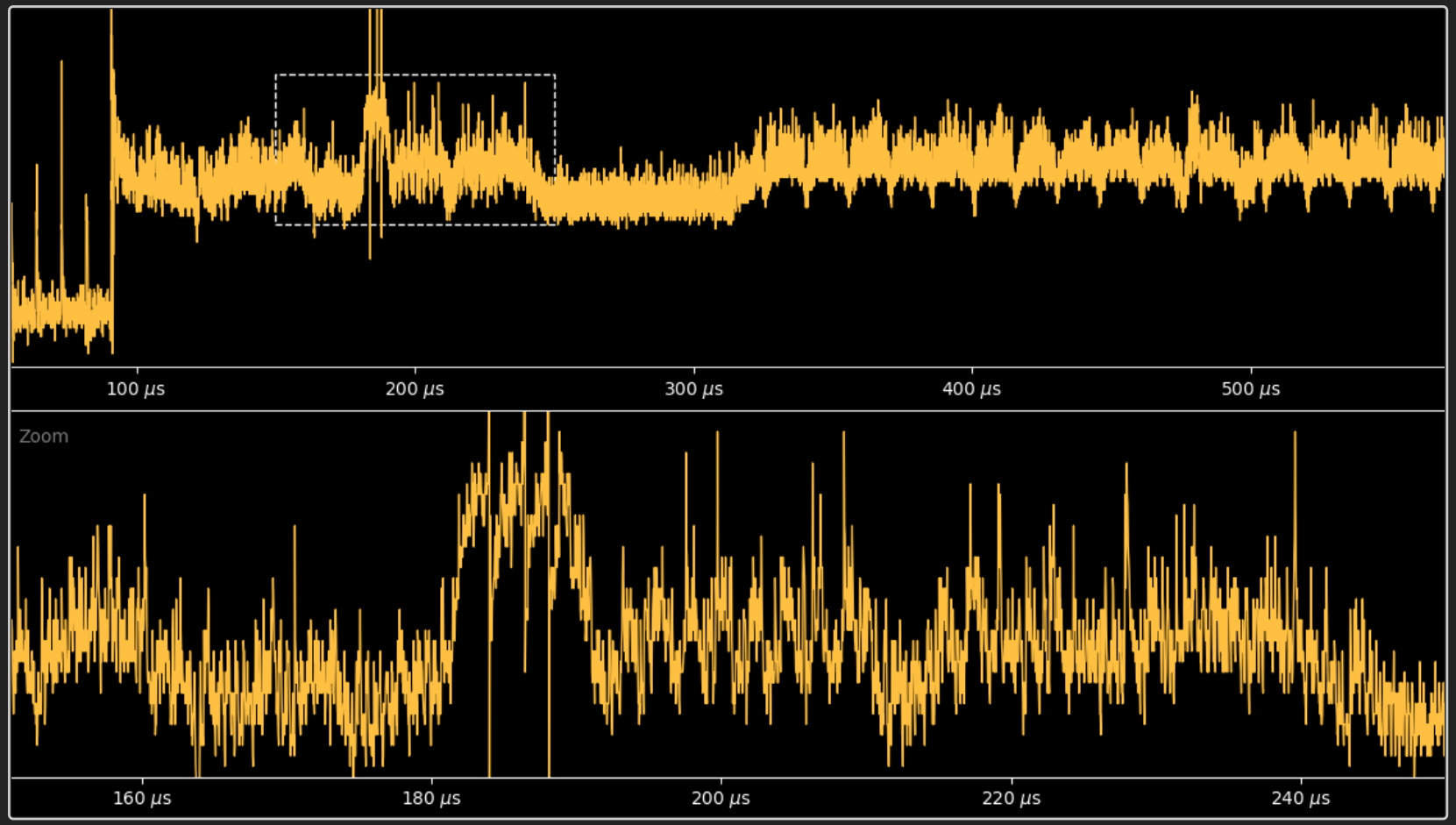
Power traces comparison



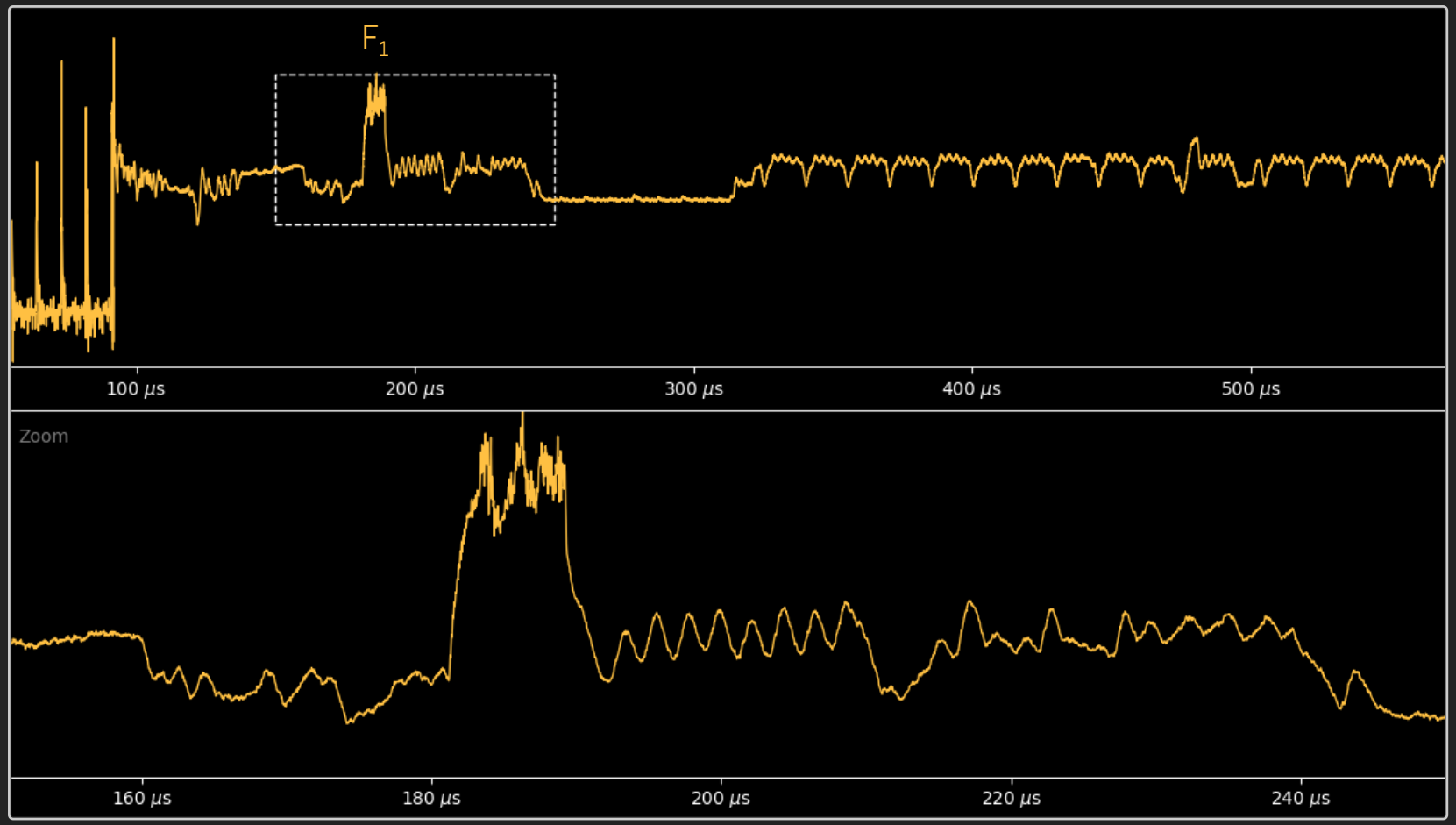
Power traces comparison



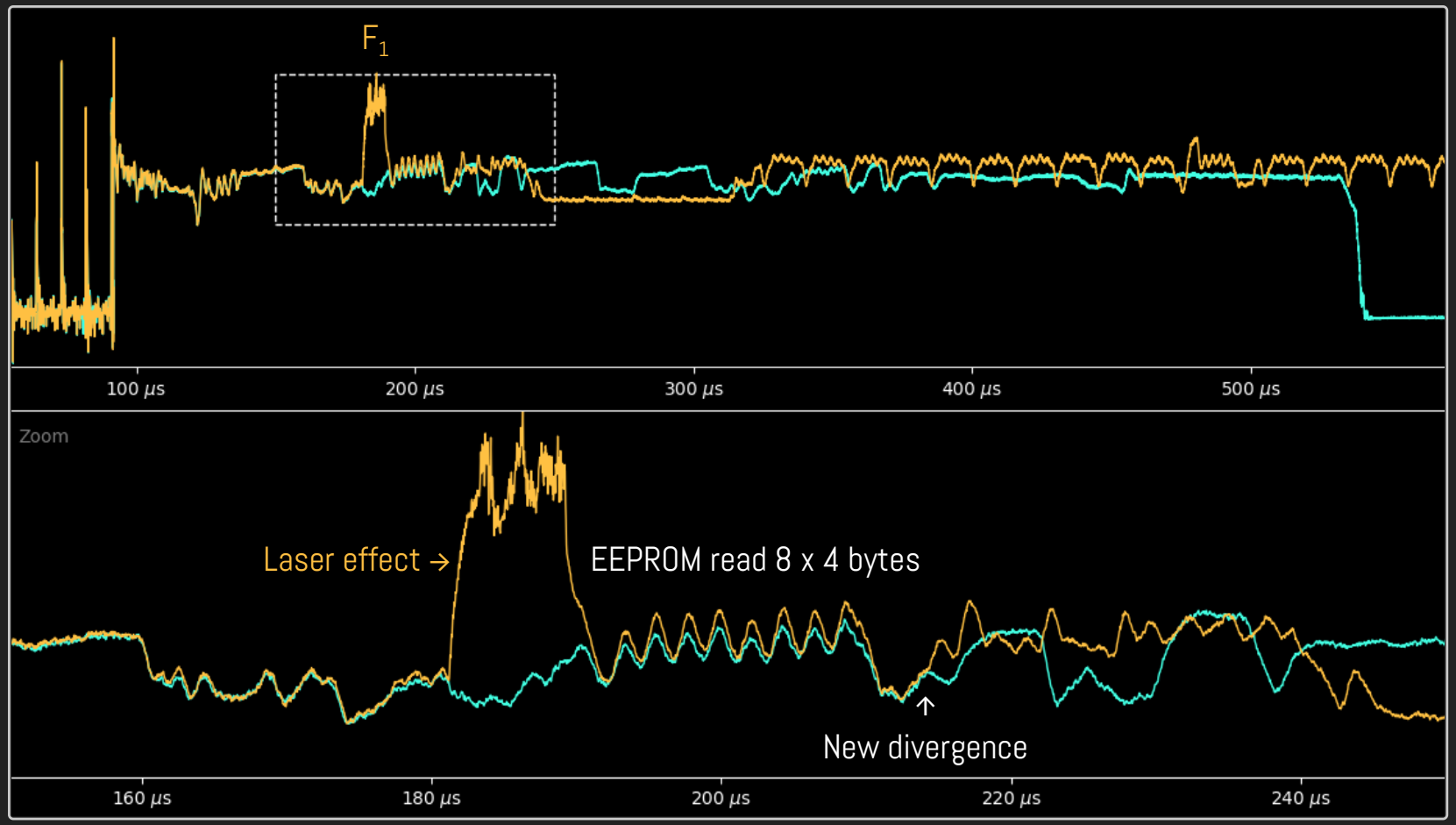
1 fault


100 µs
200 µs
300 µs
400 µs
500 µs
Zoom
160 µs
180 µs
200 µs
220 µs
240 µs

1 fault (averaged)

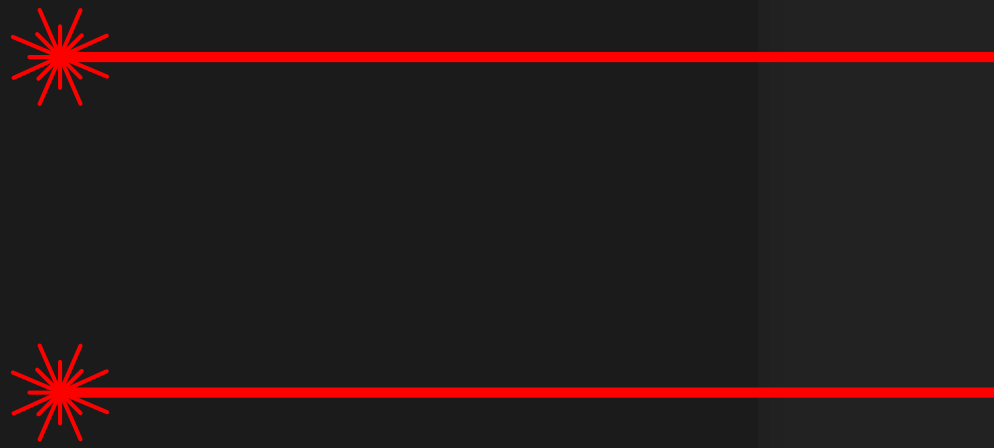


1 fault (averaged)

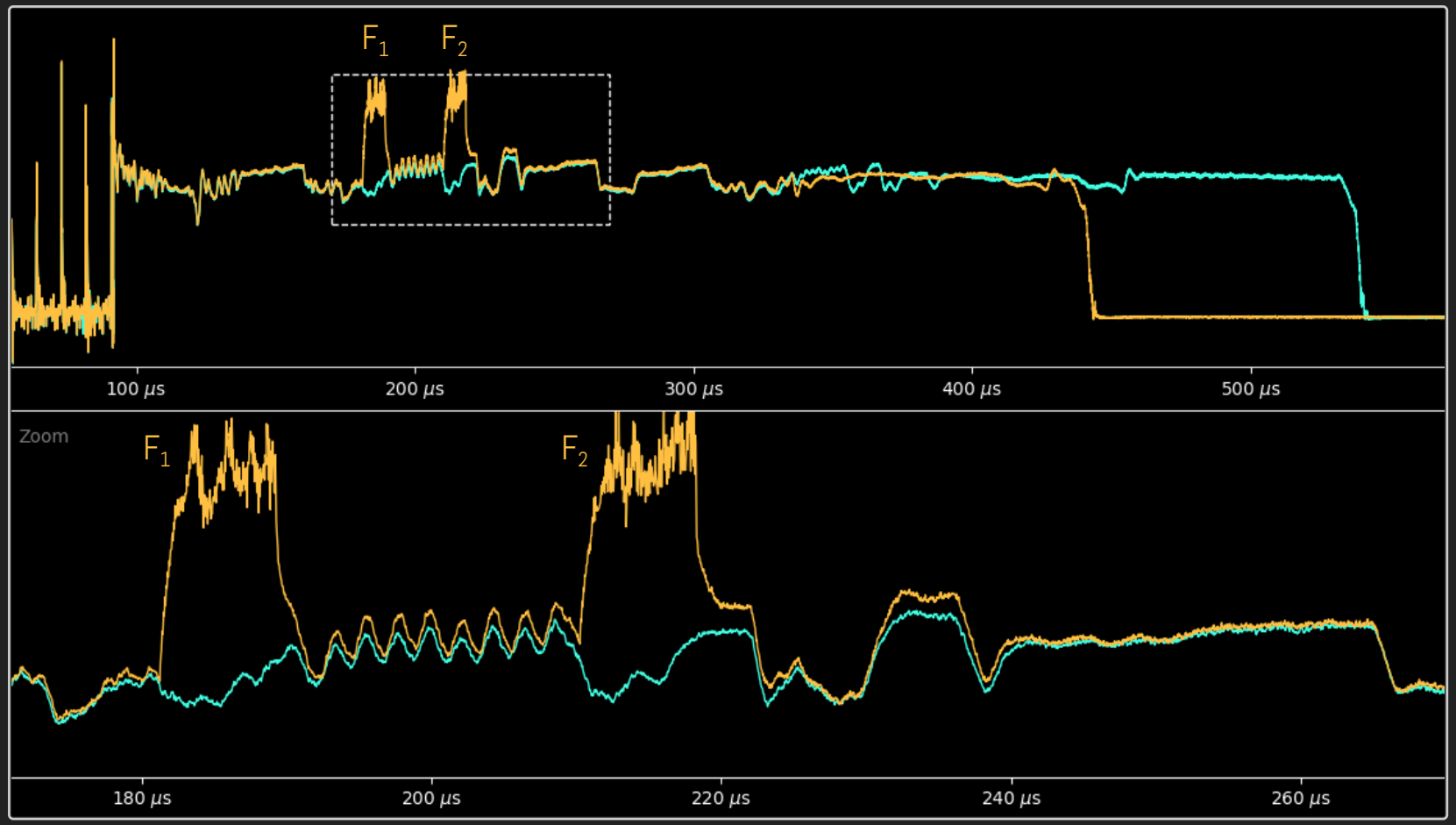


Attack plan

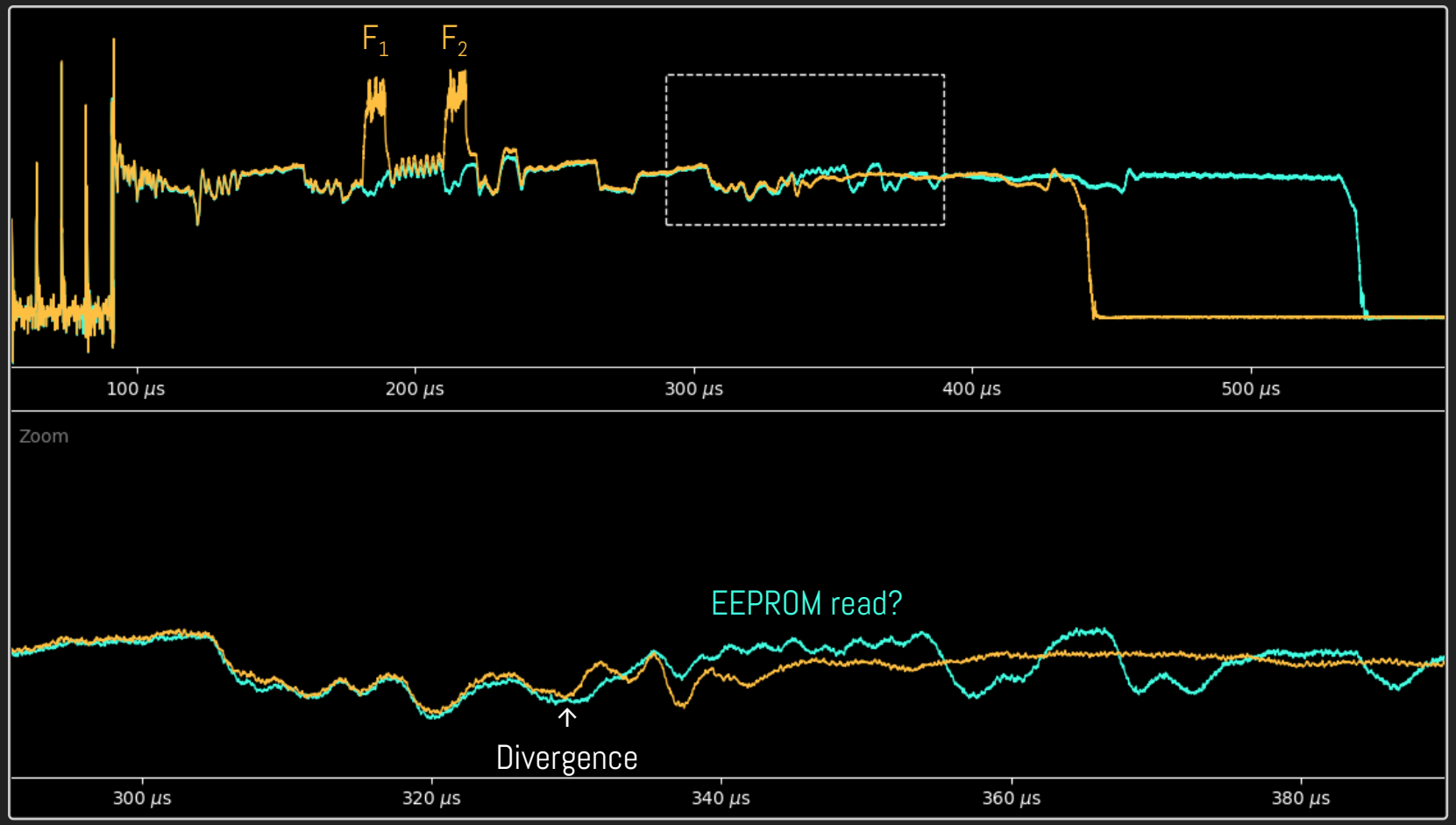
```
1  void read_command(int slot){
2    uint16_t config;
3
4    // First check
5    eeprom_read(get_config_address(slot), &config, 2);
6    if (config & IS_SECRET){
7      i2c_transmit(EXECUTION_ERROR);
8      return;
9    }
10
11   uint8_t buf[32];
12   eeprom_read(get_data_address(slot), buf, 32);
13
14   // Second check
15   eeprom_read(get_config_address(slot), &config, 2);
16   if (config & IS_SECRET){
17     i2c_transmit(EXECUTION_ERROR);
18     return;
19   }
20
21   // Send response
22   i2c_transmit(OK, buf);
23 }
```



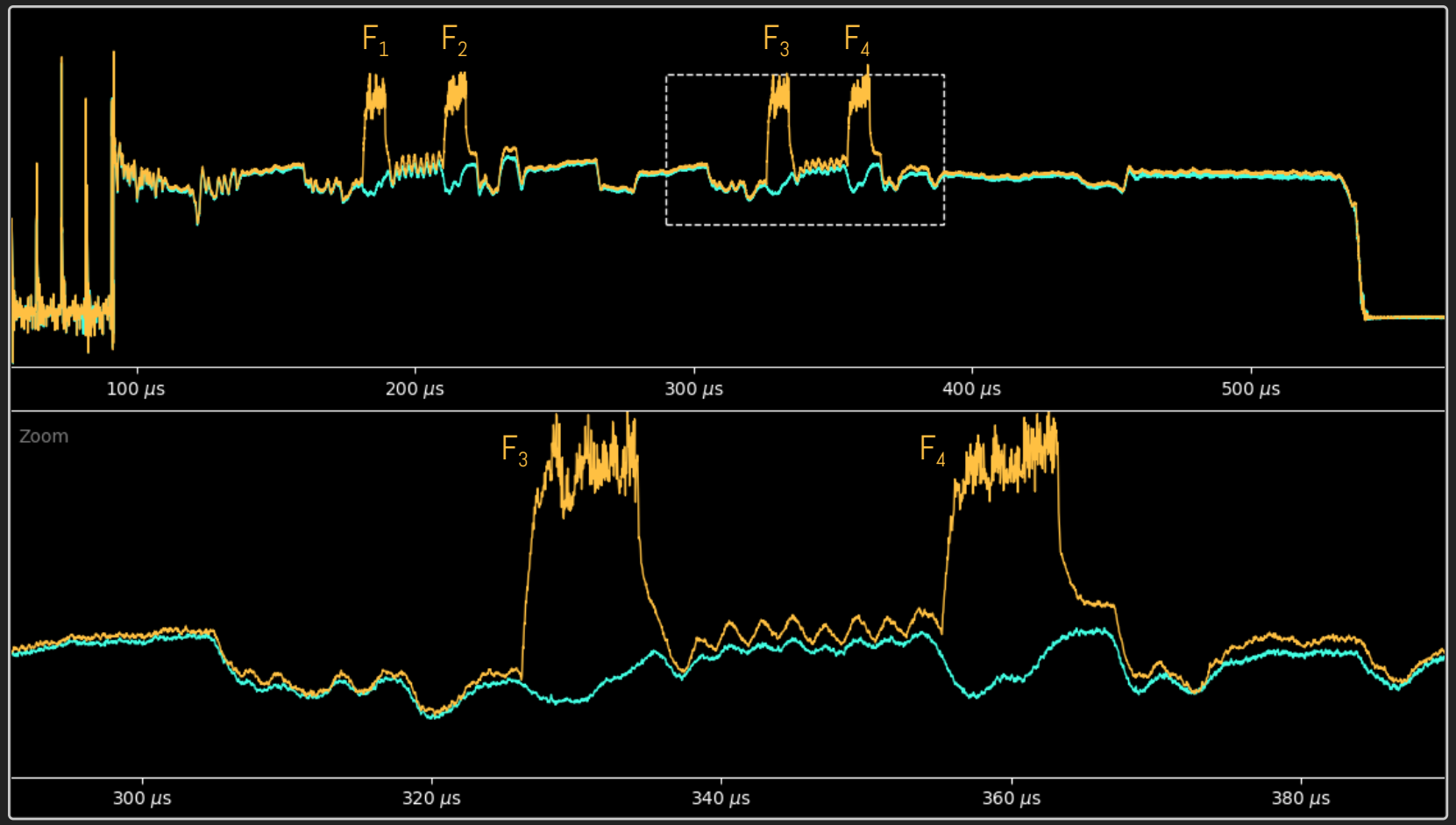
2 faults (averaged)



2 faults (averaged)



4 faults (averaged)



Success?

4 faults injected during execution of the Read command

OK status and 32 data bytes returned by the chip!

Data is incorrect
and looks encrypted...

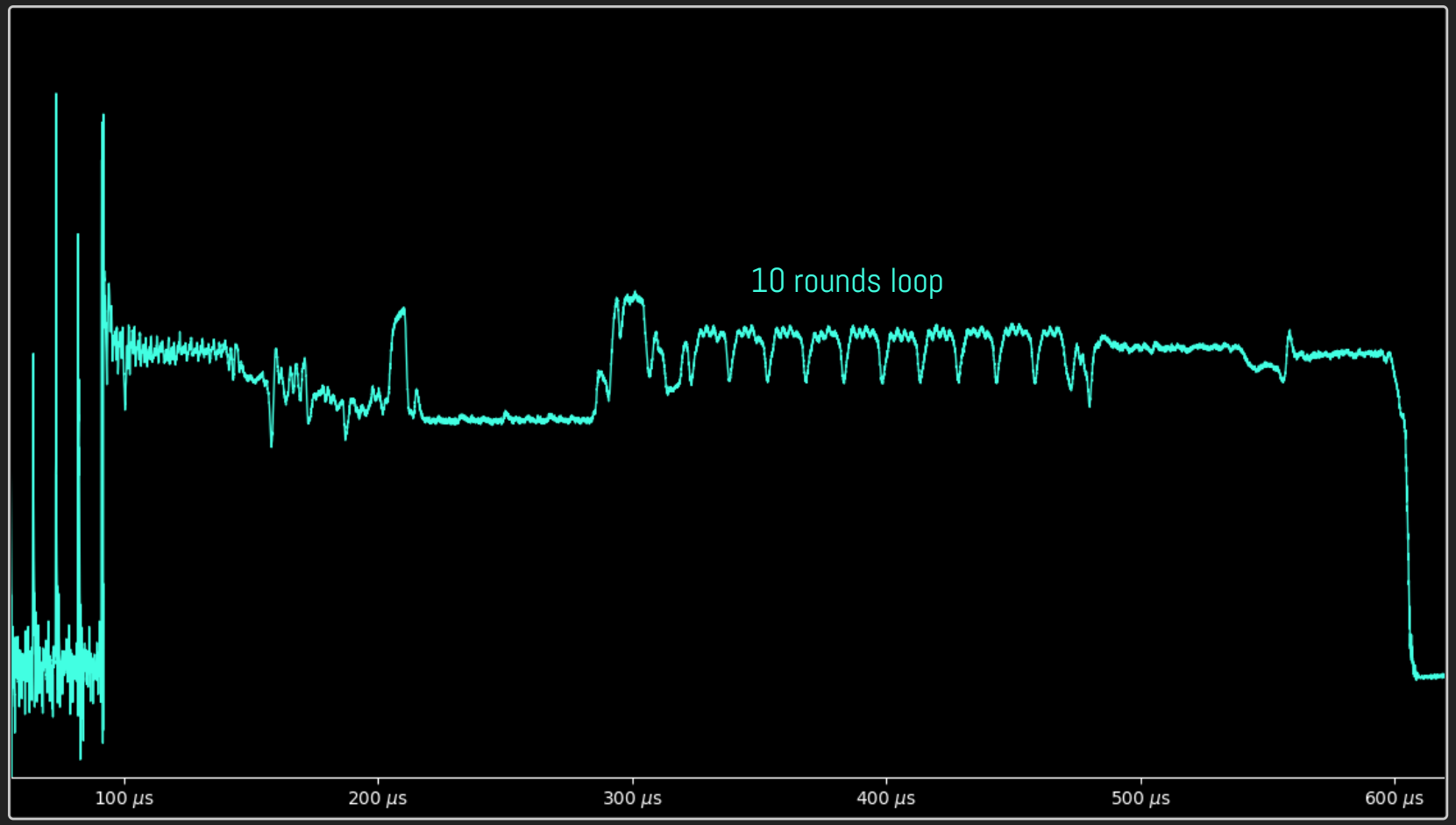
AES command

ATECC608A new command to encrypt or decrypt using AES

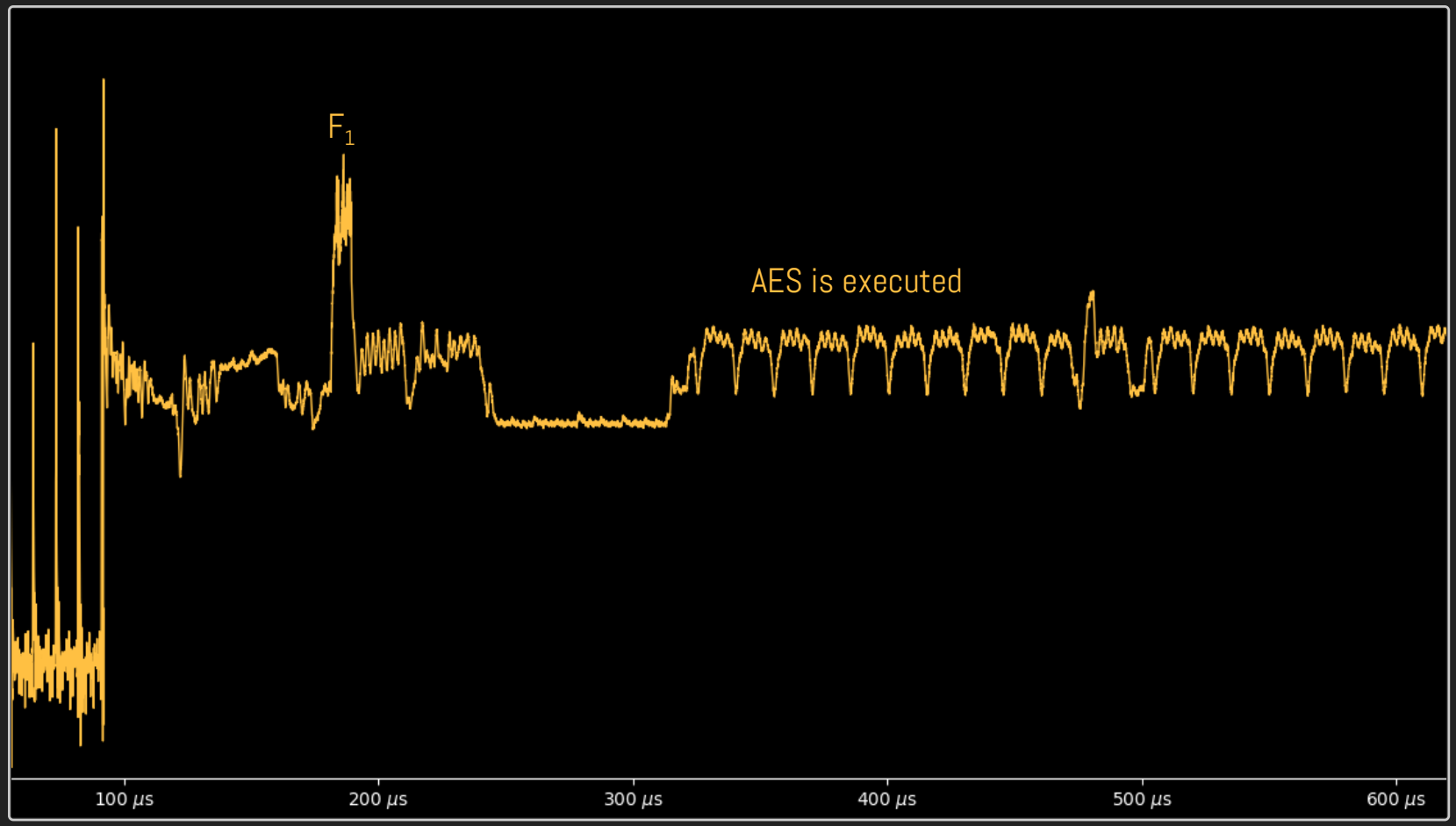
Hardware AES-128

Let's have a look...

AES command power trace (averaged)



Faulted Read power trace (averaged)



Ah! New hypothesis

Secret files are stored encrypted in EEPROM
Public files are not...

Execution path MUST NOT be the same as for public files

The firmware checks the IS_SECRET flag in EEPROM
to apply decryption or not

Faults F_2 and F_4 prevents decryption

Attack plan

```
1  void internal_get_slot_data(int slot, uint8_t* dest){
2    eeprom_read(get_data_address(slot), dest, 32);
3    uint16_t config;
4    eeprom_read(get_config_address(slot), &config, 2); // Don't fault here!
5    if (config & IS_SECRET){
6      aes_decrypt(dest, SOME_INTERNAL_KEY);
7    }
8  }
9
10 void read_command(int slot){
11   uint16_t config;
12   // First check
13   eeprom_read(get_config_address(slot), &config, 2);
14   if (config & IS_SECRET){
15     i2c_transmit(EXECUTION_ERROR);
16     return;
17   }
18   // First data fetch
19   uint8_t buf_a[32];
20   internal_get_slot_data(slot, buf_a);
21
22   // Second check
23   eeprom_read(get_config_address(slot), &config, 2);
24   if (config & IS_SECRET){
25     i2c_transmit(EXECUTION_ERROR);
26     return;
27   }
28   // Second data fetch
29   uint8_t buf_b[32];
30   internal_get_slot_data(slot, buf_b);
31
```

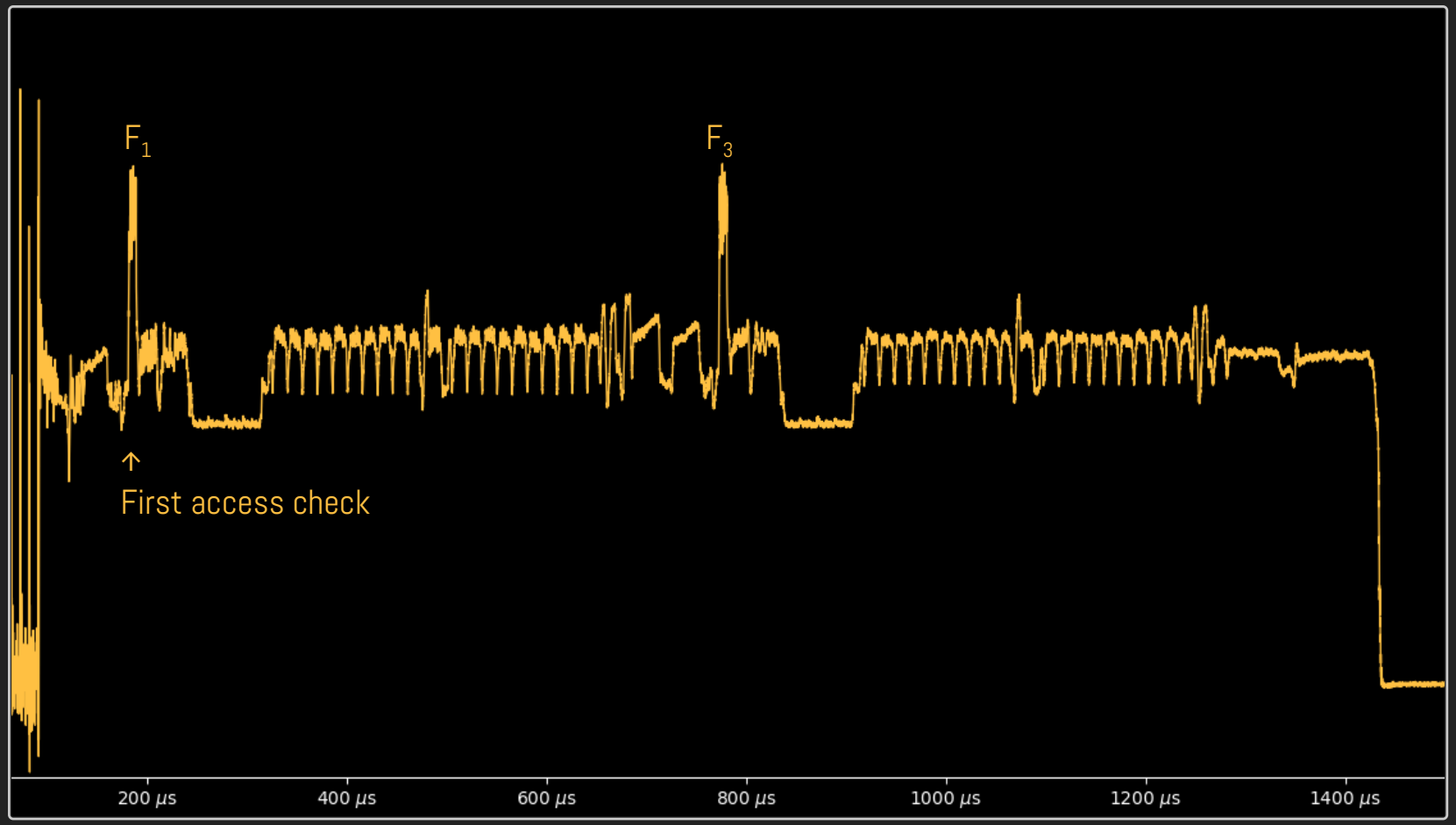
Attack plan

```
 8 }
 9
10 void read_command(int slot){
11   uint16_t config;
12   // First check
13   eeprom_read(get_config_address(slot), &config, 2);
14   if (config & IS_SECRET){
15     i2c_transmit(EXECUTION_ERROR);
16     return;
17   }
18   // First data fetch
19   uint8_t buf_a[32];
20   internal_get_slot_data(slot, buf_a);
21
22   // Second check
23   eeprom_read(get_config_address(slot), &config, 2);
24   if (config & IS_SECRET){
25     i2c_transmit(EXECUTION_ERROR);
26     return;
27   }
28   // Second data fetch
29   uint8_t buf_b[32];
30   internal_get_slot_data(slot, buf_b);
31
32   // Double read checking
33   if (memcmp(buf_a, buf_b, 32)){
34     i2c_transmit(EXECUTION_ERROR);
35   } else {
36     i2c_transmit(OK, buf_a);
37   }
38 }
```

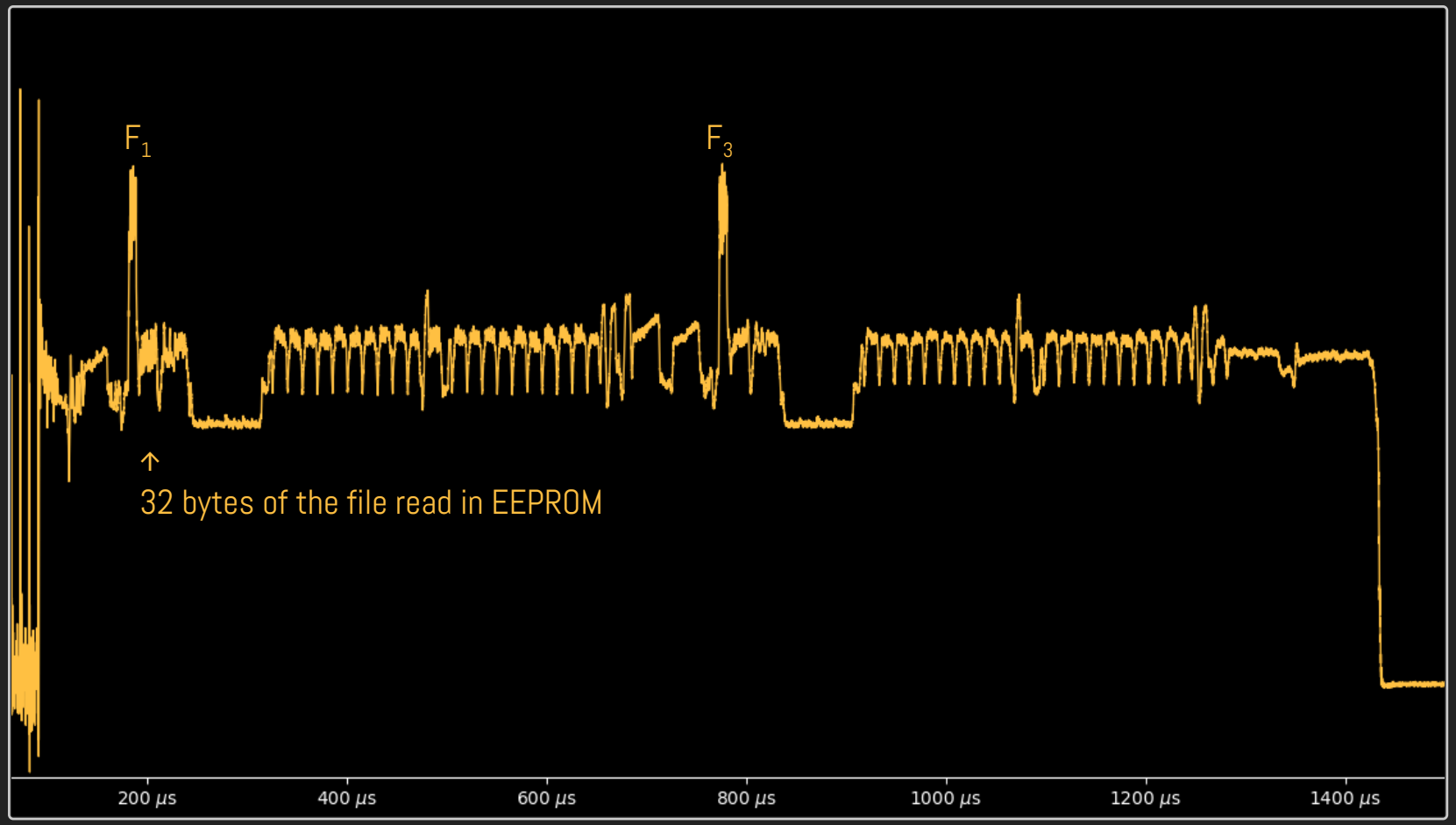
Attack plan

```
 8 }
 9
10 void read_command(int slot){
11   uint16_t config;
12   // First check
13   eeprom_read(get_config_address(slot), &config, 2);
14   if (config & IS_SECRET){
15     i2c_transmit(EXECUTION_ERROR);
16     return;
17   }
18   // First data fetch
19   uint8_t buf_a[32];
20   internal_get_slot_data(slot, buf_a);
21
22   // Second check
23   eeprom_read(get_config_address(slot), &config, 2);
24   if (config & IS_SECRET){
25     i2c_transmit(EXECUTION_ERROR);
26     return;
27   }
28   // Second data fetch
29   uint8_t buf_b[32];
30   internal_get_slot_data(slot, buf_b);
31
32   // Double read checking
33   if (memcmp(buf_a, buf_b, 32)){
34     i2c_transmit(EXECUTION_ERROR);
35   } else {
36     i2c_transmit(OK, buf_a);
37   }
38 }
```

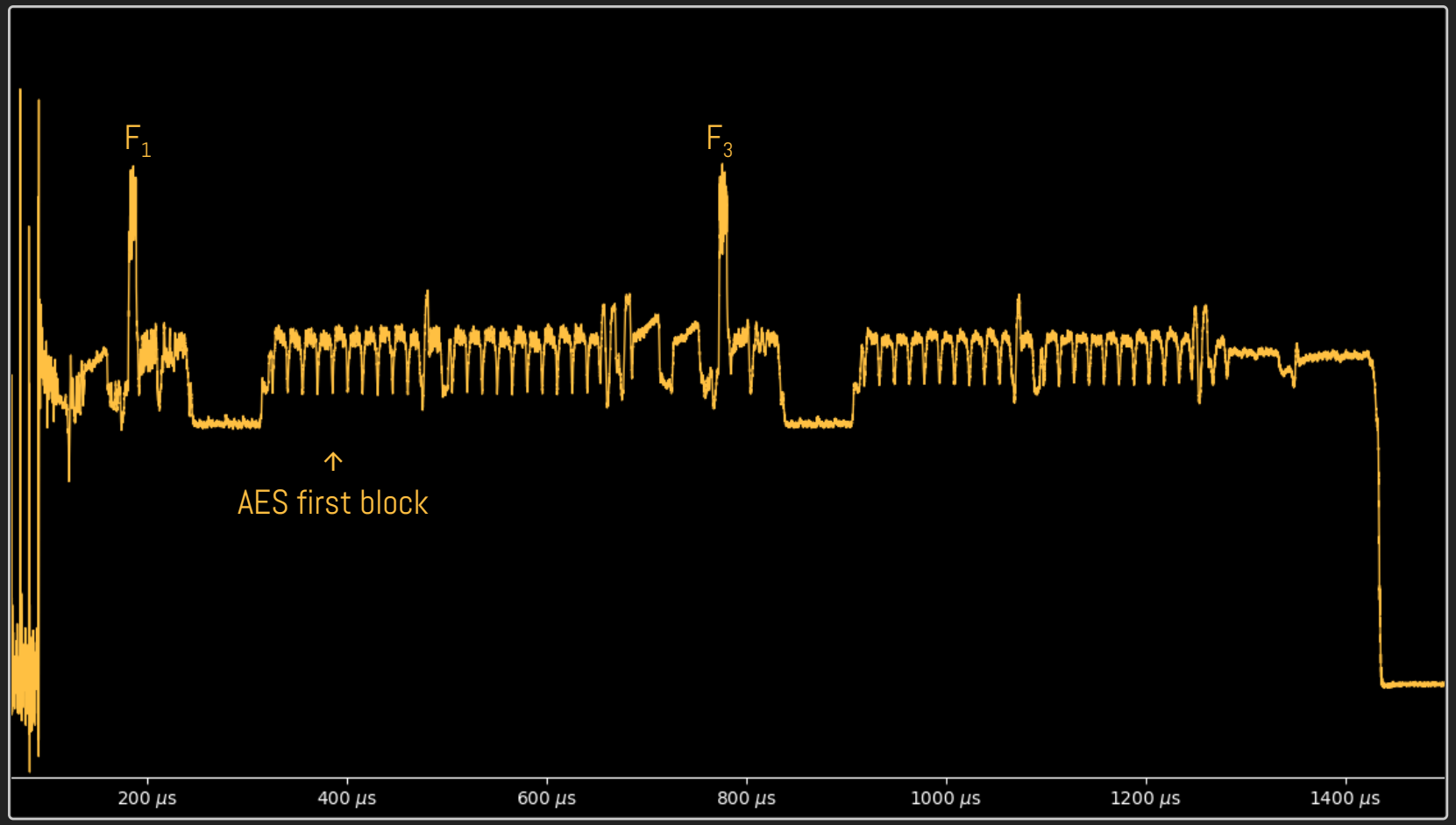
2 faults OK (averaged)



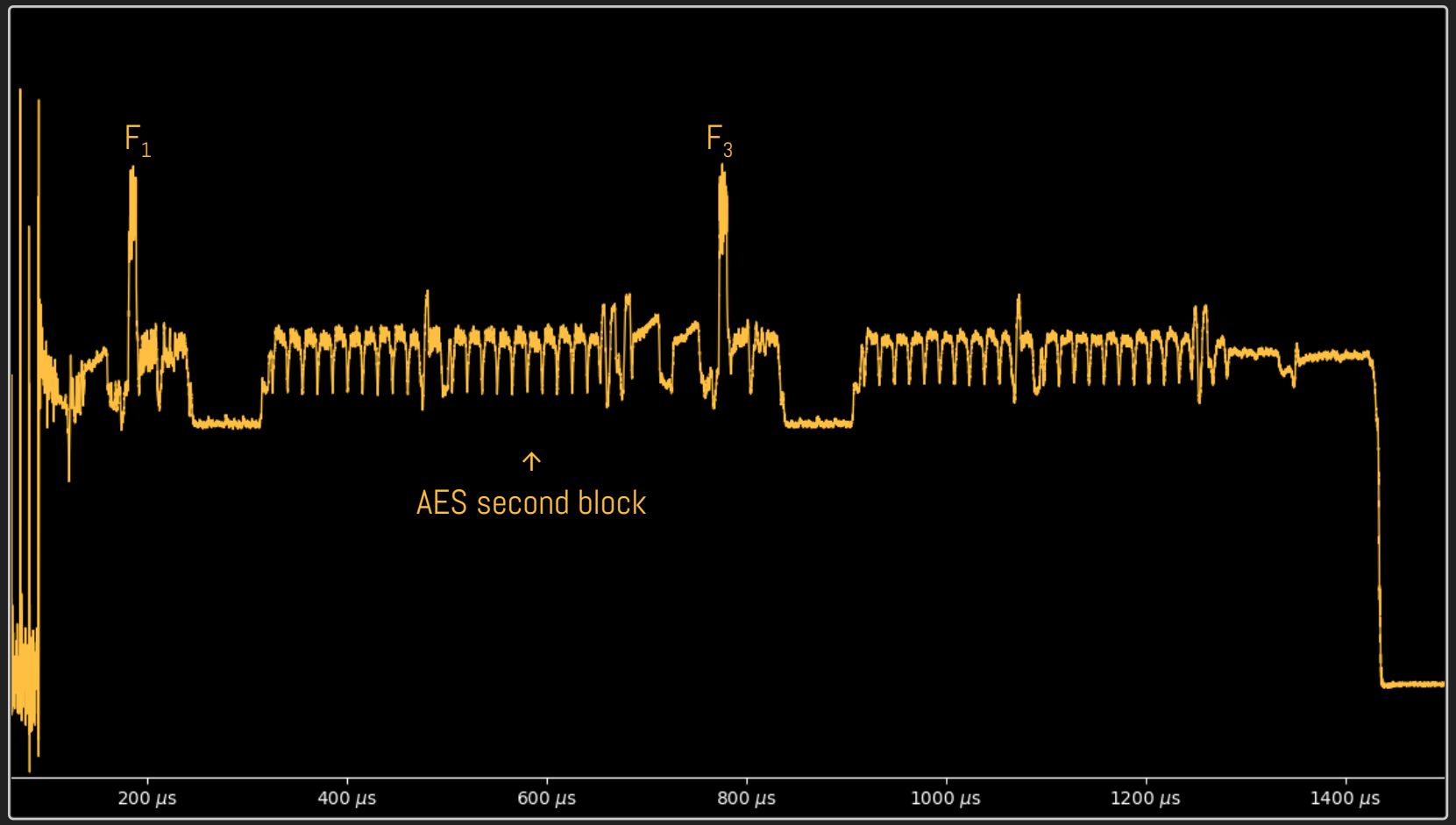
2 faults OK (averaged)



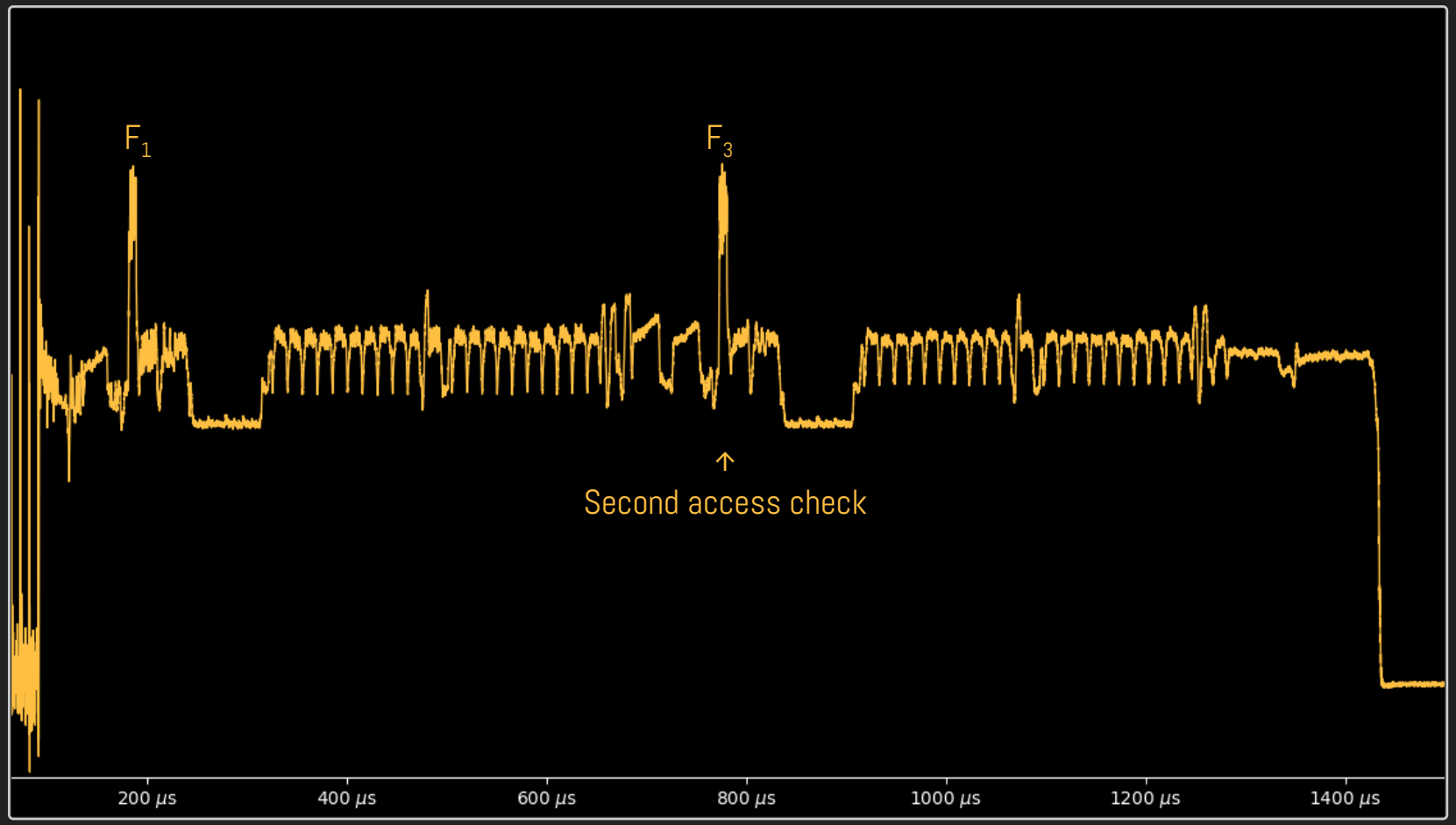
2 faults OK (averaged)



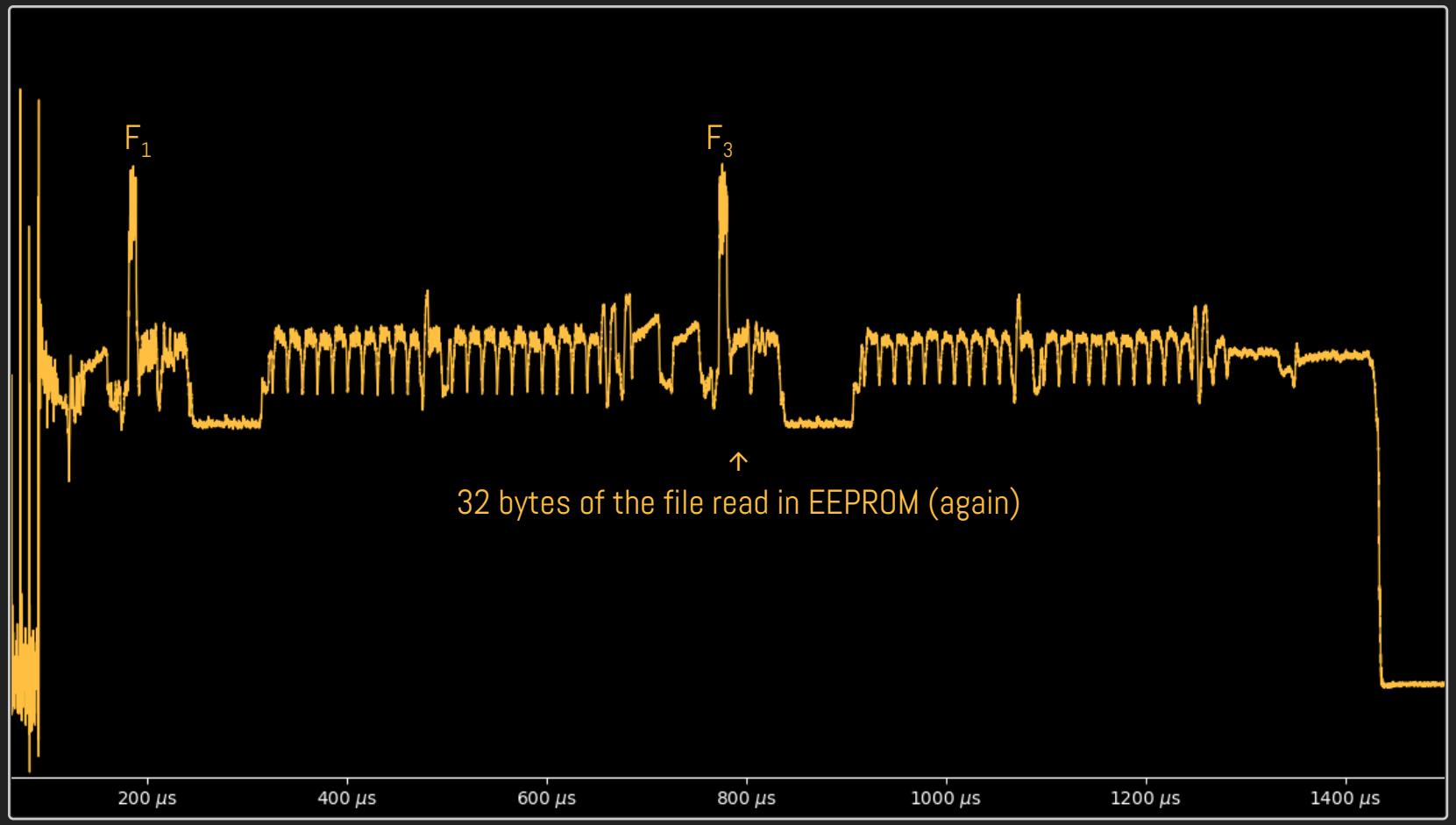
2 faults OK (averaged)



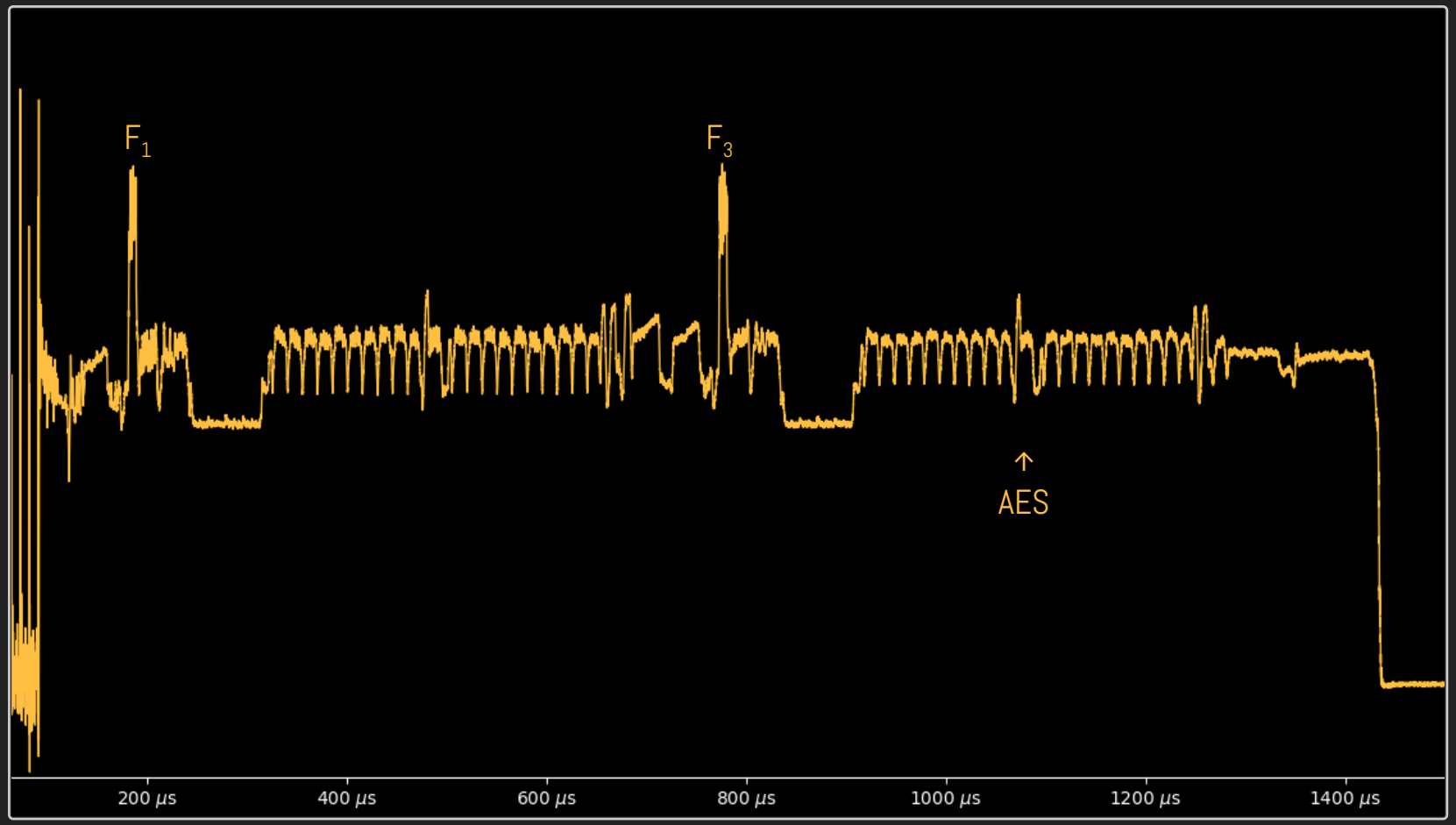
2 faults OK (averaged)



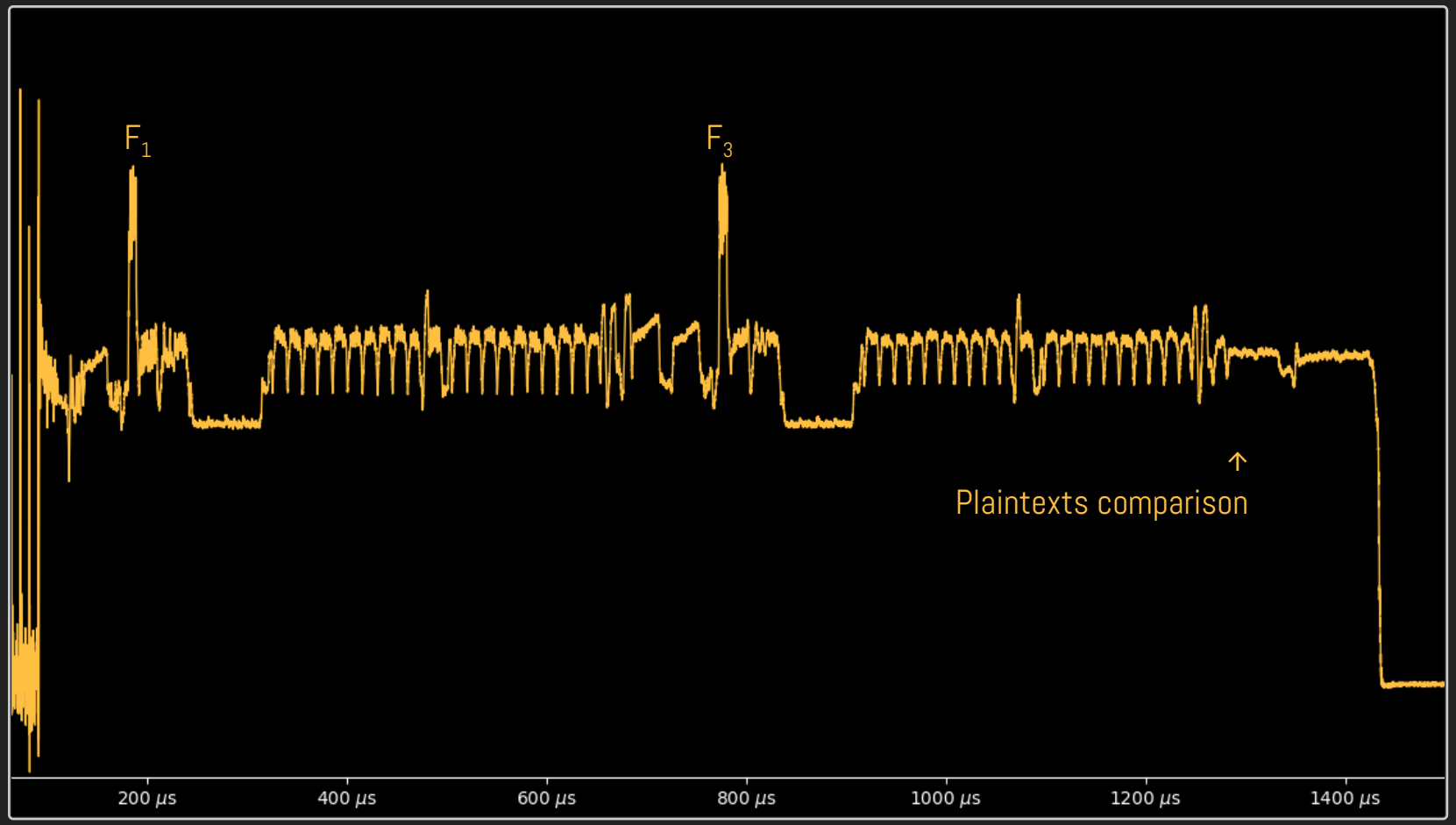
2 faults OK (averaged)



2 faults OK (averaged)



2 faults OK (averaged)



Success!

2 faults injected during Read command

Returned data is now correct!

Attack is reproducible

Tested on a real Coldcard Mk3 device:
Seed recovered successfully

Difficulties

Averaging traces is not always possible

Multiple Fault Injection is difficult

With 4 faults:

$$P_{success} = P(F_1).P(F_2).P(F_3).P(F_4)$$

If $P(F_i) = 5\%$

Then $P_{success} = 1/160000$

Parameters of each fault injection must be optimized

Our success rates

$$P(F_1) = 95.8\%$$

$$P(F_2) = 91.1\%$$

$$P(F_3) = 97.8\%$$

$$P(F_4) = 93.5\%$$

$$P(F_1).P(F_2).P(F_3).P(F_4) = 79.8\%$$

$$P(F_1).P(F_3) = 93.7\%$$

Possible counter-measures

Light sensors

Clock jitter, random delays

Memory protection with Error Detection Codes

Killcard

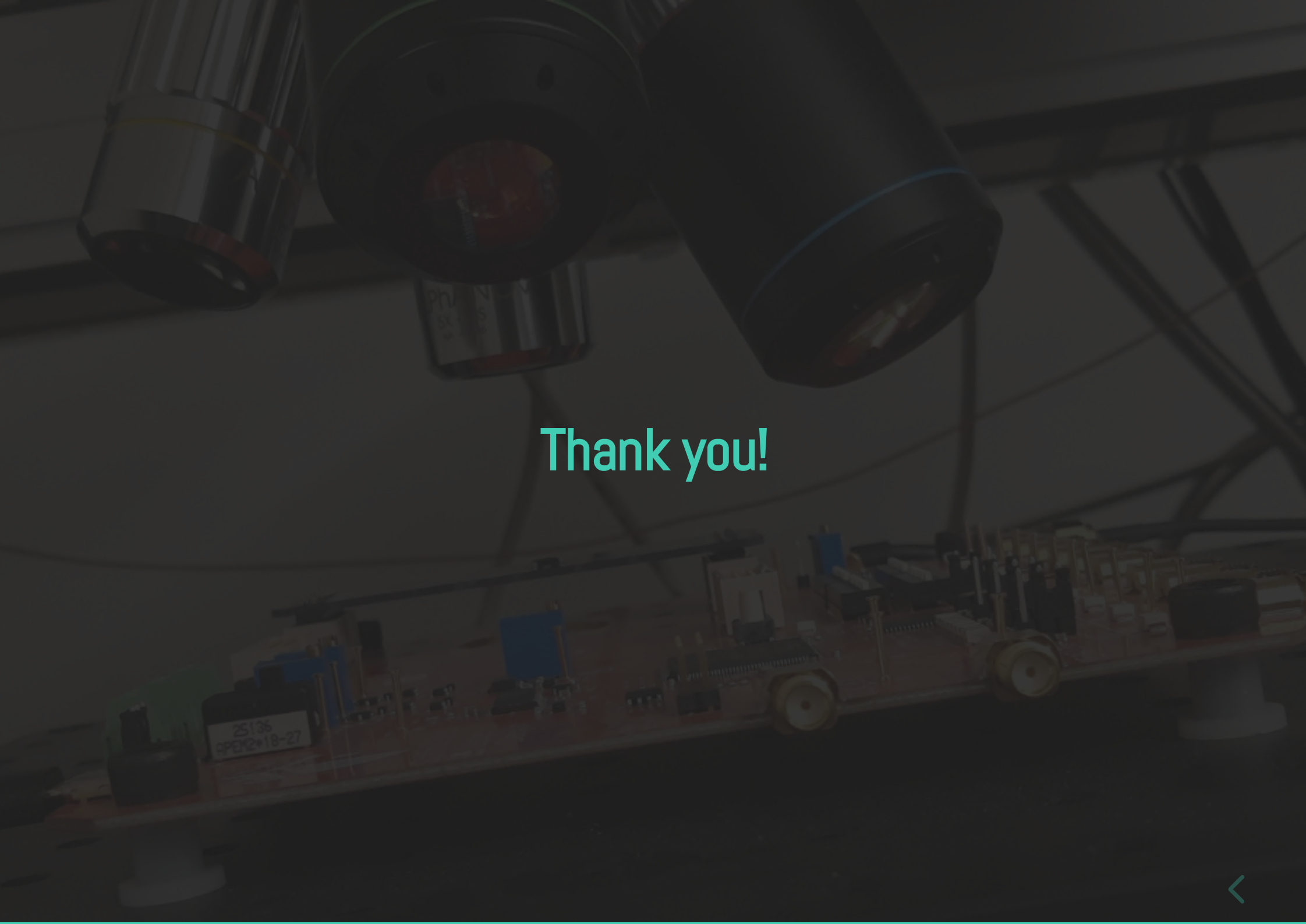
Conclusion

ATECC608A vulnerable against Double Laser Fault Injection

Old Coldcard Mk3 are vulnerable
Check for you SE version in the device!

Secure Element
JIL High security level
Equivalent to EAL6/7 according to Microchip

Much more resistant than the previous ATECC508A generation
But should include more hardware counter-measures


Thank you!