Scaling up fault injection simulation campaigns

Ambre Iooss

01/10/2025

Synacktiv

SYNACKTIV

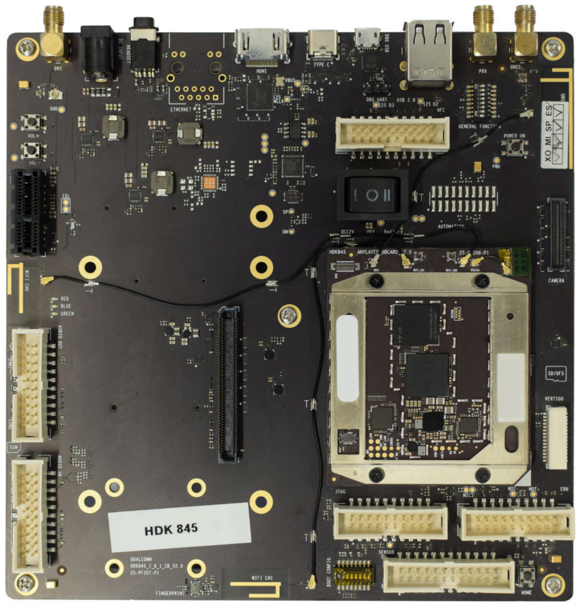
- French offensive security company
- 180+ security experts
- 5 departments:
  - Pentest / Redteam
  - RE / VR
  - Development
  - Incident Response
  - Revel.io

Introduction

Goal: Running unsigned code on a smartphone with configured secure boot. Target the boot ROM.

Qualcomm SDM845 System-on-Chip, 2018

Boot ROM dumped using devboard JTAG.



Problems:

- Modern SoC
- Large boot ROM (~200 kB)
- Package-on-Package: DRAM stacked on SoC

Prior work on older SoC

SYNACKTIV

Hector Marco, BlackHat Europe 2022

Side-channel divergence and power glitching: secure boot bypass on Qualcomm **MSM8916** (2014).

Planning

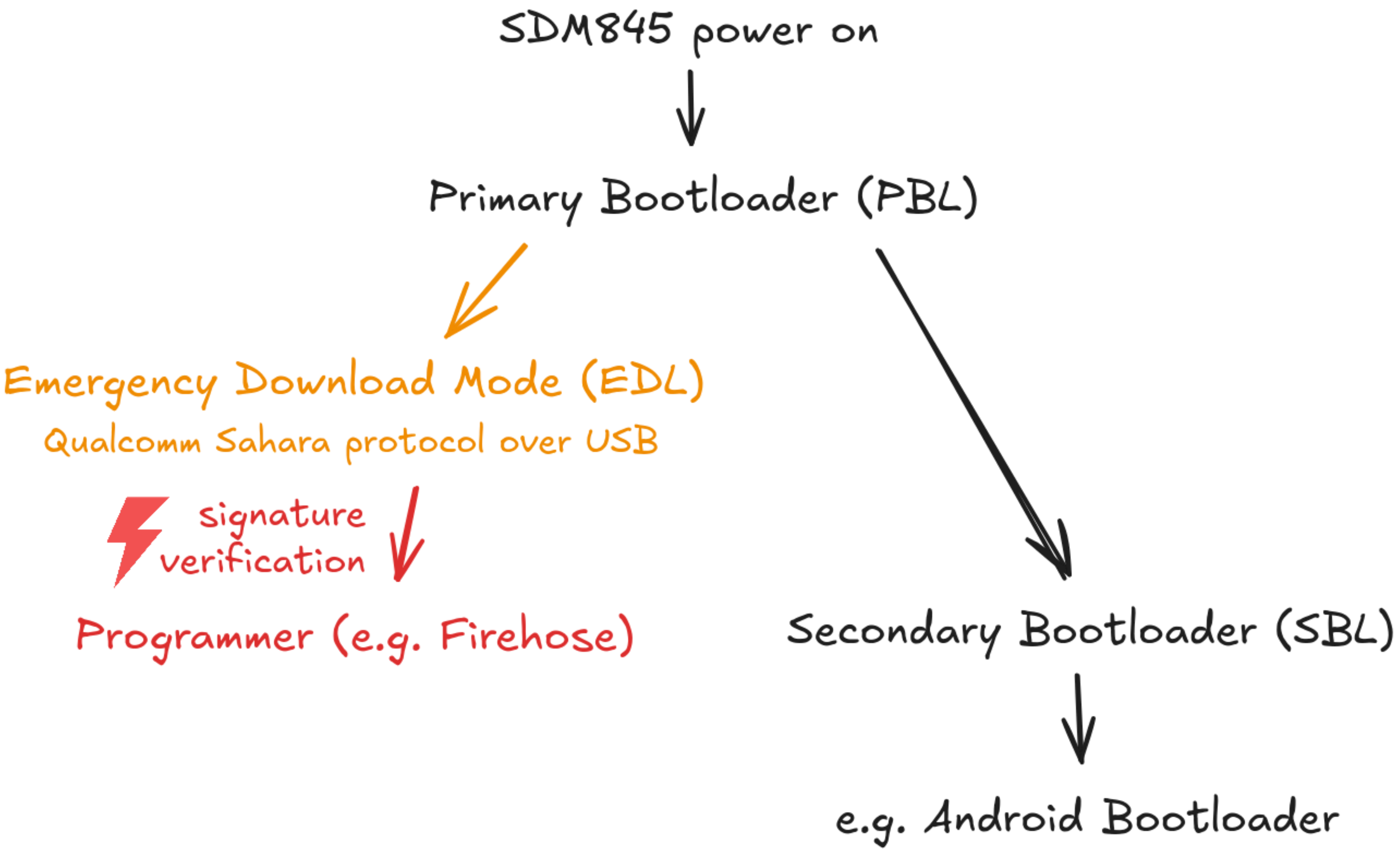
SYNACKTIV

1. Fault injection **simulation** (with optimisations) on SDM845 boot ROM
2. **Simulation to reality** mapping
3. **Real-world** fault injection campaign and failures

Fault injection simulation

and optimisations

Fault injection target



Fault injection target

Same scenario as MSM8916 BlackHat talk:

- Programmer is a ELF file with a segment containing a certificate chain.
- Signature verifications chain:
  1. Fused public key
  2. Root CA certificate
  3. Attestation CA certificate
  4. Loader certificate
  5. Hash table signature

Target: bypass Root CA certificate signature verification.

Fault injection simulation

Python tool based on Unicorn-Engine: https://github.com/Ledger-Donjon/rainbow/

```
emu = rainbow_aarch64()
emu.load("bootrom.elf")
emu.hook_bypass("generate_random", lambda e: e[e["x0"]] = random.randbytes(e["x1"]))
emu.start(0x08000000, 0, count=1000)
fault_skip(emu)  # inject fault after 1000 instructions
emu.start(emu["pc"], 0)
```

Alternative with QEMU TCG: https://github.com/erdnaxe/qemu-fault-plugins

```
qemu-system-arm -machine netduinoplus2 -nographic -d plugin \
    -drive if=none,format=qcow2,file=snapshot.qcow2 -loadvm snapshot \
    -plugin libstoptrigger.so,addr=0x08001235,addr=0x08004019:129,icount=4000000:130 \
    -plugin libskipinsn.so,icount=1000
```

Simulation optimisation #1

Lazy: start simulation from reset address

Consequence: simulation ETA is >3 years

First low-hanging fruits: cut "bad-code" flows early by replacing them with `BRK #0`.

Some bad code paths to patch:

- `BL #0` instructions (can be automatically replaced)
- USB error handlers

Simulation optimisation #2

Observations:

- Counting instructions can be slow (GDB protocol, QEMU TCG scoreboard)
- Breakpoints on addresses is more robust (survive small hooking changes)
- Emulation must be deterministic for a fault injection campaign to make sense

Solution: record an execution trace and use it as a reference for fault campaigns

Execution trace contains list of:

```
struct basic_block_info {
  uint64_t address;
  uint32_t current_cpu;
  uint32_t instructions_count;
  uint8_t  instructions_size[instructions_count];
};
```

This trace can be processed to find blocks executed only once, and fault them first.

Fault by `(address, execution_count)`

Simulation results

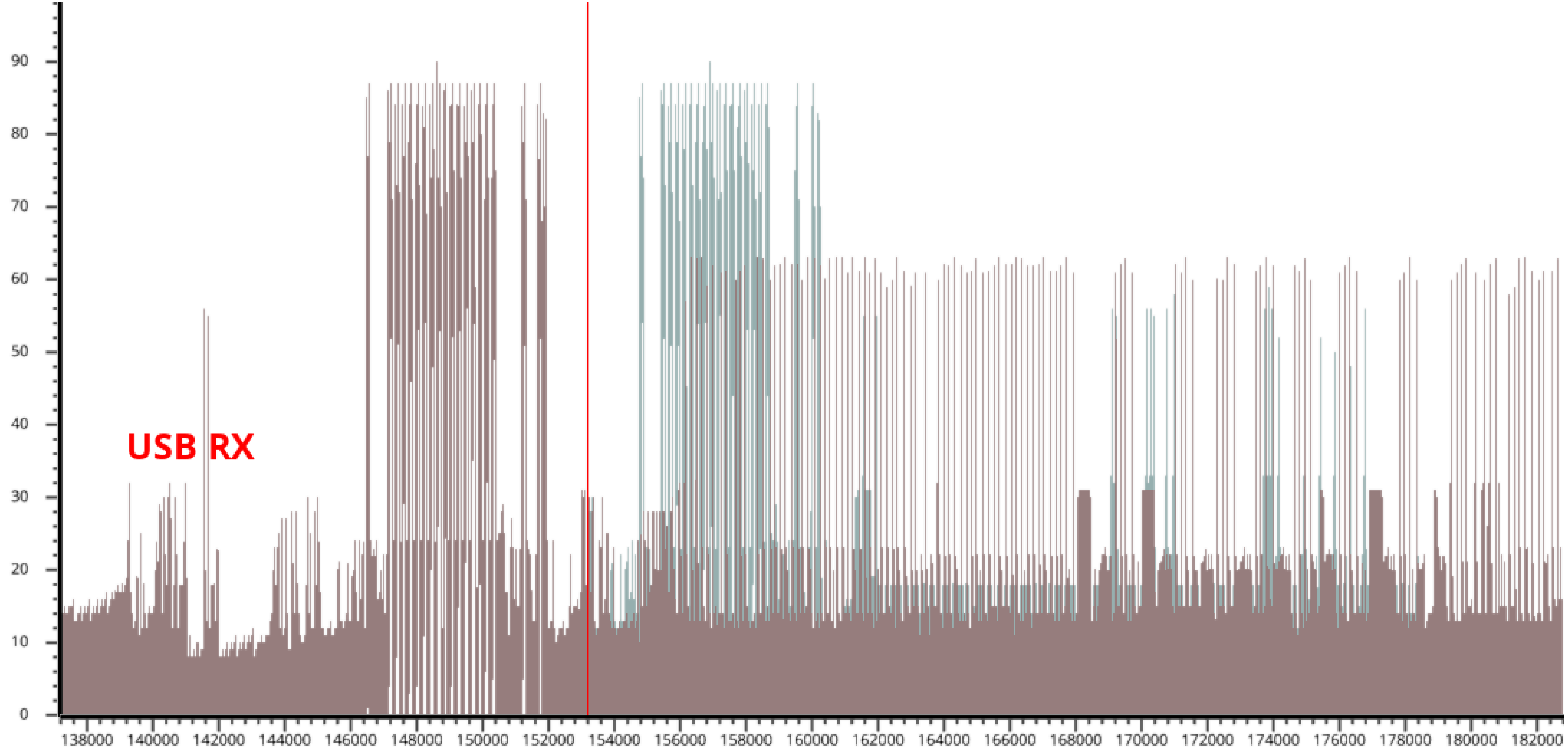
SYNACKTIV

Fault model	✓ Bypass	⚠ Error detected	⚡ Crash
Single instruction skip	0x0031E688:0 0x0031F060:1 **0x0031F064:1** 0x0031F08C:1 ... (25 total)	420+1957 total	17624+ total
Stuck destination register at 0x00000000	0x0031F060:1 **0x0031F064:1**	23+277 total	5861+ total
Stuck destination register at 0xFFFFFFFF	**0x0031F064:1**	5+142 total	14484+ total

Target: second execution at address `0x0031F064`

Static analysis: `0x0031F064` seems related to reading fuses

Side-channel simulation



Hamming Weight of the destination register, relative to the instruction count

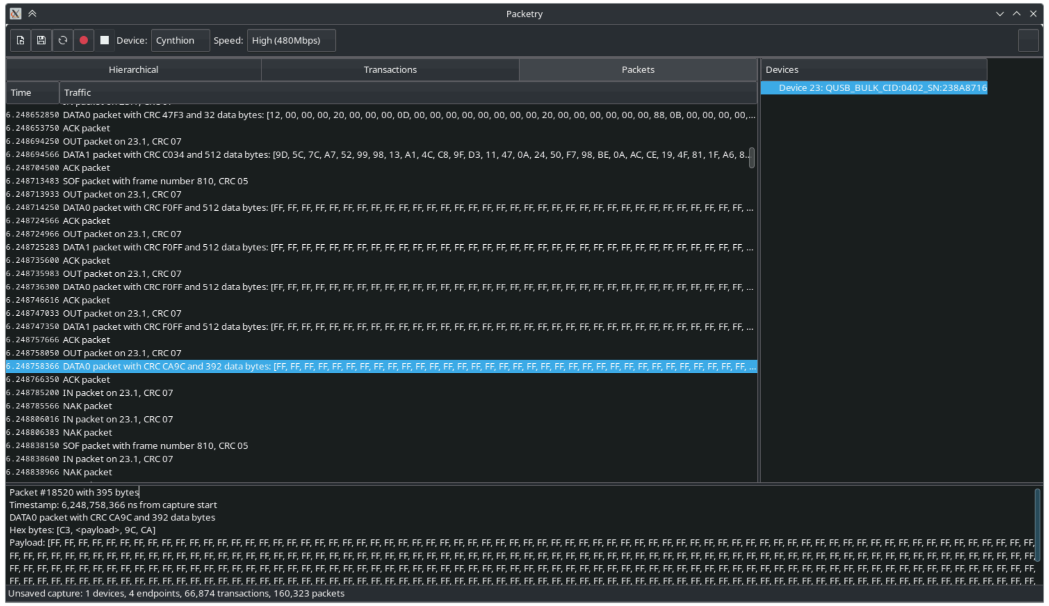
Simulation to reality mapping

USB triggering

Problem: need a hardware trigger as a reference for faults injection

Simulation conclusion: fault after the last packet of the loader hash table segment

Solution: modify Cynthion USB analyser to raise a trigger on the USB packet



Side-Channel Analysis

SYNACKTIV

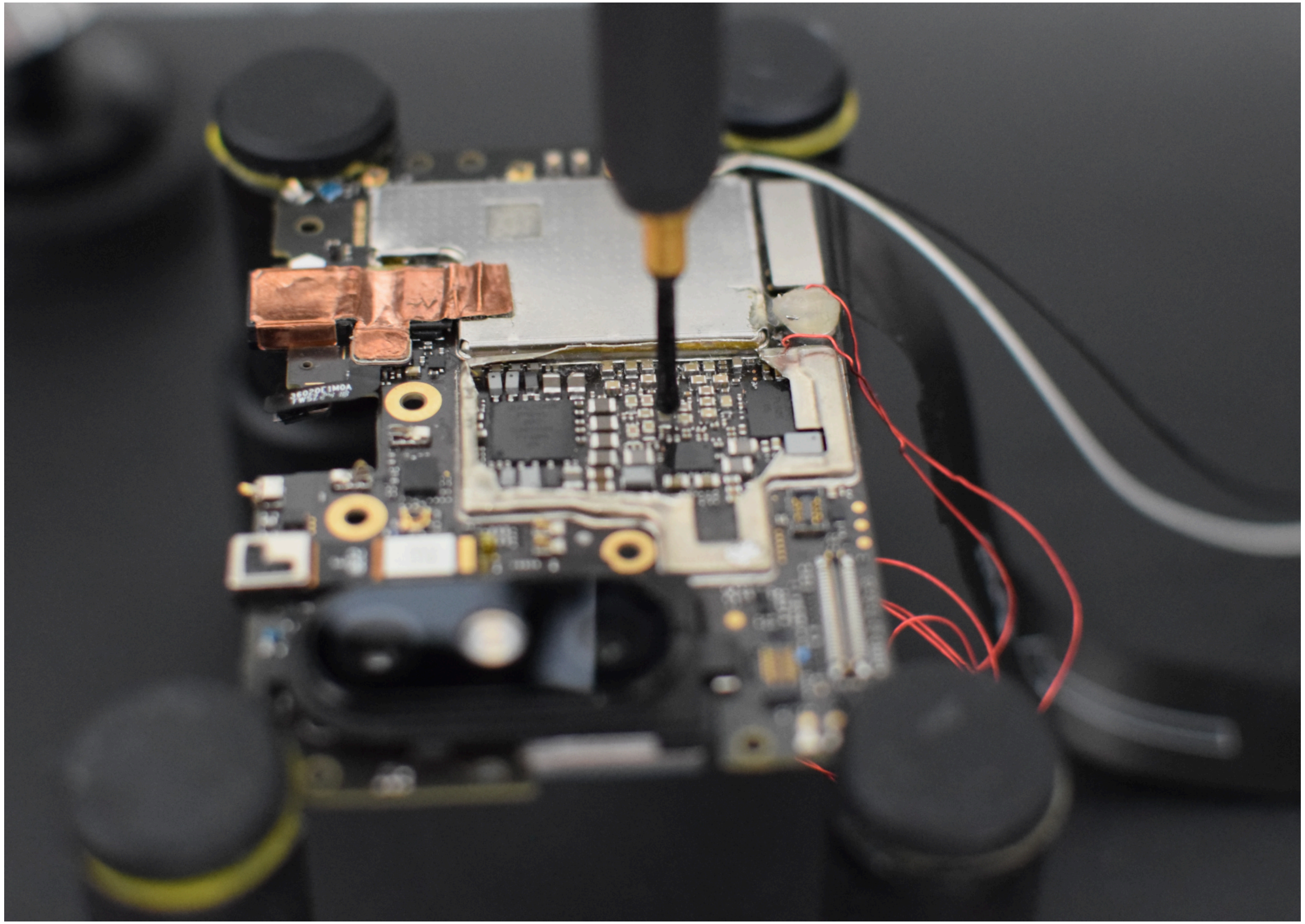
Problem: need side-channel traces to match fault simulation patterns

Smartphone: Xiaomi Mi 8, PCB boardview, schematics and EDL loader are leaked online

Power rails identification (behind SoC):

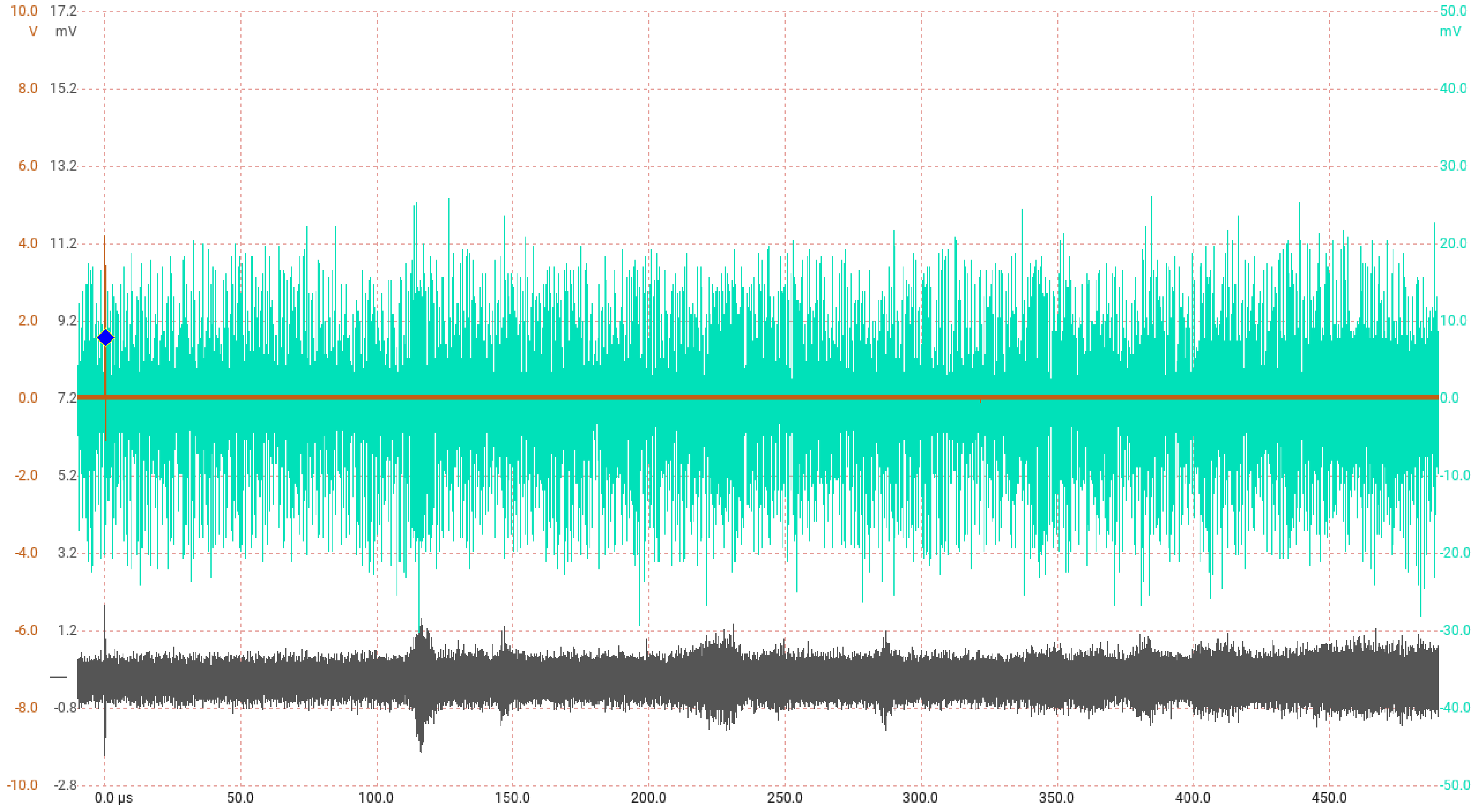
- PM845 - Power Management IC for SDM845
- **VDD_APC0** - Application Processor Core 0 power domain
- VDD_APC1 - Application Processor Core 1 power domain, **off during boot**
- VDD_CX - Digital power domain directly supplied by Core crystal oscillator (CXO)
- VDD_MX - Memory power domain

Side-Channel Analysis



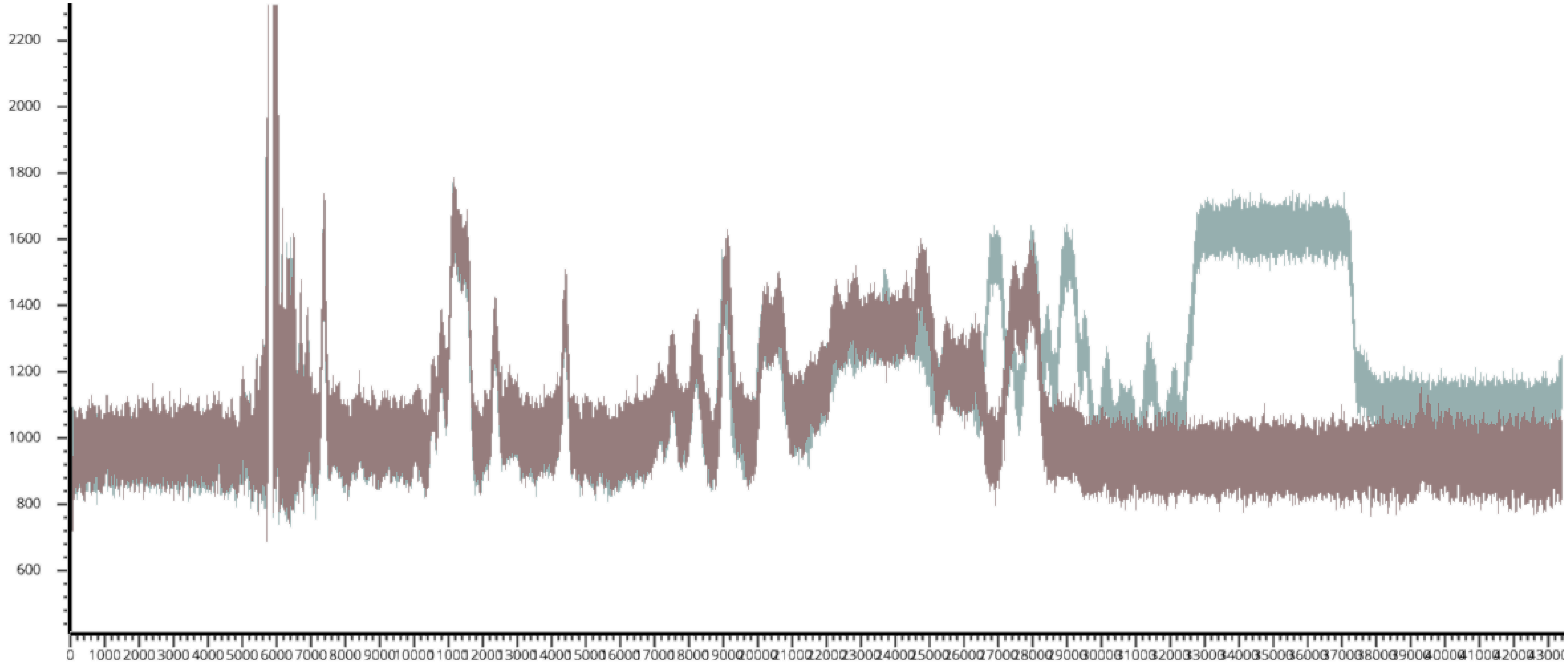
Side-Channel Analysis

Some patterns emerge from averaging without any post-processing!

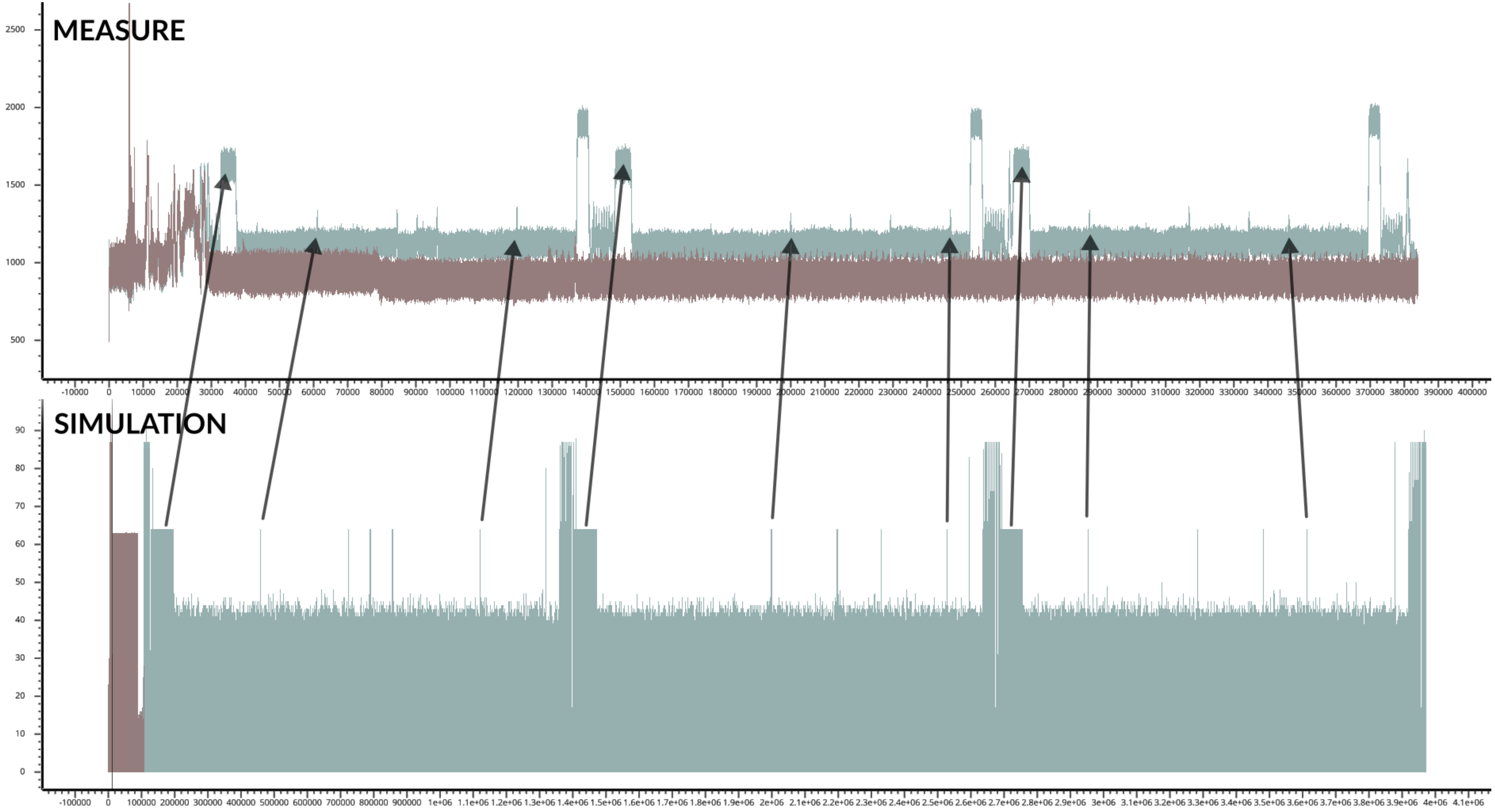


Side-Channel Analysis

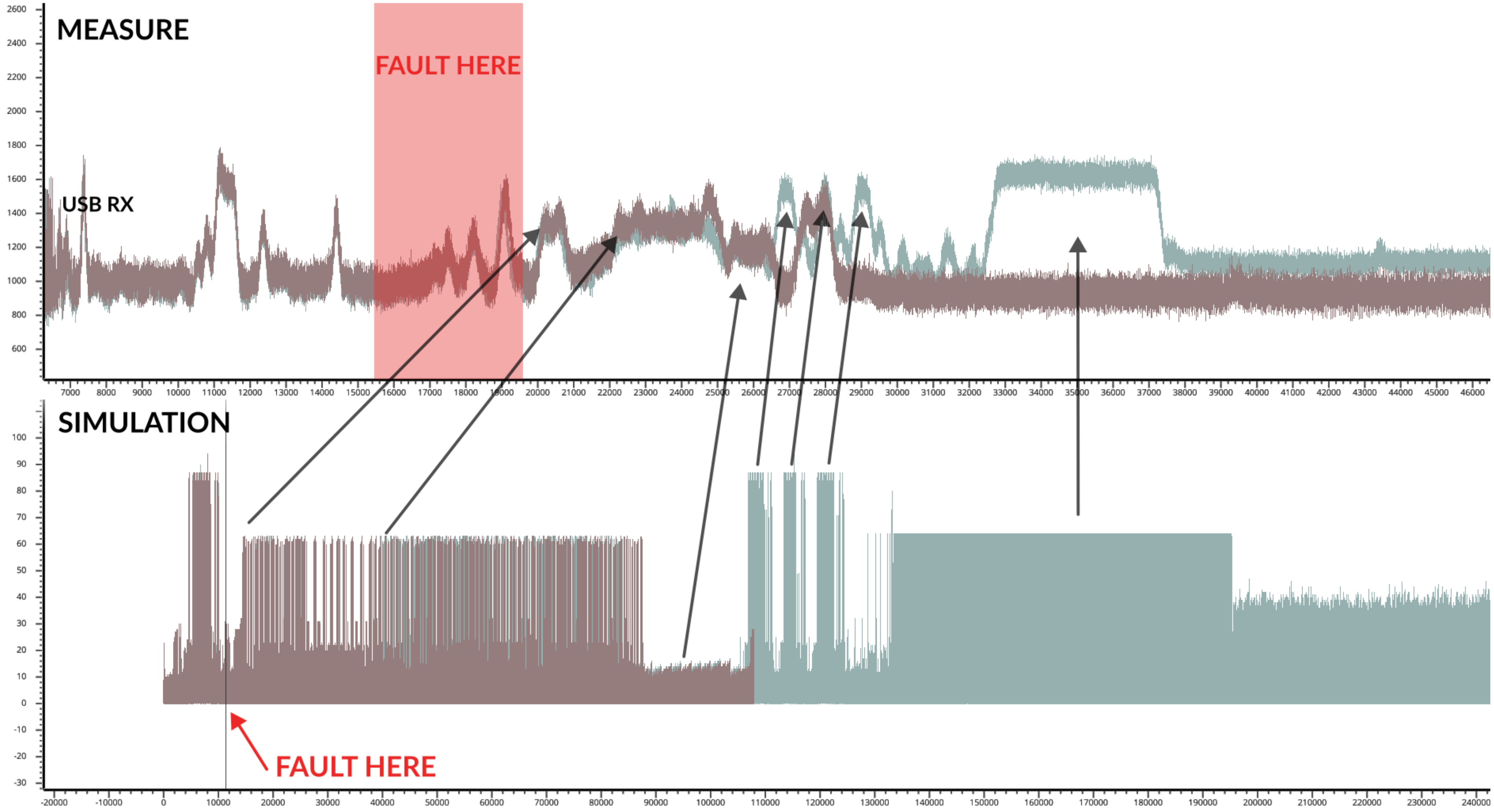
Average of 1000 synced traces



Simulation mapping



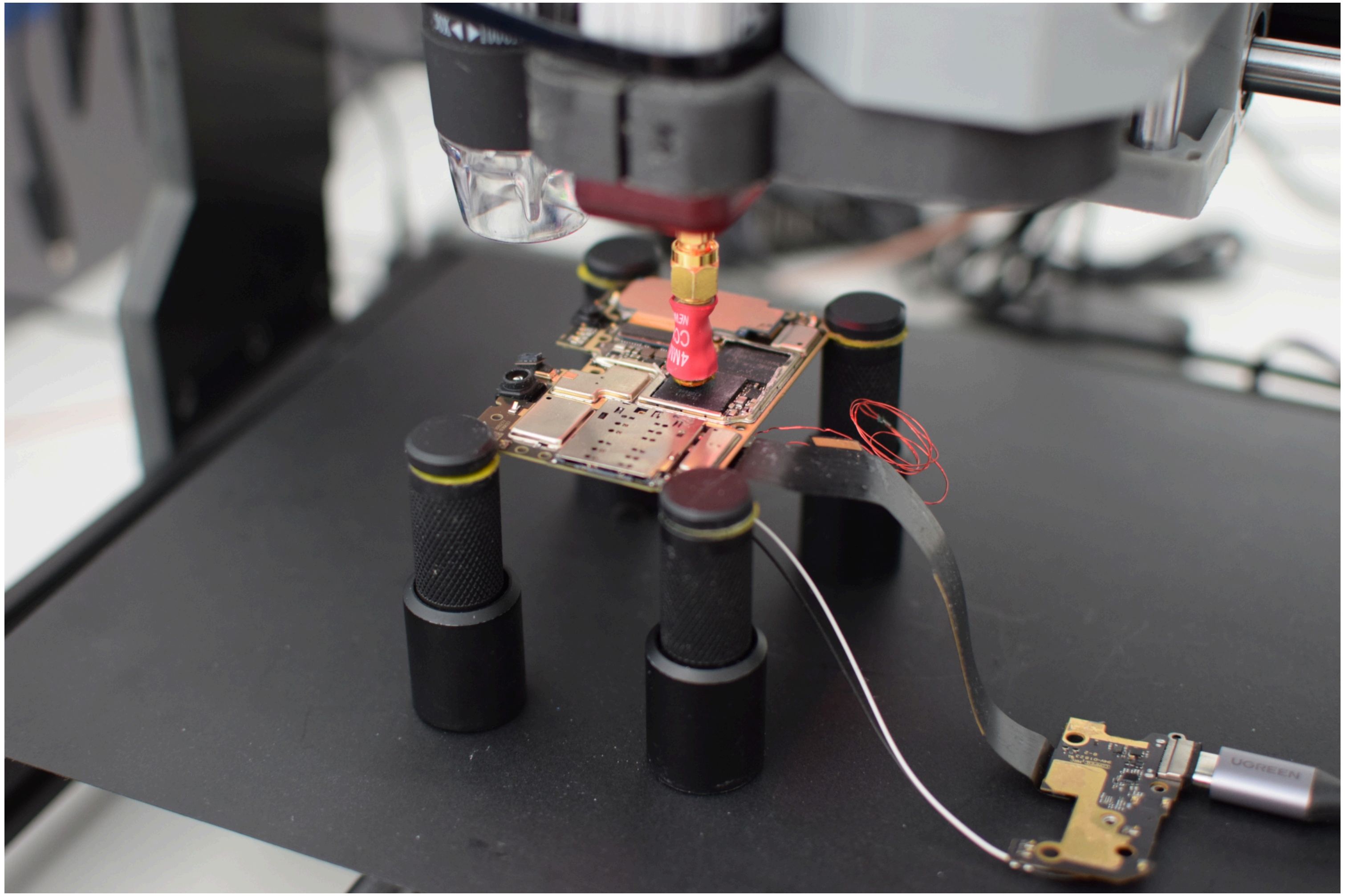
Simulation mapping



Fault injection campaign

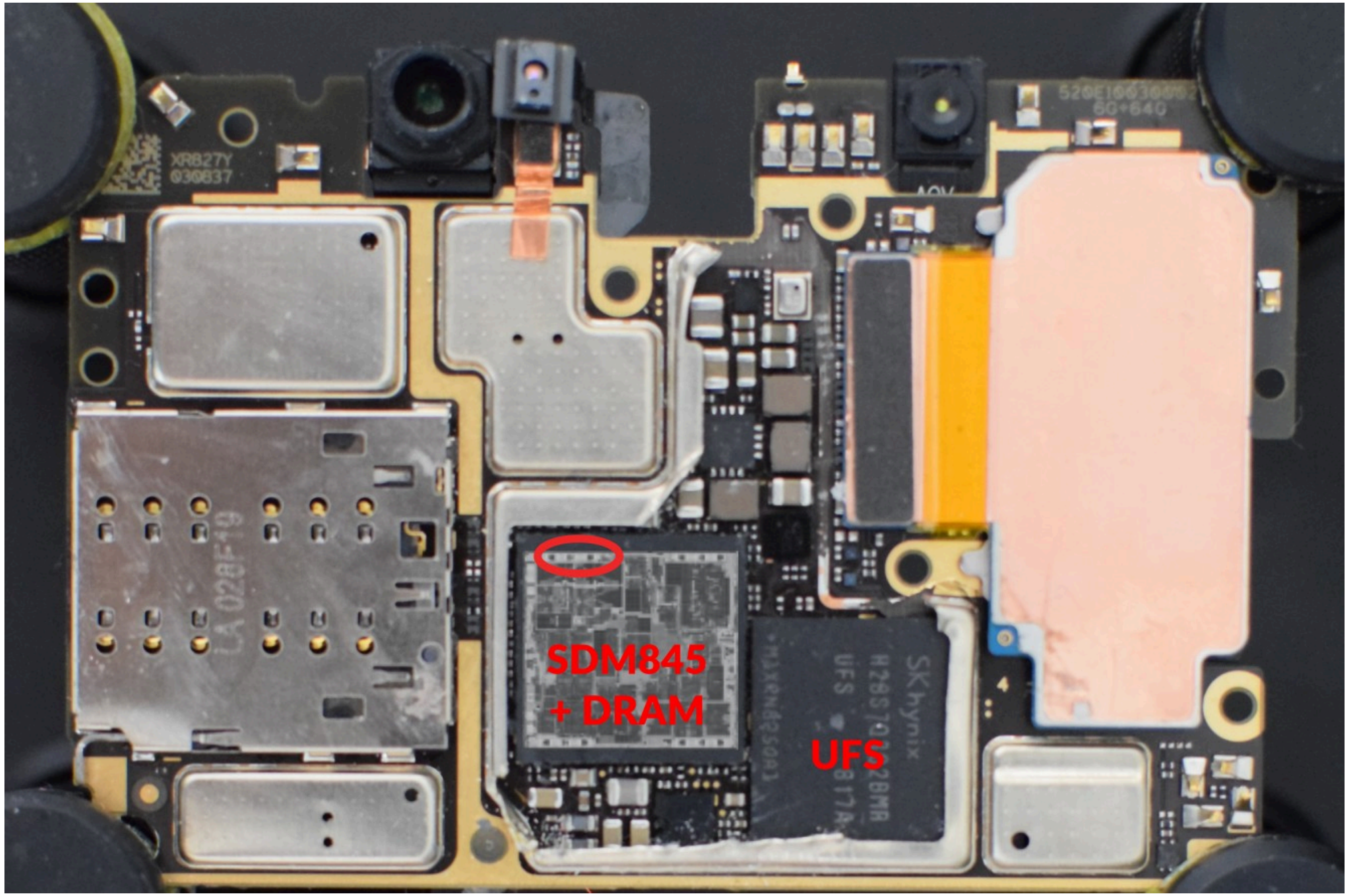
EMFI above SoC

Problem: DRAM stacked on SoC acts as a shield.



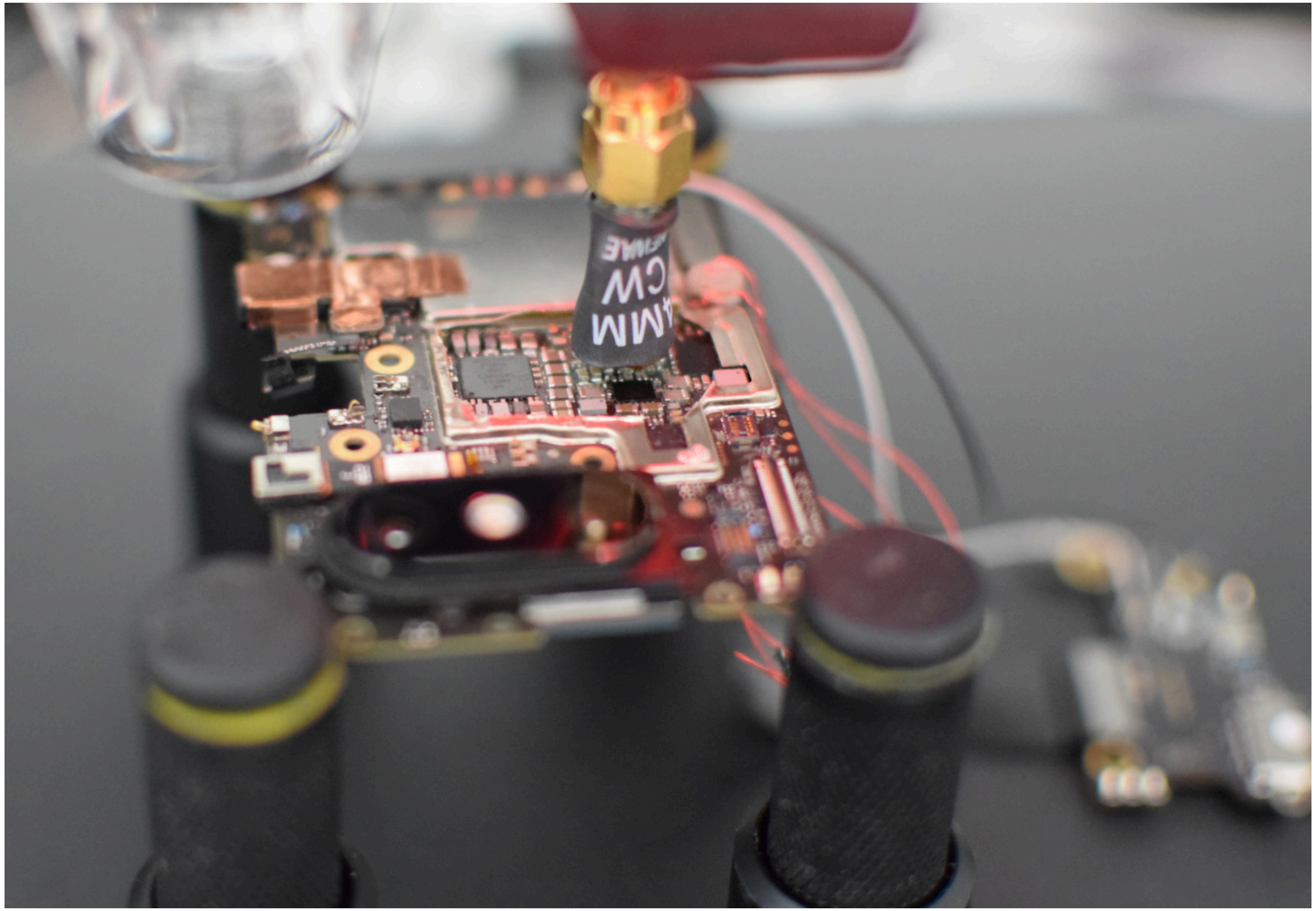
EMFI above SoC

Observation: crash on the side at 500V



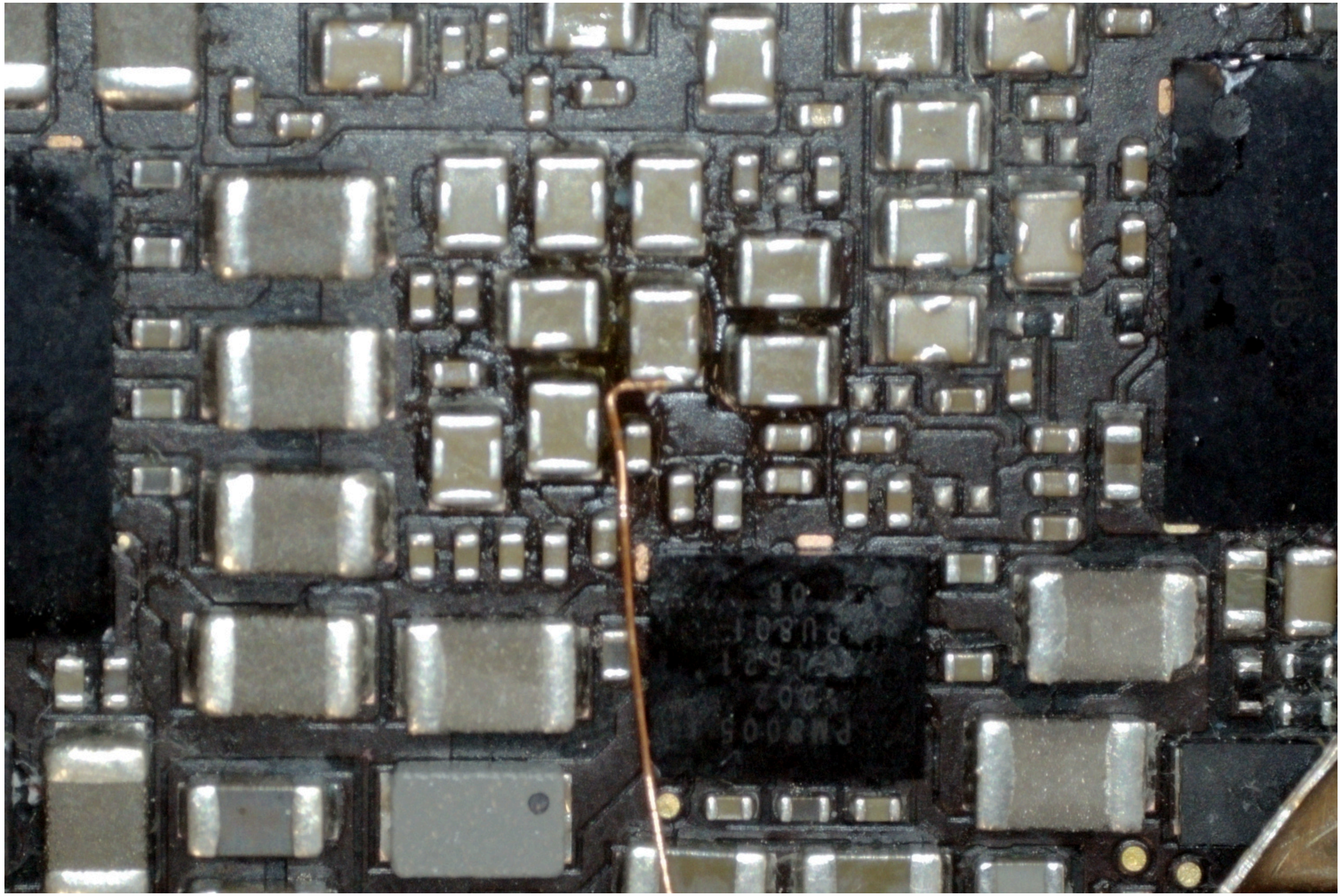
EMFI on capacitors

Problem: four-terminals capacitors have reduced EM sensibility.

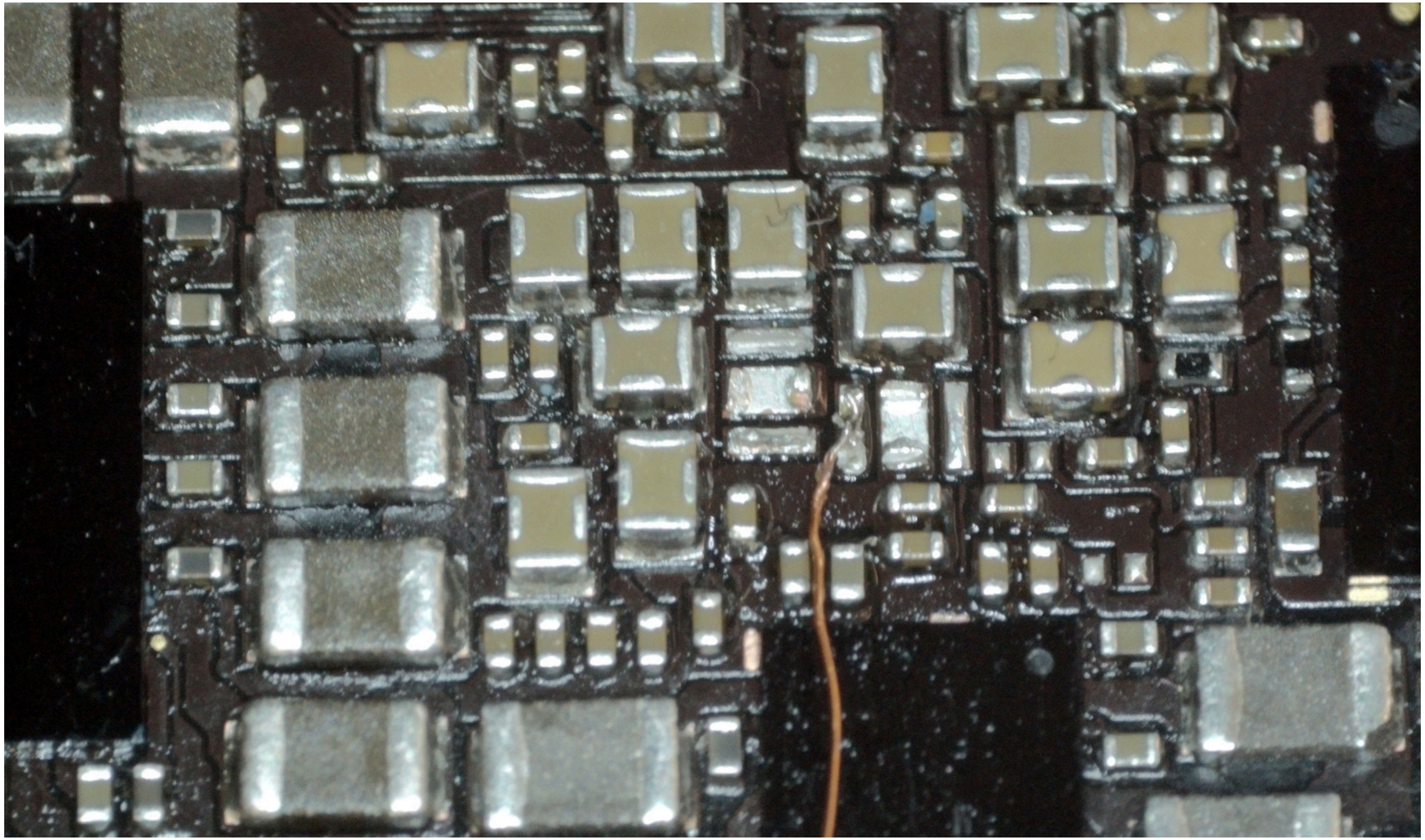


Power glitching with capacitors

Problem: no effects with ground crowbar glitch.



Power glitching without capacitors



Conclusion

- 7 unknown CMD response
- 4 read data error 0x0D (fault happens too early)
- 10 USB error
- no interesting faults and very low fault rate

Further static analysis reveals hardening (double checks, secured booleans, added jitter).

SDM845 has fault injection hardening that MSM8916 did not have.

Discussion

Bibliography

- Clément Fanjas and al., "PoP DRAM: A new EMFI approach based on EM-induced glitches on SoC", https://cea.hal.science/cea-04948475v1
- B. Kerler, unofficial Qualcomm Firehose / Sahara tools, https://github.com/bkerler/edl
- Qualcomm Glossary, postmarketOS Wiki, https://wiki.postmarketos.org/wiki/Qualcomm_Glossary
- Niclas Kühnapfel and al., 2022, "EM-Fault It Yourself: Building a Replicable EMFI Setup for Desktop and Server Hardware", https://arxiv.org/abs/2209.09835
- Hector Marco, 2022, BlackHat Europe, "Vlind Glitch: A Blind VCC Glitching Technique to Bypass the Secure Boot of the Qualcomm MSM8916 Mobile SoC"

SYNACKTIV

https://synacktiv.com

https://bsky.app/profile/synacktiv.com

https://www.linkedin.com/company/synacktiv