Introduction to Fault Attacks

Ileana Buhan, March 2024
@ileanabuhan

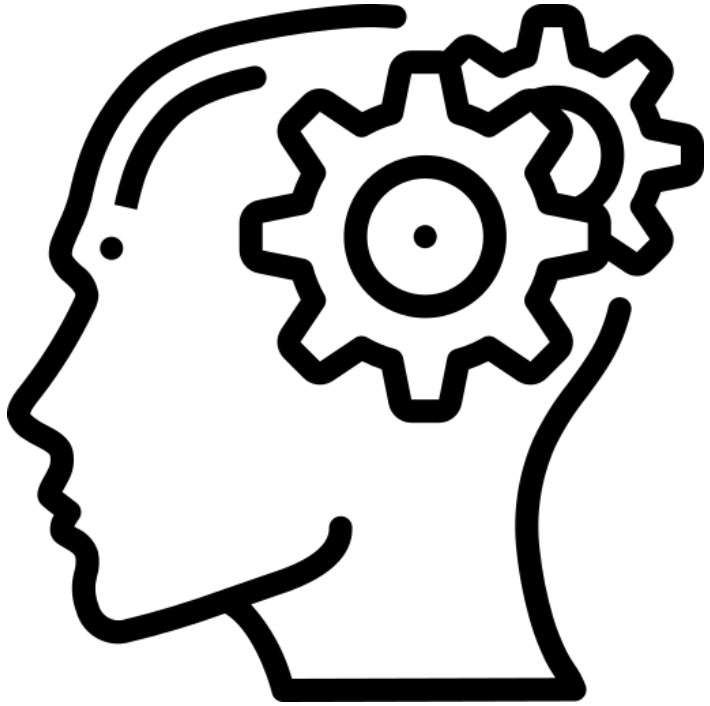
Radboud University

Take a few minutes

Why are fault attacks dangerous?

Who is our adversary?

Fault attacks vs Side-channel attacks?



Attack categories

Side-channel attacks:

- passive attacks - the device behaves within specifications
- use physical characteristics of the device to extract information
- can be black-box

Fault attacks

- active attacks - use abnormal conditions causing malfunctions in the system
- triggering of faults is random - device behaves outside specifications
- often it is hard to explain the physical cause of a fault attack
- To be effective you need info about the target operation

This lecture

1. Selecting the target
2. Tools for fault attacks
3. Device characterization
4. Case study: recovery of the PIN for a TREZOR hardware wallet

Selecting the target

A fault attack example

```
1   userauth_passwd(struct ssh *ssh, const char *method)
2   {
3       char *password = NULL;
4       int authenticated = 0, r;
5       u_char change;
6       size_t len = 0;
7
8       if ((r = sshpkt_get_u8(ssh, &change)) != 0 ||
9           (r = sshpkt_get_cstring(ssh, &password, &len)) != 0 ||
10          (change && (r = sshpkt_get_cstring(ssh, NULL, NULL)) != 0) ||
11          (r = sshpkt_get_end(ssh)) != 0) {
12          freezero(password, len);
13          fatal_fr(r, "parse packet");
14      }
15      if (change)
16          logit("password change not supported");
17      else if (PRIVSEP(auth_password(ssh, password)) == 1)
18          authenticated = 1;
19      freezero(password, len);
20      return authenticated;
```

Listing from: https://github.com/openssh/openssh-portable/blob/master/auth2-passwd.c

A fault attack example

password = *store field from ssh structure*

```
1   userauth_passwd(struct ssh *ssh, const char *method)
2   {
3       char *password = NULL;
4       int authenticated = 0, r;
5       u_char change;
6       size_t len = 0;
7
8       if ((r = sshpkt_get_u8(ssh, &change)) != 0 ||
9           (r = sshpkt_get_cstring(ssh, &password, &len)) != 0 ||
10          (change && (r = sshpkt_get_cstring(ssh, NULL, NULL)) != 0) ||
11          (r = sshpkt_get_end(ssh)) != 0) {
12          freezero(password, len);
13          fatal_fr(r, "parse packet");
14      }
15      if (change)
16          logit("password change not supported");
17      else if (PRIVSEP(auth_password(ssh, password)) == 1)
18          authenticated = 1;
19      freezero(password, len);
20      return authenticated;
```

Listing from: https://github.com/openssh/openssh-portable/blob/master/auth2-passwd.c

A fault attack example

authenticated = *determines if* password *is correct*

```
1   userauth_passwd(struct ssh *ssh, const char *method)
2   {
3       char *password = NULL;
4       int authenticated = 0, r;
5       u_char change;
6       size_t len = 0;
7
8       if ((r = sshpkt_get_u8(ssh, &change)) != 0 ||
9           (r = sshpkt_get_cstring(ssh, &password, &len)) != 0 ||
10          (change && (r = sshpkt_get_cstring(ssh, NULL, NULL)) != 0) ||
11          (r = sshpkt_get_end(ssh)) != 0) {
12          freezero(password, len);
13          fatal_fr(r, "parse packet");
14      }
15      if (change)
16          logit("password change not supported");
17      else if (PRIVSEP(auth_password(ssh, password)) == 1)
18          authenticated = 1;
19      freezero(password, len);
20      return authenticated;
```

Listing from: https://github.com/openssh/openssh-portable/blob/master/auth2-passwd.c

A fault attack example

change = *does the user request a change of* password

```
1   userauth_passwd(struct ssh *ssh, const char *method)
2   {
3       char *password = NULL;
4       int authenticated = 0, r;
5       u_char change;
6       size_t len = 0;
7
8       if ((r = sshpkt_get_u8(ssh, &change)) != 0 ||
9           (r = sshpkt_get_cstring(ssh, &password, &len)) != 0 ||
10          (change && (r = sshpkt_get_cstring(ssh, NULL, NULL)) != 0) ||
11          (r = sshpkt_get_end(ssh)) != 0) {
12          freezero(password, len);
13          fatal_fr(r, "parse packet");
14      }
15      if (change)
16          logit("password change not supported");
17      else if (PRIVSEP(auth_password(ssh, password)) == 1)
18          authenticated = 1;
19      freezero(password, len);
20      return authenticated;
```

Listing from: https://github.com/openssh/openssh-portable/blob/master/auth2-passwd.c

A fault attack example

len = *determines the length of of the* password

```
1   userauth_passwd(struct ssh *ssh, const char *method)
2   {
3       char *password = NULL;
4       int authenticated = 0, r;
5       u_char change;
6       size_t len = 0;
7
8       if ((r = sshpkt_get_u8(ssh, &change)) != 0 ||
9           (r = sshpkt_get_cstring(ssh, &password, &len)) != 0 ||
10          (change && (r = sshpkt_get_cstring(ssh, NULL, NULL)) != 0) ||
11          (r = sshpkt_get_end(ssh)) != 0) {
12          freezero(password, len);
13          fatal_fr(r, "parse packet");
14      }
15      if (change)
16          logit("password change not supported");
17      else if (PRIVSEP(auth_password(ssh, password)) == 1)
18          authenticated = 1;
19      freezero(password, len);
20      return authenticated;
```

Listing from: https://github.com/openssh/openssh-portable/blob/master/auth2-passwd.c

A fault attack example

Checks the format of password *and determines its length* len

Deals with the request to change the password

Check if the password *is correct*

```
1   userauth_passwd(struct ssh *ssh, const char *method)
2   {
3       char *password = NULL;
4       int authenticated = 0, r;
5       u_char change;
6       size_t len = 0;
7
8       if ((r = sshpkt_get_u8(ssh, &change)) != 0 ||
9           (r = sshpkt_get_cstring(ssh, &password, &len)) != 0 ||
10          (change && (r = sshpkt_get_cstring(ssh, NULL, NULL)) != 0) ||
11          (r = sshpkt_get_end(ssh)) != 0) {
12          freezero(password, len);
13          fatal_fr(r, "parse packet");
14      }
15      if (change)
16          logit("password change not supported");
17      else if (PRIVSEP(auth_password(ssh, password)) == 1)
18          authenticated = 1;
19      freezero(password, len);
20      return authenticated;
```

Listing from: https://github.com/openssh/openssh-portable/blob/master/auth2-passwd.c

A fault attack example

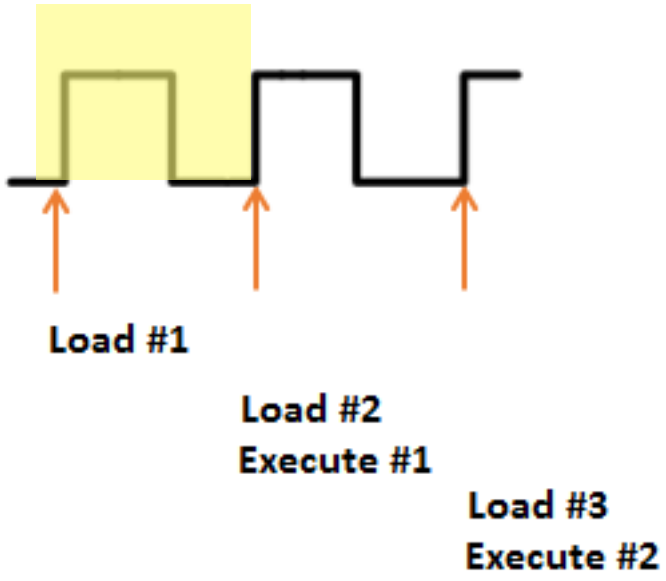
This of three ways in
which you could fault this code.

```
1   userauth_passwd(struct ssh *ssh, const char *method)
2   {
3       char *password = NULL;
4       int authenticated = 0, r;
5       u_char change;
6       size_t len = 0;
7
8       if ((r = sshpkt_get_u8(ssh, &change)) != 0 ||
9           (r = sshpkt_get_cstring(ssh, &password, &len)) != 0 ||
10          (change && (r = sshpkt_get_cstring(ssh, NULL, NULL)) != 0) ||
11          (r = sshpkt_get_end(ssh)) != 0) {
12          freezero(password, len);
13          fatal_fr(r, "parse packet");
14      }
15      if (change)
16          logit("password change not supported");
17      else if (PRIVSEP(auth_password(ssh, password)) == 1)
18          authenticated = 1;
19      freezero(password, len);
20      return authenticated;
```

Listing from: https://github.com/openssh/openssh-portable/blob/master/auth2-passwd.c

Changing the execution flow

Normal clock cycle



Short lock cycle

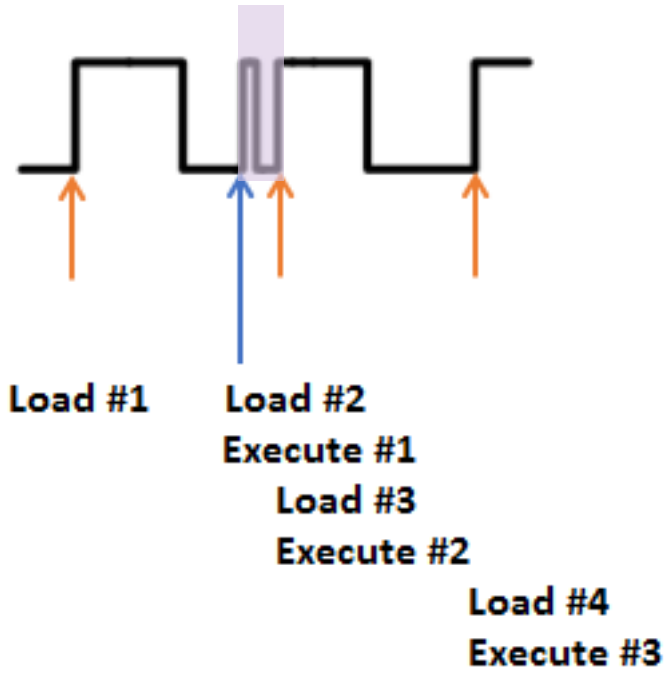
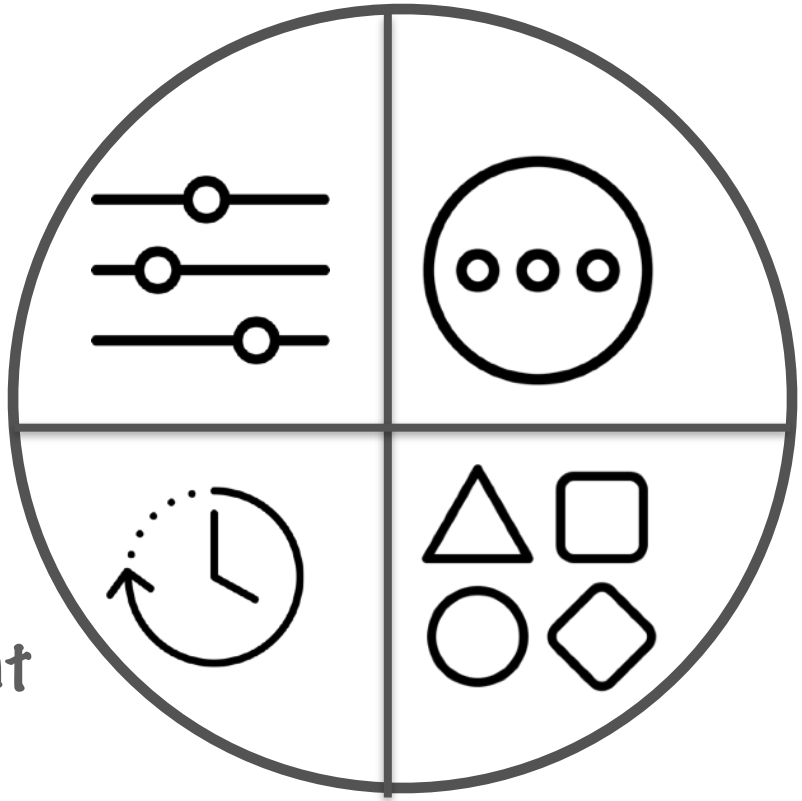


Figure adapted from *Tutorial A2 Introduction to Glitch Attacks*, Colin O'Flynn
https://wiki.newae.comV4:Tutorial_A2_Introduction_to_Glitch_Attacks_(including_Glitch_Explorer)

The fault model

Control

- Precise
- Loose
- None



Granularity

- Bit
- Byte
- Word

Duration

- Permanent
- Transient

Fault type

- Stuck-at (zero or one)
- Flip
- Random

Notes on choosing a suitable target

- Some faults are easier to achieve than other faults
- The more precise the timing needs to be or the more specific the effect, the harder
- One fault vs double fault
- The machine is not executing C code, so looking at assembly code, brings us closer to the machine

Fault attack tools

A typical setup for fault attacks

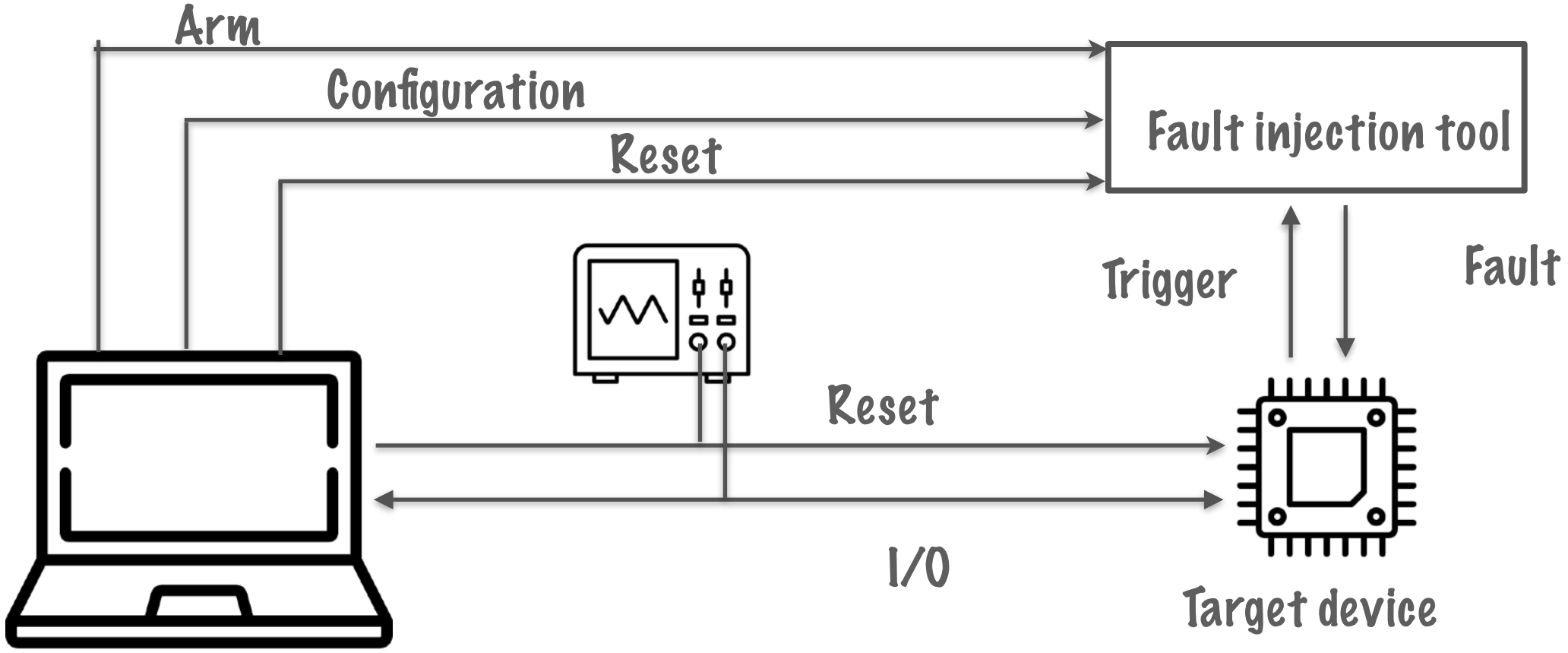
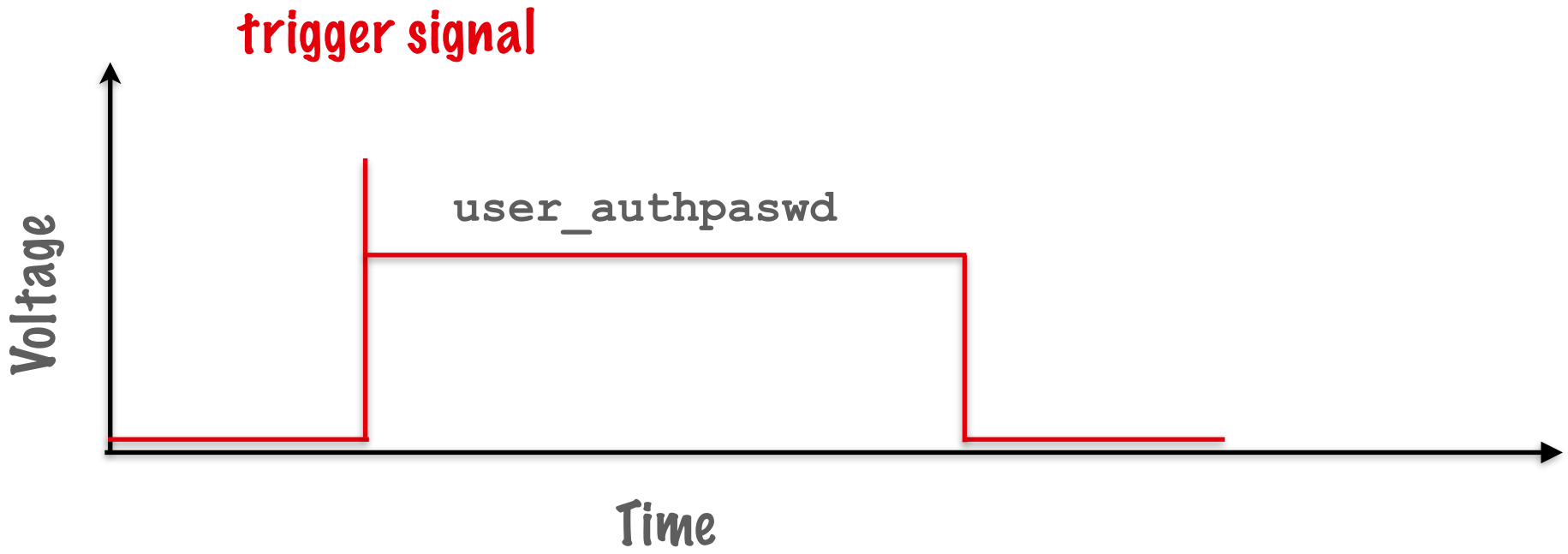


Figure inspired from *The hardware hacking handbook, Breaking Security with Hardware Attacks,* Jasper van Woundenberg Colin O'Flynn

The trigger signal



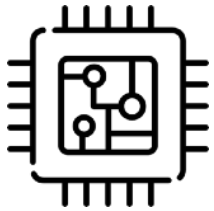
Fault attack methods

Voltage glitching

Clock glitching

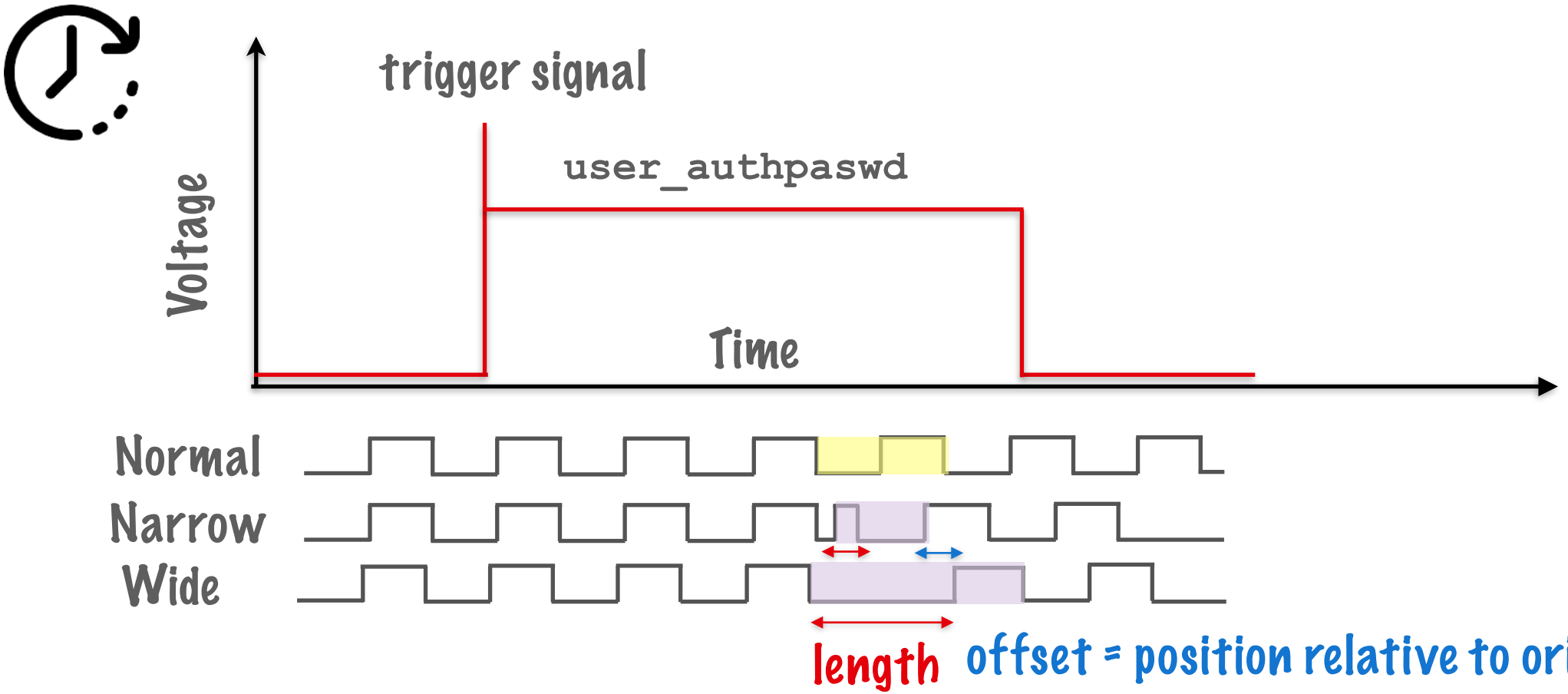
Electromagnetic fault injection

Laser fault injection



Body Biasing Injection

Clock glitching



Fault attack methods

Voltage glitching

Clock glitching

External clock line

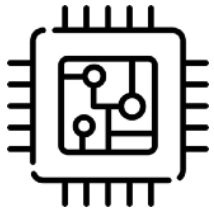
Simple to execute

Noninvasive

Equipment is affordable

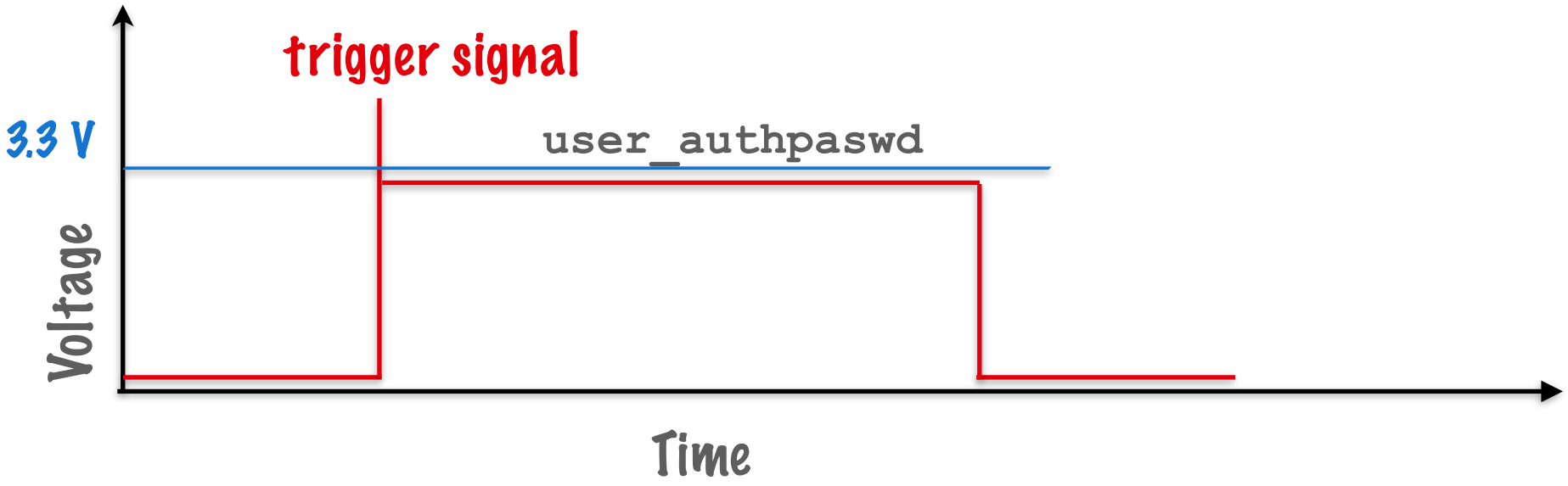
EM fault injection

Laser fault injection

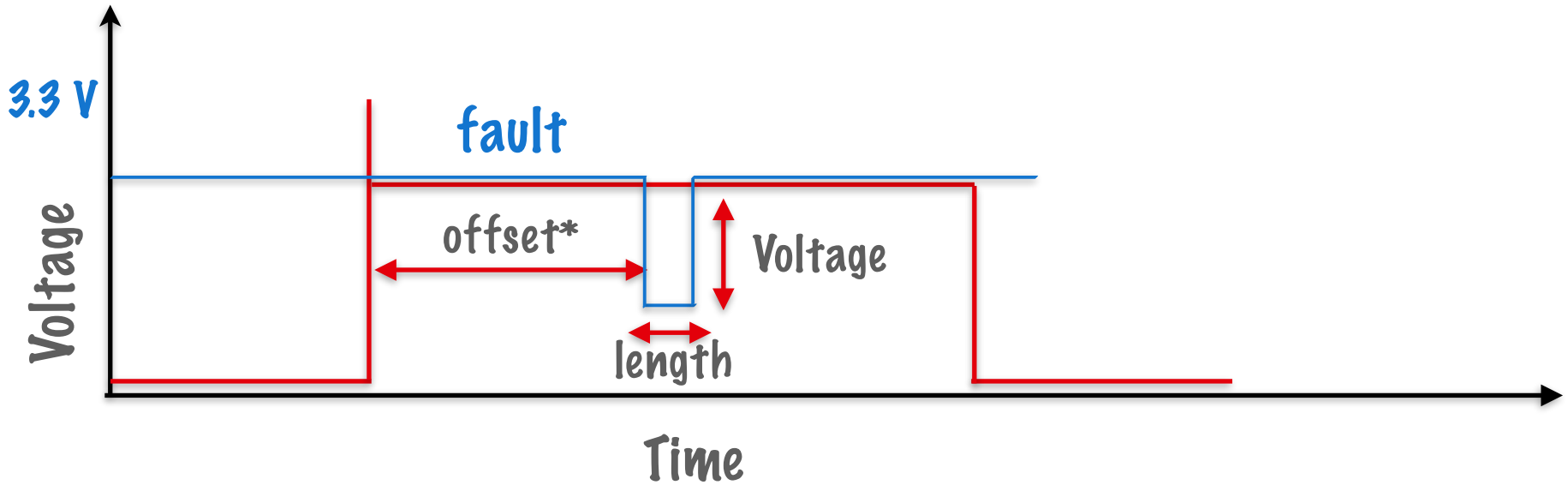


Body Biasing Injection

Voltage glitching

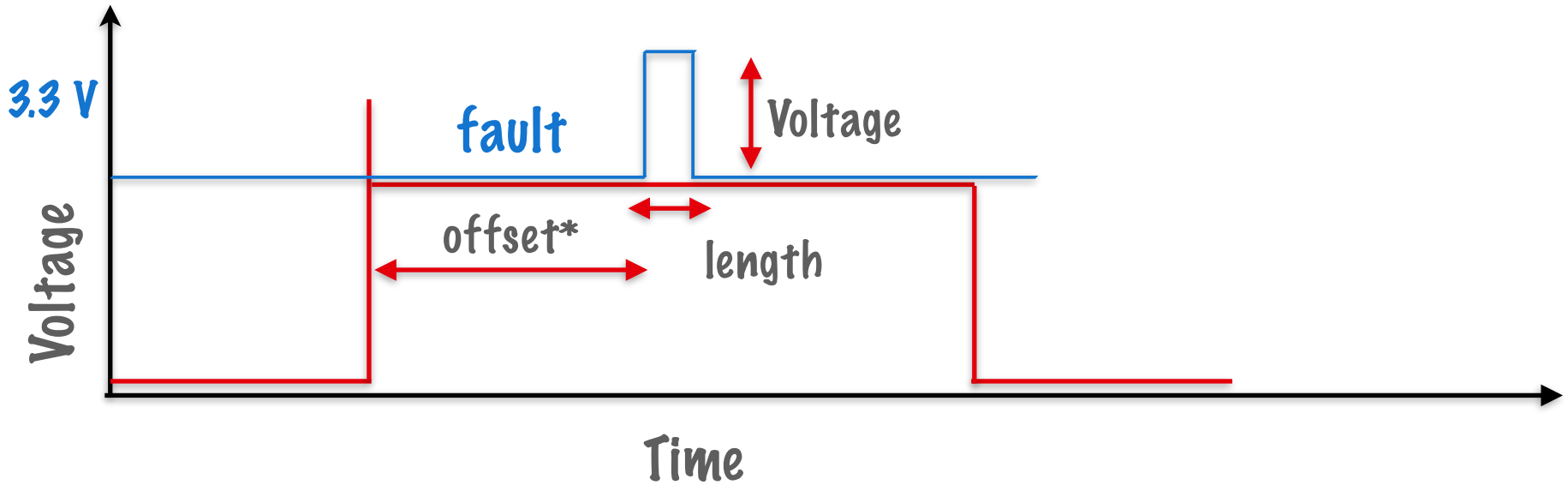


Voltage glitching



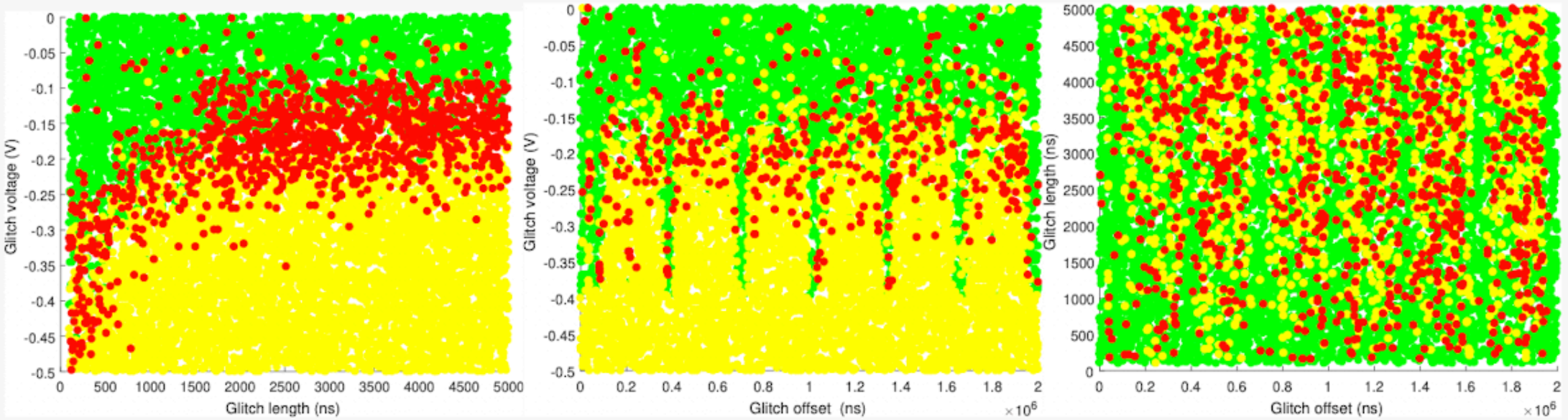
* sometimes also called delay

Voltage glitching



* sometimes also called delay

Searching for effective faults



Voltage fault injection results, Normal (green), Inconclusive (yellow), Successful (red).

Niels Samwel, Lejla Batina: Practical Fault Injection on Deterministic Signatures: The Case of EdDSA. AFRICACRYPT 2018: 306-321

Fault attack methods

Remove capacitors

Simple to execute
Equipment is affordable

Voltage glitching

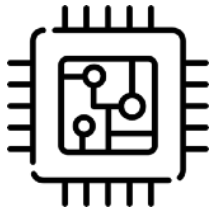
Clock glitching

External clock line

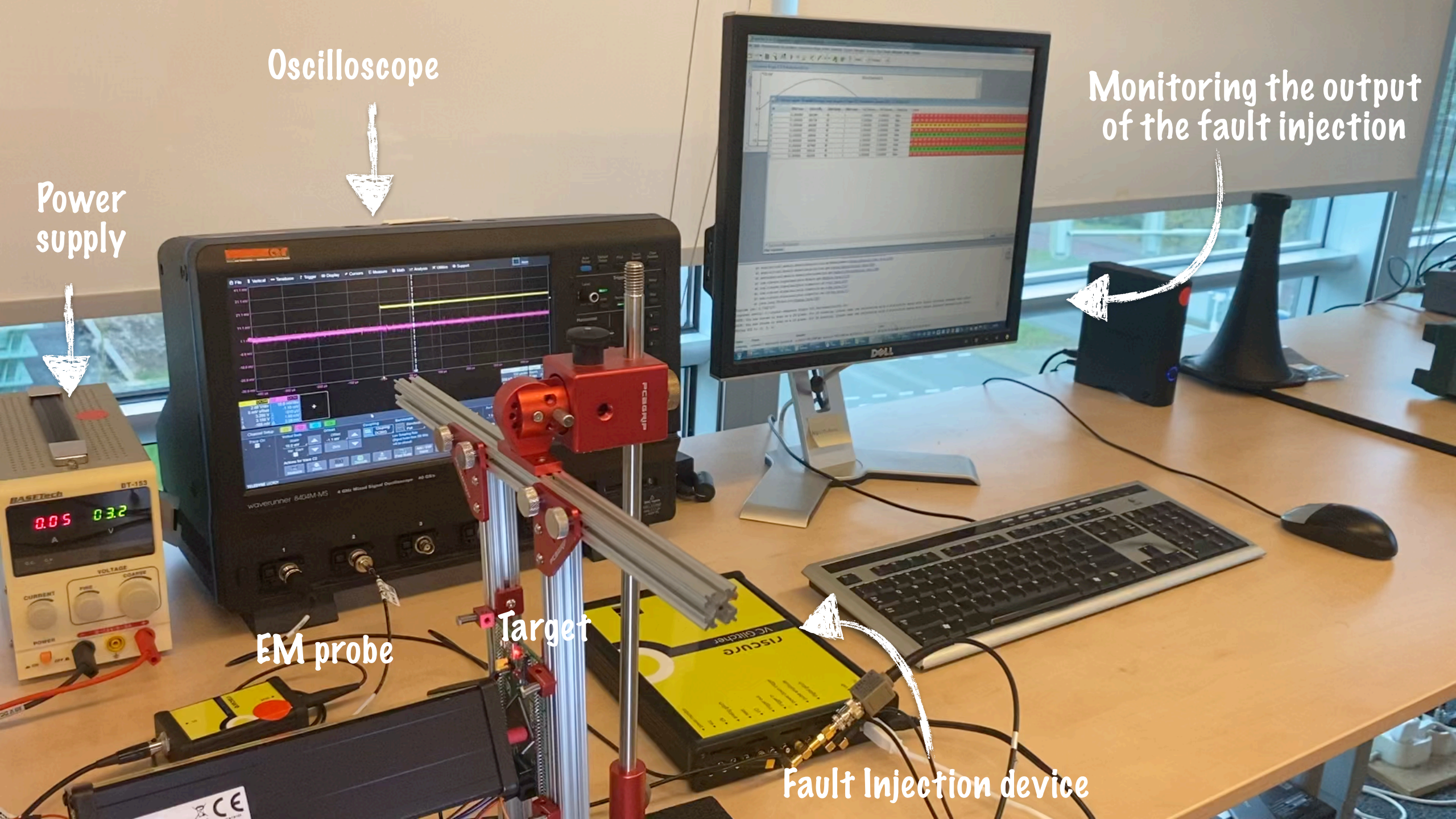
Simple to execute
Noninvasive
Equipment is affordable

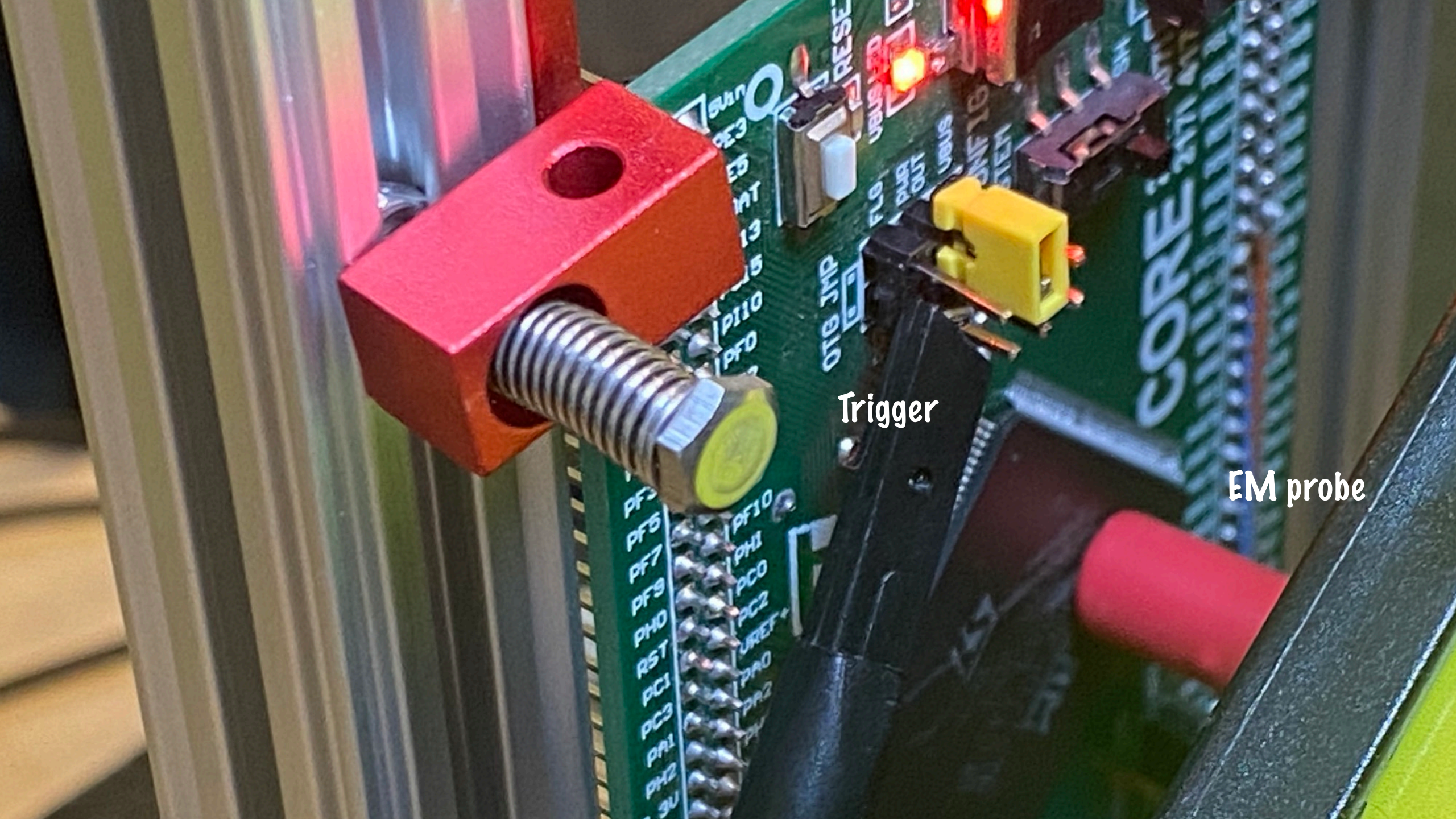
EM fault injection

Laser fault injection

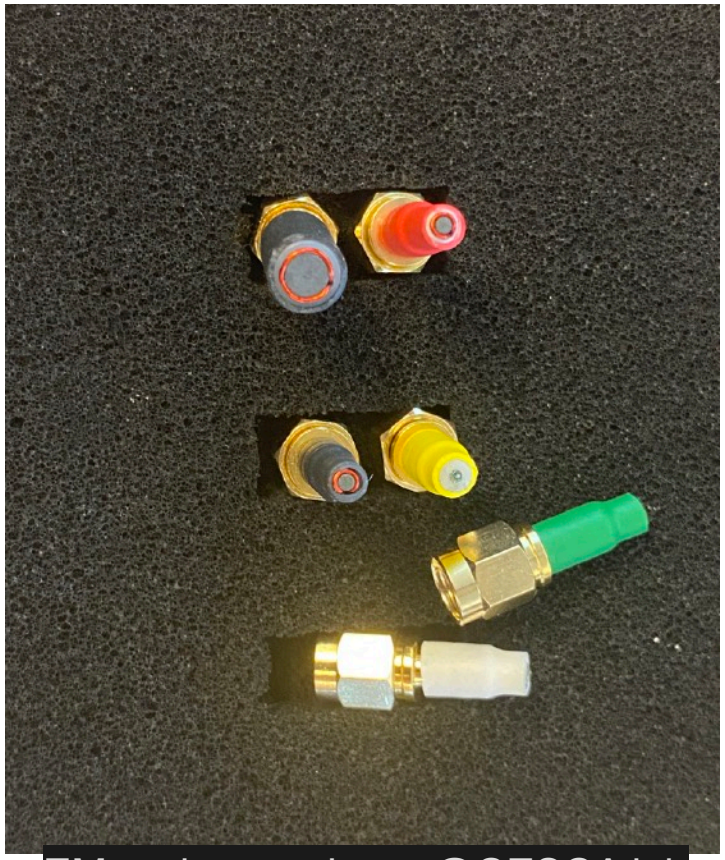


Body Biasing Injection


Oscilloscope
Power supply
Monitoring the output of the fault injection
EM probe
Target
Fault Injection device


Trigger
EM probe

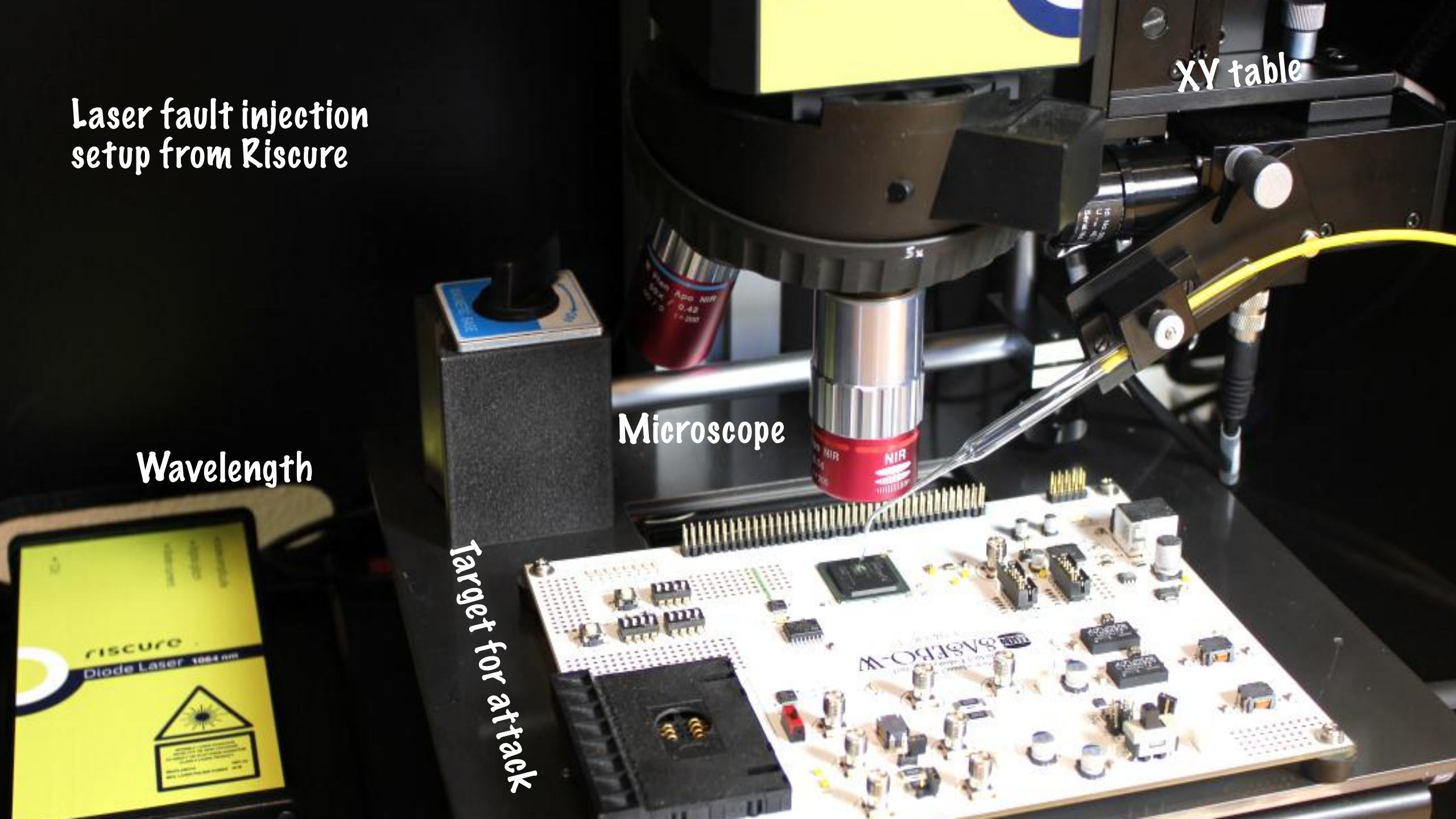
Probes for fault attacks

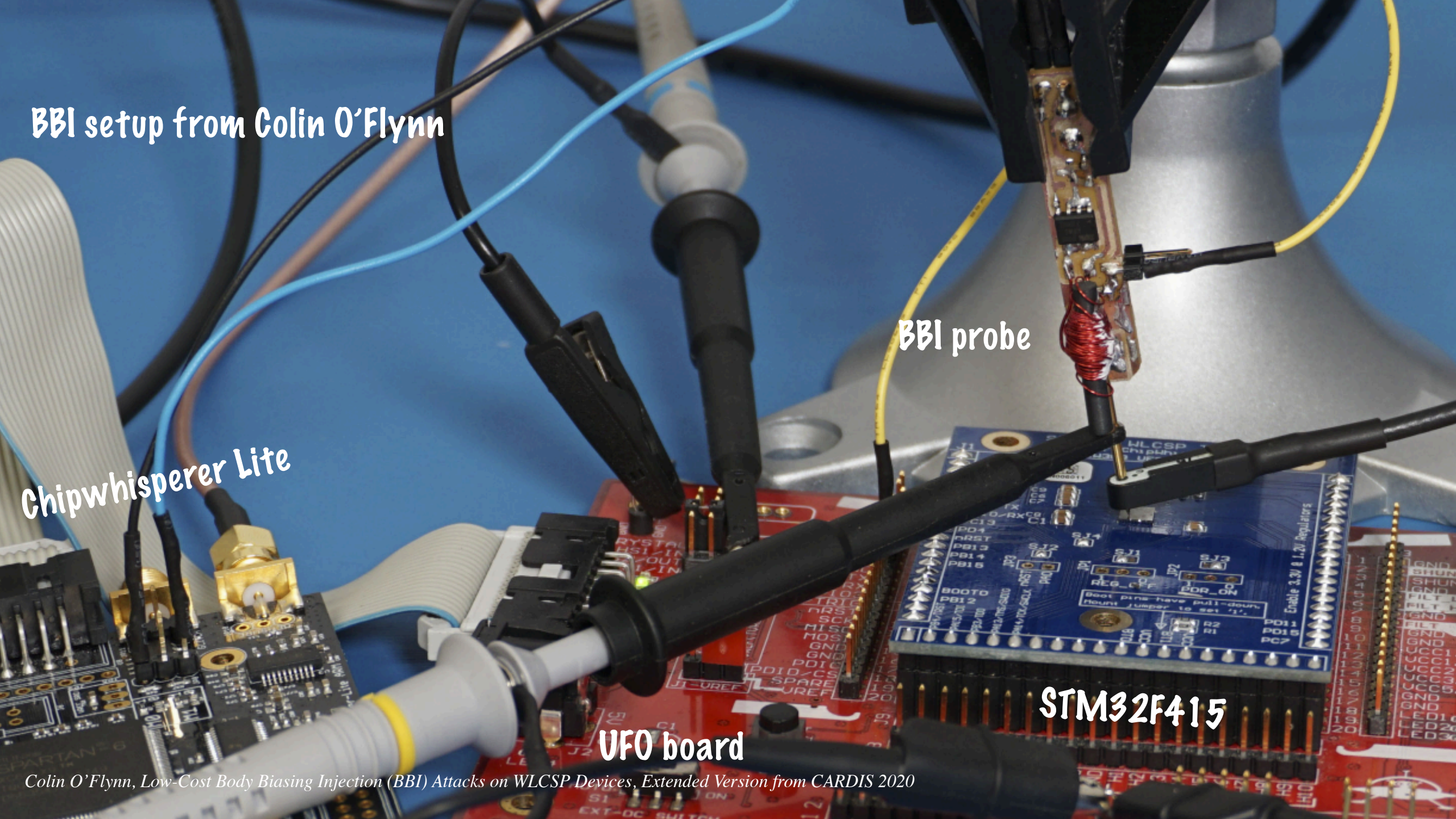


EM probes we have @CESCA lab

PicoEMP manufactured @CESCA lab

https://hackaday.com/2022/01/15/glitch-your-way-to-reverse-engineering-glory-with-the-picoemp/#more-515590


Laser fault injection
setup from Riscure
XY table
Microscope
Wavelength
Target for attack
riscure
Diode Laser 1064 nm



Colin O'Flynn, Low-Cost Body Biasing Injection (BBI) Attacks on WLCSP Devices, Extended Version from CARDIS 2020

Fault attack methods

Remove capacitors

Equipment is affordable

Simple to execute

Voltage glitching

Clock glitching

External clock line

Simple to execute

Noninvasive

Equipment is affordable

Large search space

Equipment is affordable

Simple to execute

EM fault injection

Laser fault injection

Training to operate lasers

Large parameter space

Expensive

Decap

Double faults possible

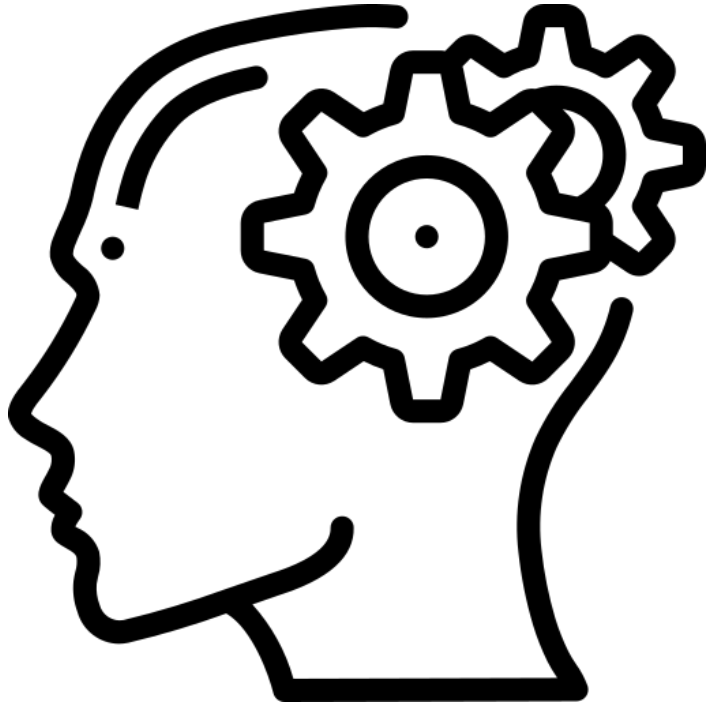
Very effective

Device characterization

Take a few minutes

What is the purpose of target characterization?

How would you characterize a target?



Target characterization

Set trigger high

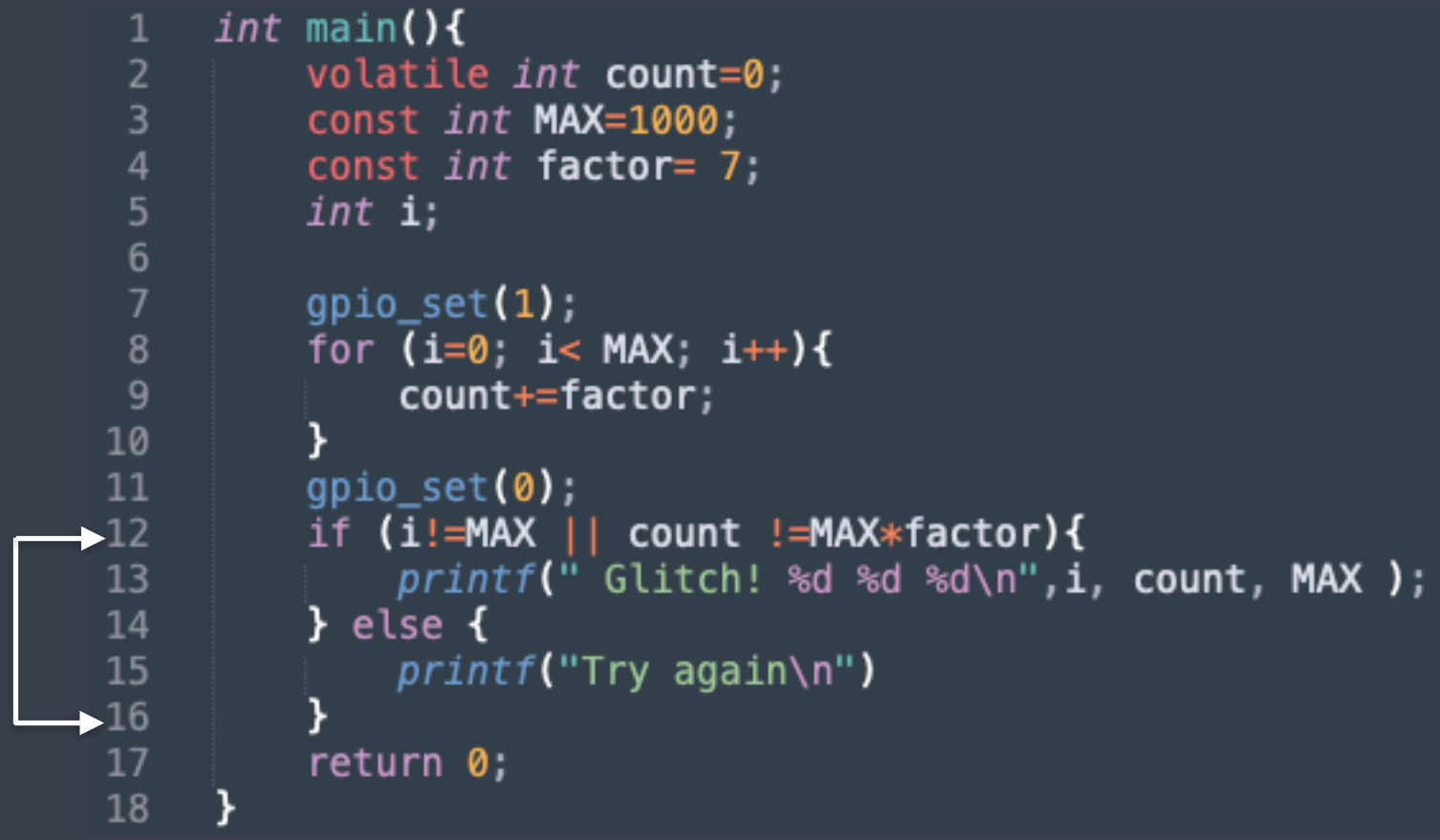
Set trigger low

```
1   int main(){
2       volatile int count=0;
3       const int MAX=1000;
4       const int factor= 7;
5       int i;
6
7       gpio_set(1);
8       for (i=0; i< MAX; i++){
9           count+=factor;
10      }
11      gpio_set(0);
12      if (i!=MAX || count !=MAX*factor){
13          printf(" Glitch! %d %d %d\n",i, count, MAX );
14      } else {
15          printf("Try again\n")
16      }
17      return 0;
18  }
```

Code from *The hardware hacking handbook, Breaking Security with Hardware Attacks,* Jasper van Woundenberg Colin O'Flynn

Target characterization

Checking the result of glitching

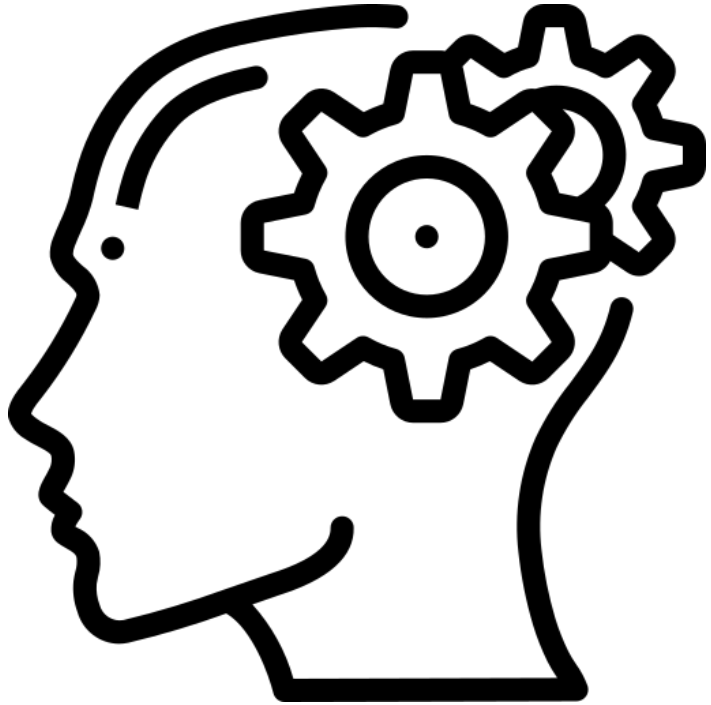


```
1   int main(){
2       volatile int count=0;
3       const int MAX=1000;
4       const int factor= 7;
5       int i;
6
7       gpio_set(1);
8       for (i=0; i< MAX; i++){
9           count+=factor;
10      }
11      gpio_set(0);
12      if (i!=MAX || count !=MAX*factor){
13          printf(" Glitch! %d %d %d\n",i, count, MAX );
14      } else {
15          printf("Try again\n")
16      }
17      return 0;
18  }
```

Code from *The hardware hacking handbook, Breaking Security with Hardware Attacks,* Jasper van Woundenberg Colin O'Flynn

Take a few minutes

Can you think of other tests?



Case study: recovery of the PIN for a TREZOR hardware wallet


TREZOR
https://www.youtube.com/watch?v=dT9y-KQbqi4

Crypto wallets

Private key 

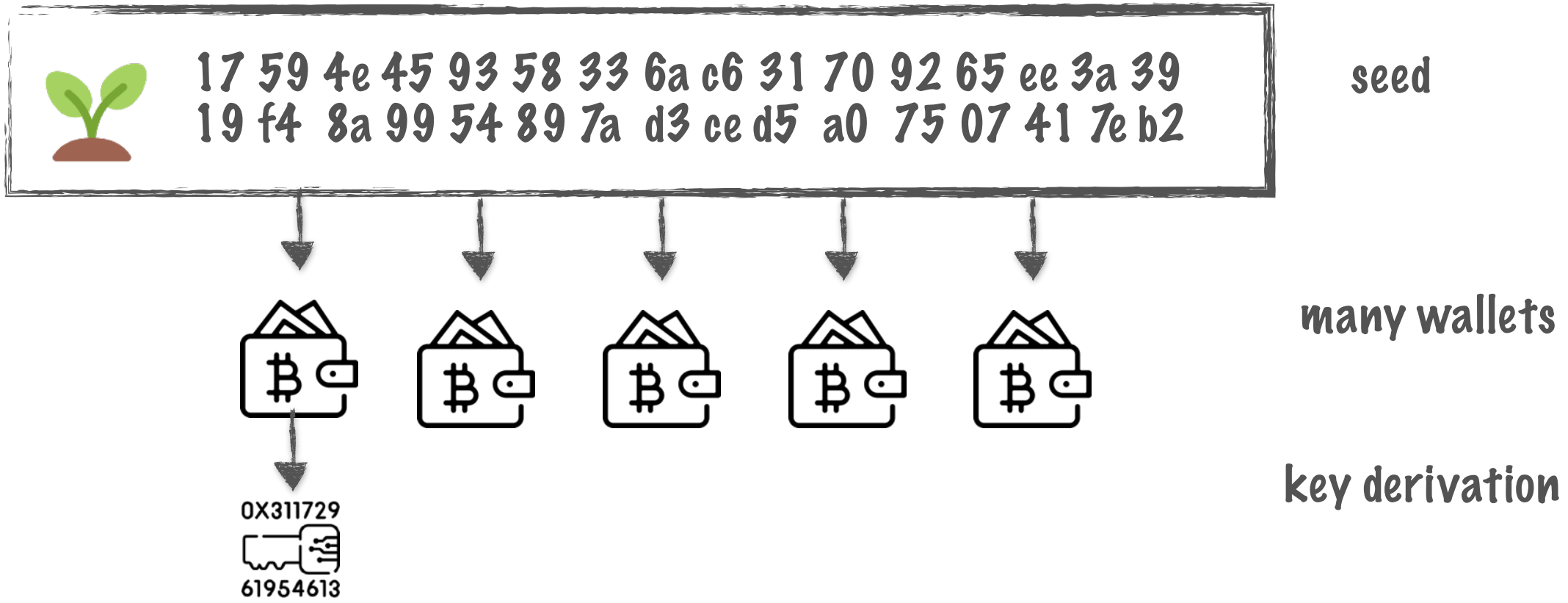
Public key

Used to send coins

Used to receive coins

You need a new key pair for each cryptocurrency/wallet

BIP32/39 and BIP44

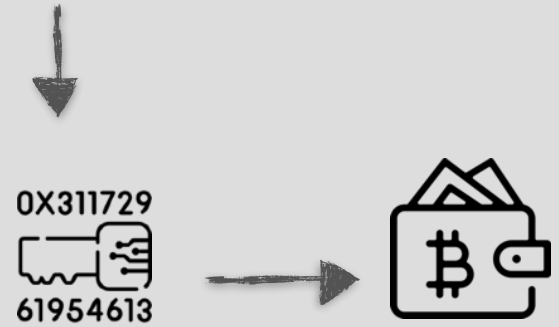


1 2 3 4

Source: 35C3 - wallet.fail.
https://www.youtube.com/watch?v=Y1OBIGslgGM

Hardware Crypto wallets

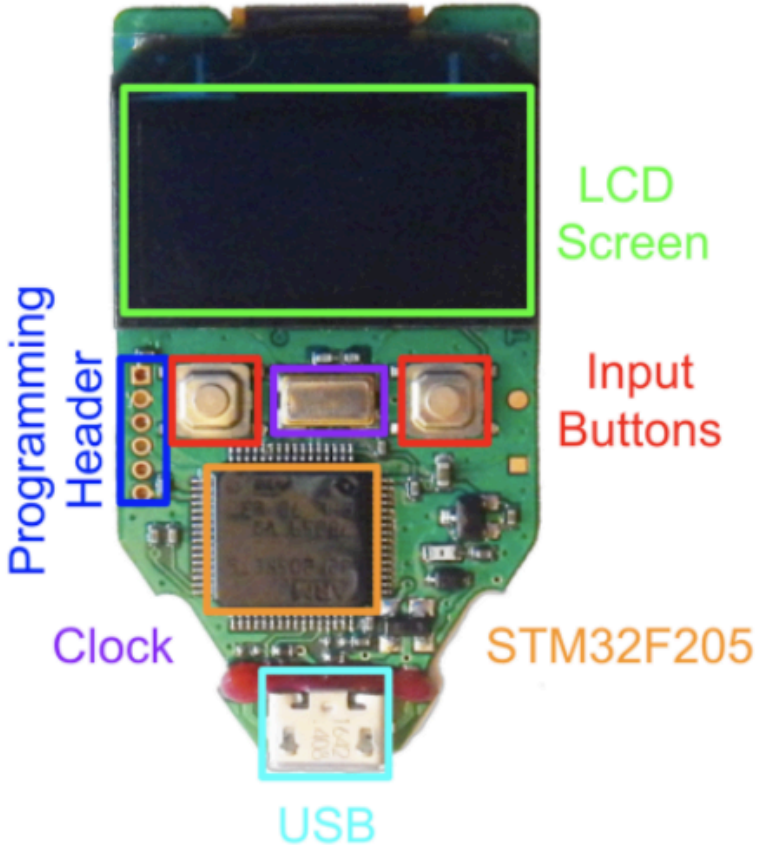
17 59 4e 45 93 58 33 6a c6 31 70 92 65 ee 3a 39
19 f4 8a 99 54 89 7a d3 ce d5 a0 75 07 41 7e b2



1 2 3 4

Trezor One

Source: https://blog.gridplus.io/hardware-wallet-vulnerabilities-f20688361b88

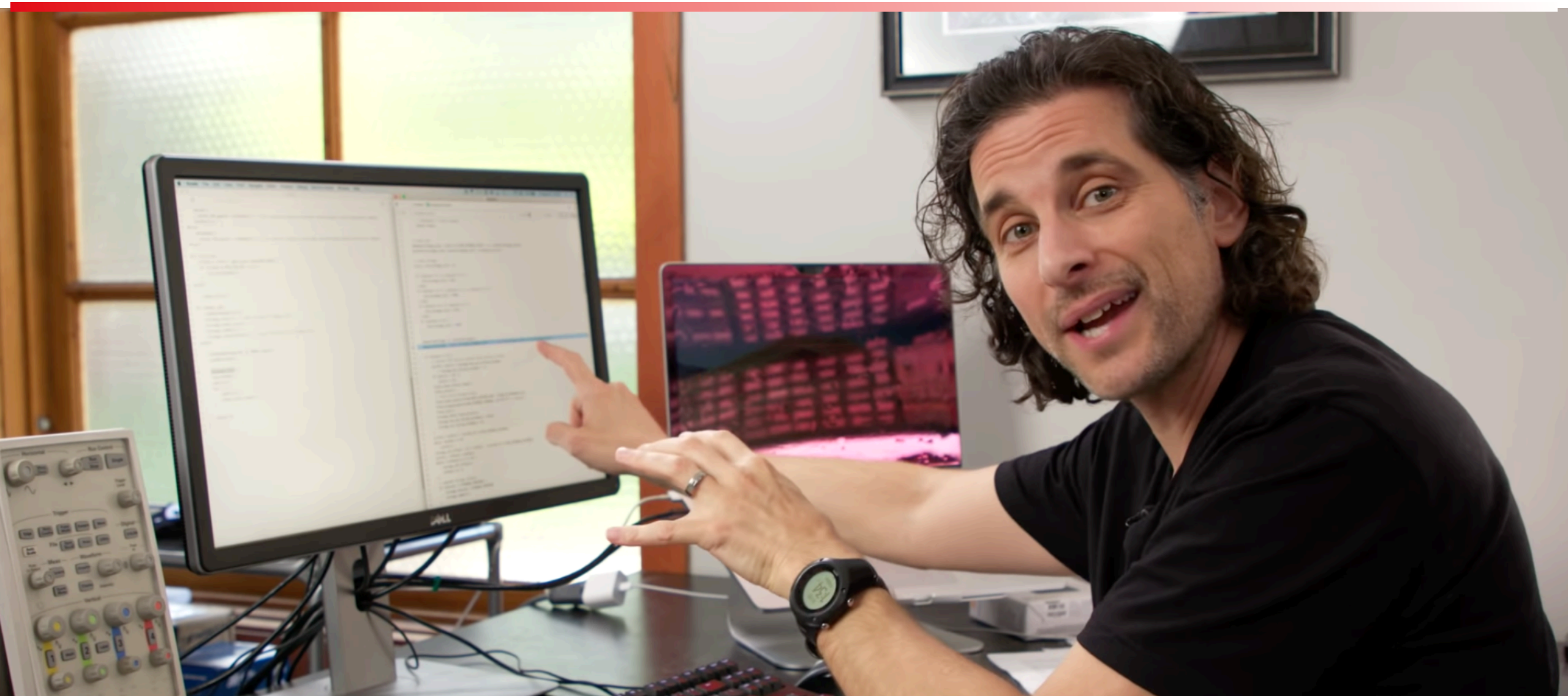


Security features of the ARM Cortex-M3

RDP	Value	Behavior
RDP0	0xAA	Full Access
RDP1	NOT 0xAA or 0xCC	Ability to read RAM + Regs, no flash access, no single stepping
RDP2	0xCC	No access

WALLET.FAIL

Selecting the target for the attack

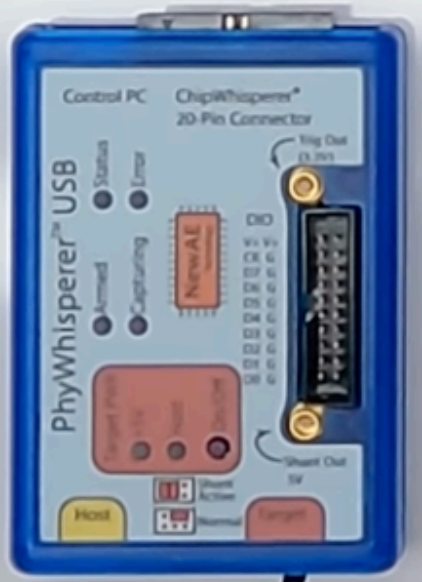


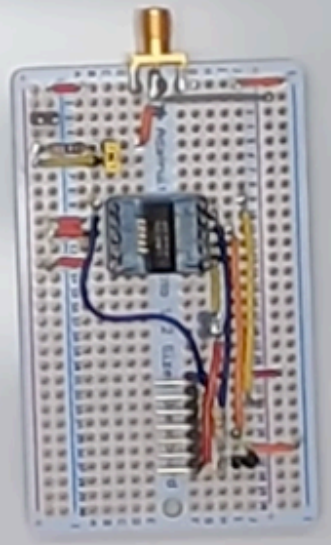
Selecting the target for the attack

```
145
146     // copy storage
147     size_t old_storage_size = 0;
148
149     if (version == 1 || version == 2) {
150         old_storage_size = 460;
151     } else
152     if (version == 3 || version == 4 || version == 5) {
153         old_storage_size = 1488;
154     } else
155     if (version == 6 || version == 7) {
156         old_storage_size = 1496;
157     } else
158     if (version == 8) {
159         old_storage_size = 1504;
160     }
161
162     memset(&storage, 0, sizeof(Storage));
163     memcpy(&storage, (void *)(FLASH_STORAGE_START + 4 + sizeof(storage_uuid)), old_storage_size);
164
165     if (version <= 5) {
166         // convert PIN failure counter from version 5 format
167         uint32_t pinctr = storage.has_pin_failed_attempts
168             ? storage.pin_failed_attempts : 0;
169         if (pinctr > 31) {
170             pinctr = 31;
171         flash_clear_status_flags();
172         flash_unlock();
173         // erase extra storage sector
174         flash_erase_sector(FLASH_META_SECTOR_LAST, FLASH_CR_PROGRAM_X32);
175         flash_program_word(FLASH_STORAGE_PINAREA, 0xffffffff << pinctr);
176         flash_lock();
177         storage_check_flash_errors();
178         storage.has_pin_failed_attempts = false;
179         storage.pin_failed_attempts = 0;
```

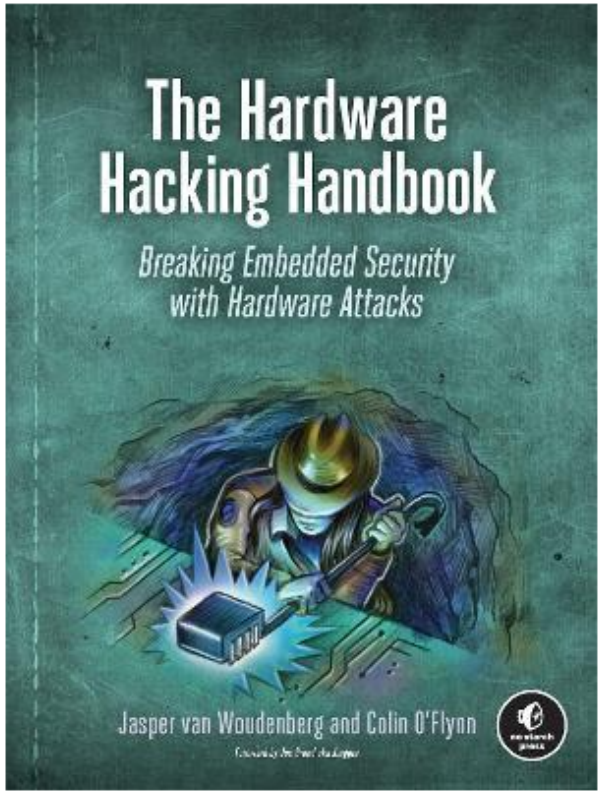
only in < v1.6.0

12 weeks
target selection
+
device characterization





X Marks the spot



Detailed description of an attack on Trezor wallet

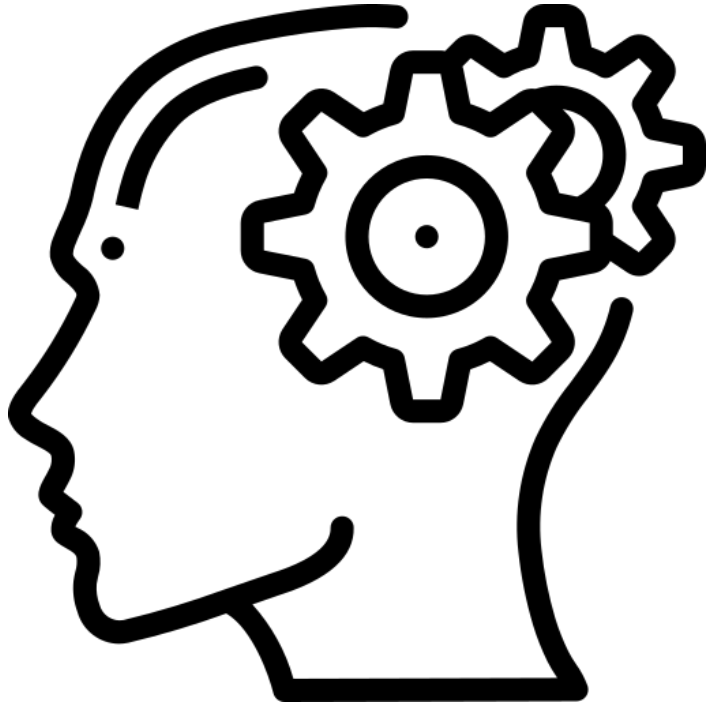
EM in place of voltage glitching

Interesting practise target

1 2 3 4

Countermeasures

Can you think how to protect your target against FI attacks?



Conclusions

Fault attacks are powerful and practical

It can be harder to protect against FI than SCA

Attacks are typically tailored to one use case

Sophisticated attacks to attack cryptographic implementations

Combination attacks exist which use both SCA and FI