
black hat®
ASIA 2026
APRIL 21-24, 2026
MARINA BAY SANDS / SINGAPORE

When Flash Reveals Its Secrets

Advanced Glitching Leveraging Hidden CPU–eMMC Behavior

April 2026

Who we are ?

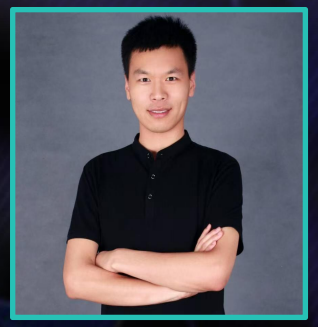
China Telecom Cybersecurity Technology

CloudHunter Tense Lab

CloudHunter Tense Lab is a security lab under China Telecom Cybersecurity Technology CloudHunter Product Line. Guided by the philosophy of exploring the obscure and uncovering hidden risks, we focus on cutting-edge underlying security research and advanced offensive-defensive breakthroughs.

Core Research Areas:

- In-depth Vulnerability discovery
- Fault Injection & Hardware Security
- Reverse Engineering & Stealthy Behavior Analysis

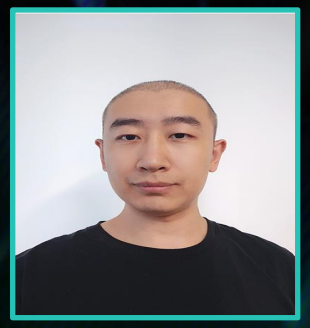


Jie Fu

Security Engineer & Head of CloudHunter Team

Shaohua Zhang

Security Engineer & CloudHunter



Yang Chen

Security Engineer specializing in Binary Reverse Engineering

Yujie Lu

Binary Security Engineer

Qiang Qin

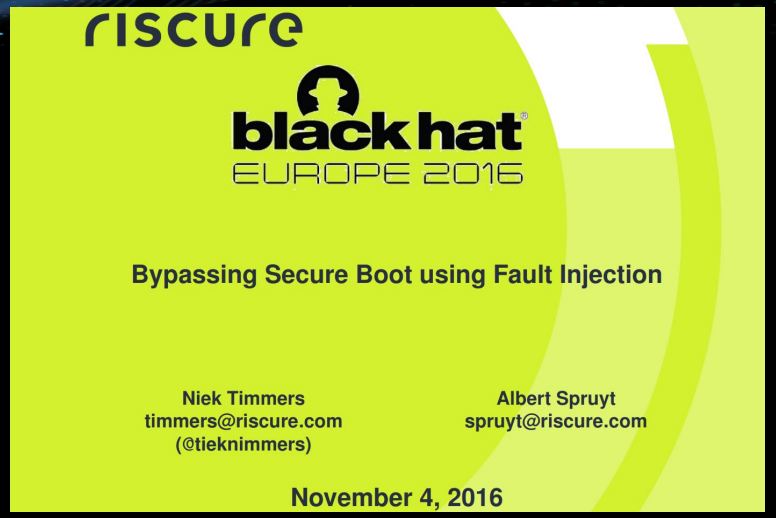
Senior Security Expert, China Telecom

Research Background

Secure Boot Security Mechanisms and Attack Challenges

- What is Secure Boot ?
  - the root of trust in modern embedded systems.
  - Ensures firmware integrity through chained signature verification.
- How to How to Bypass ?
  - Fault Injection
- Traditional Fault Injection Limitations:
  - Unclear timing window
  - Hardware damage risk
  - Low success rate

Secure Boot Security Mechanisms and Attack Challenges



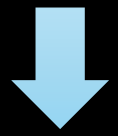
....

- Precise timing positioning
- Low attack success rate
- Narrow attack window
- Hardware damage risk
- Complex platform differences
- ...

- Trigger alignment
- High-voltage / EM glitch control
- Secure mechanism analysis
- ...

Our Job：Cross-Domain Signal Analysis for Precise Timing Positioning

- Analyze micro-characteristics of CPU electromagnetic radiation.
- Analyze underlying timing behaviors of eMMC and DDR.
- Construct a cross-domain hardware signal correlation model.
- Correlate electromagnetic and bus-level signals.
- Calibrate the real execution time of Secure Boot verification logic.



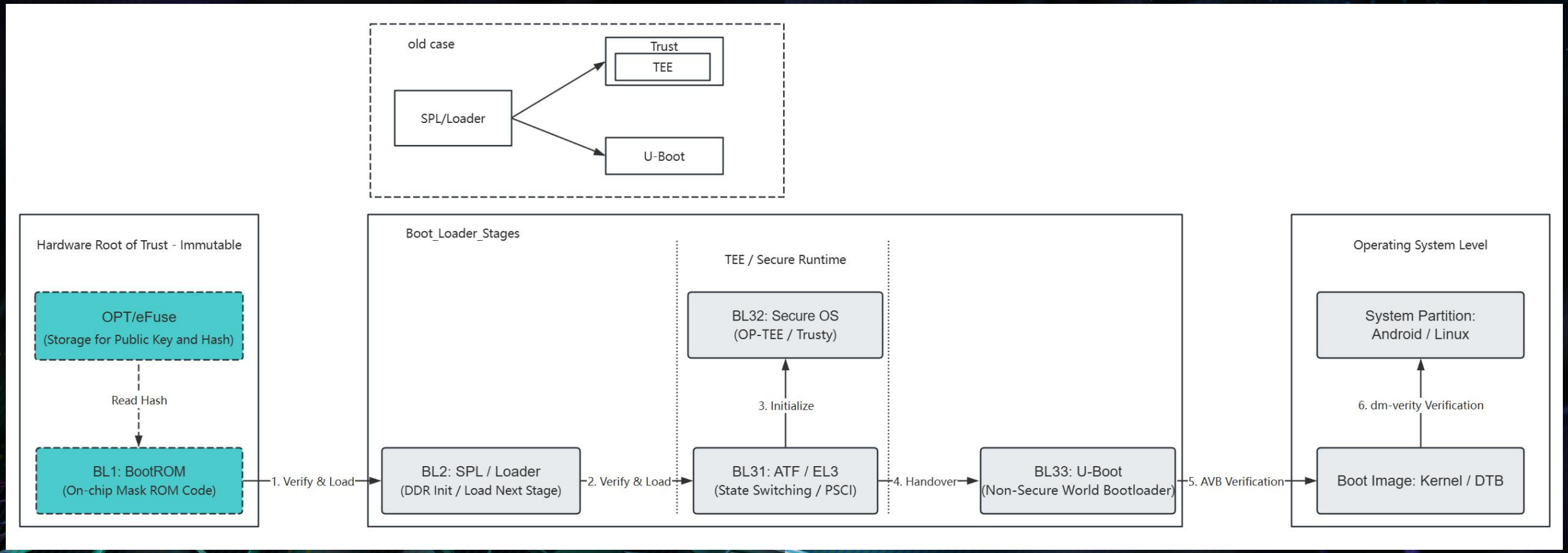
Precise Timing Positioning

Today's Agenda

- **Preliminary work**
  - Overview of the conventional Secure Boot flow
  - Configuration of the Secure Boot test environment
  - Modification of the selected experimental hardware
- **Serial-triggered scheme (conventional fault injection method)**
- **Our proposed method**
  - Cross-domain hardware signal acquisition and analysis
  - Serial-free triggering scheme
- **Conclusion**
- **Future work**

1. Preliminary Work

1.1 Overview of the conventional Secure Boot flow



1.2 Configuration of the Secure Boot test environment

Benchmark Environment (Normal Boot)

- development board with eFuse configured and Secure Boot enabled.
- Environment Verification: Serial debug logs ”Starting kernel...”

Abnormal Environment (Tampered Boot)

- Firmware Extraction
- Firmware Tampering
- Abnormality Verification: Serial debug logs“Secure Boot Verification Failed”,the system enters MaskROM mode.
- Serial-Free Environment Simulation: Physically disconnect the serial port.

1.3 Serial Log Comparison

Benchmark VS Abnormal

```
Trying to boot from MMC1
Trying fit image at 0x[redacted]00 sector
## Verified-boot: 1
sha256,rsa2048:dev## Verified-boot: 1
+
## Checking atf-1 0x0[redacted] ... sha256([redacted] ..) + OK
## Checking uboot 0x[redacted]0 ... sha256(b[redacted]/...) + OK
## Checking fdt 0x0[redacted]8 ... sha256(e[redacted] ..) + OK
## Checking atf-2 0x[redacted]000 ... sha256(a[redacted] ..) + OK
## Checking atf-3 0x[redacted]000 ... sha256(f[redacted] ..) + OK
## Checking optee 0x[redacted]000 ... sha256([redacted] ..) + OK
Jumping to U-Boot(0x0[redacted]000) via ARM Trusted Firmware(0x00[redacted]00)
Total: 179.121 ms

INFO:    Preloader serial: 2
NOTICE:  BL31: v2.3():v2.3-616-gdc1125f48:finley.xiao
NOTICE:  BL31: Built : 16:41:17, Jul 13 2023
INFO:    spec: 0x1
INFO:    ext 32k is not valid
INFO:    ddr: stride-en 4CH
INFO:    GICv3 without legacy support detected.
INFO:    ARM GICv3 driver initialized in EL3
INFO:    valid_cpu_msk=0xff bcore0_rst = 0x0, bcore1_rst = 0x0
INFO:    system boots from cpu-hwid-0
INFO:    idle_st=0x21fff, pd_st=0x11fff9, repair_st=0xfff70001
INFO:    dfs DDR fsp_params[0].freq_mhz= 2112MHz
INFO:    dfs DDR fsp_params[1].freq_mhz= 528MHz
```

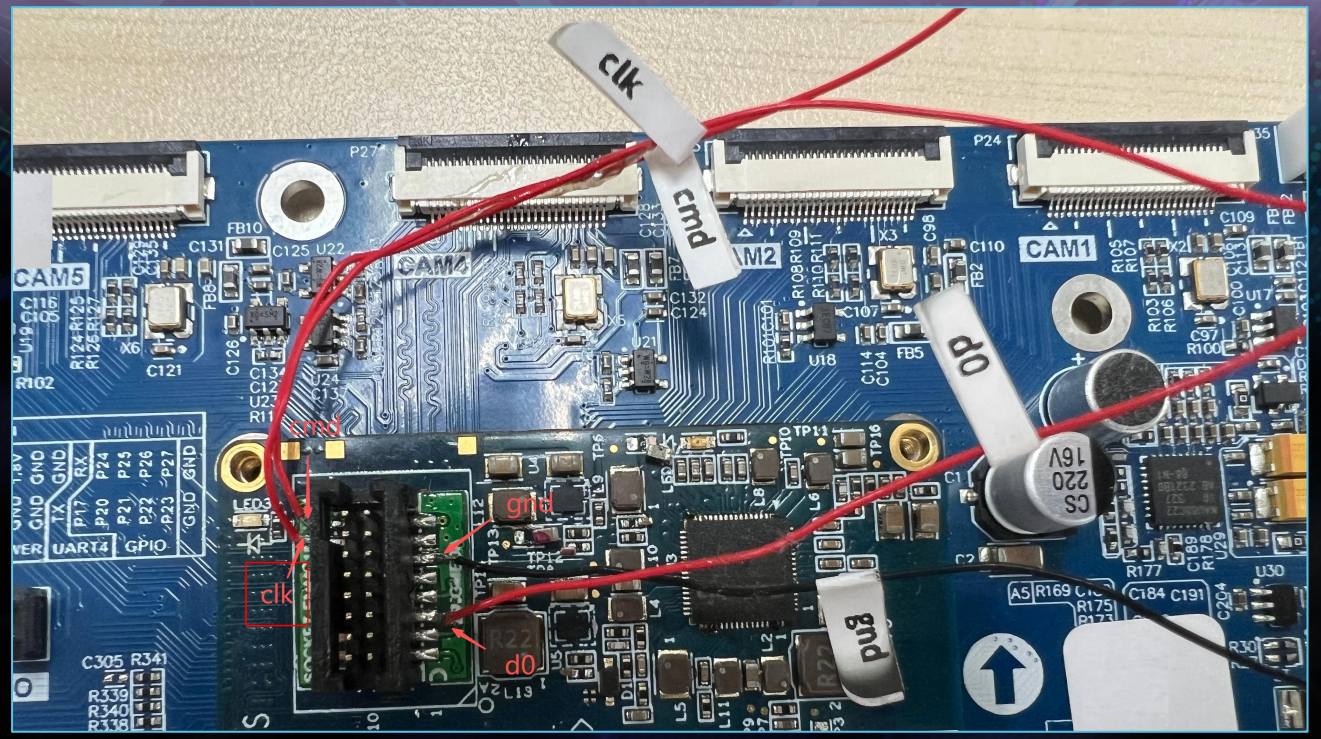
```
+
## Checking atf-1 0x0[redacted] ... sha256([redacted]bb...) + OK
## Checking uboot 0x0[redacted] ... sha256([redacted] ..) + OK
## Checking fdt 0x00[redacted] ... sha256(e[redacted] .) + OK
## Checking atf-2 0x[redacted] ... sha256([redacted] ..) + OK
## Checking atf-3 0x[redacted] ... sha256[redacted] ...) + OK
## Checking optee 0x0[redacted] ... sha256 Bad hash: ca1dbbb7673cf8c4ccbdcb49bd690d2b833c1(
 error!
Bad hash value for 'hash' hash node in 'optee' image node
Trying fit image at 0x[redacted] sector
## Verified-boot: 1
sha256,rsa2048:dev## Verified-boot: 1
+
## Checking atf-1 0x[redacted]0 ... sha25[redacted] ..) + OK
## Checking uboot 0x[redacted]00 ... sha25[redacted] ...) + OK
## Checking fdt 0x0[redacted] .. sha256([redacted] ..) + OK
## Checking atf-2 0x[redacted] ... sha256,[redacted] ...) + OK
## Checking atf-3 0x[redacted]00 ... sha256([redacted] ..) + OK
## Checking optee 0x[redacted] .. sha256 Bad h[redacted]074bad808e7707f1b3ebad4db795
 error!
Bad hash value for 'hash' hash node in 'optee' image node
Trying fit image at 0[redacted]0 sector
## Verified-boot: 1
sha256,rsa2048:dev## Verified-boot: 1
+
## Checking atf-1 [redacted]00 ... sha256([redacted]43bb...) + OK
## Checking uboot [redacted]0 ... sha256[redacted]7...) + OK
## Checking fdt [redacted] ... sha256(e5[redacted] ..) + OK
## Checking atf-[redacted]00 ... sha256[redacted]afd...) + OK
## Checking atf-3[redacted]00 ... sha25[redacted] ..) + OK
## Checking optee[redacted]0 ... sha25[redacted]bb7673cf8c4ccbdcb49bd690d2b833c1(
 error!
Bad hash value for 'hash' hash node in 'optee' image node
Trying fit image at 0[redacted]0 sector
## Verified-bo
sha256,rsa2048
+
## Checking at
## Checking ub
## Checking fd
## Checking at
## Checking at
```

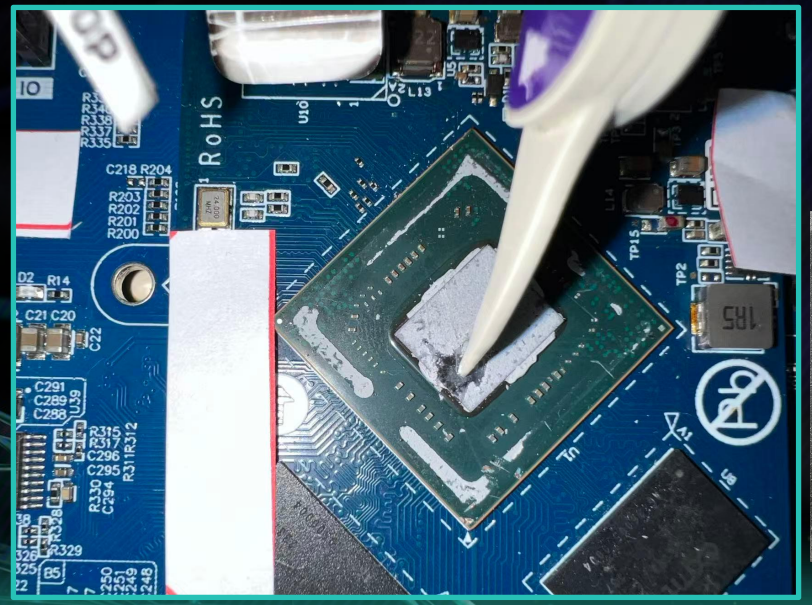
1 byte changed

```
3D:36F0h: 65 66 61 75 6C 74 3A 20 25 73 0A 00 23 23 23 20  efault: %s..###
3D:3700h: 45 52 52 4F 52 20 23 23 23 20 50 6C 65 65 73 65  ERROR ### Pleese
3D:3710h: 20 52 45 53 45 54 20 74 68 65 20 62 6F 61 72 64   RESET the board
3D:3720h: 20 23 23 23 0A 00 43 52 43 33 32 20 66 6F 72 20   ###..CRC32 for
3D:3730h: 25 30 38 6C 78 20 2E 2E 2E 20 25 30 38 6C 78 20  %08lx ... %08lx
3D:3740h: 3D 3D 3E 20 25 30 38 6C 78 0A 00 45 52 52 4F 52  ==> %08lx..ERROR
```

1.4 Hardware Modification

- **eMMC：** leading out GND, D0, and CLK signals
- **CPU Electromagnetic Leakage:** using electromagnetic probe.
- Remove some filter capacitors to enhance the glitch attack effect.

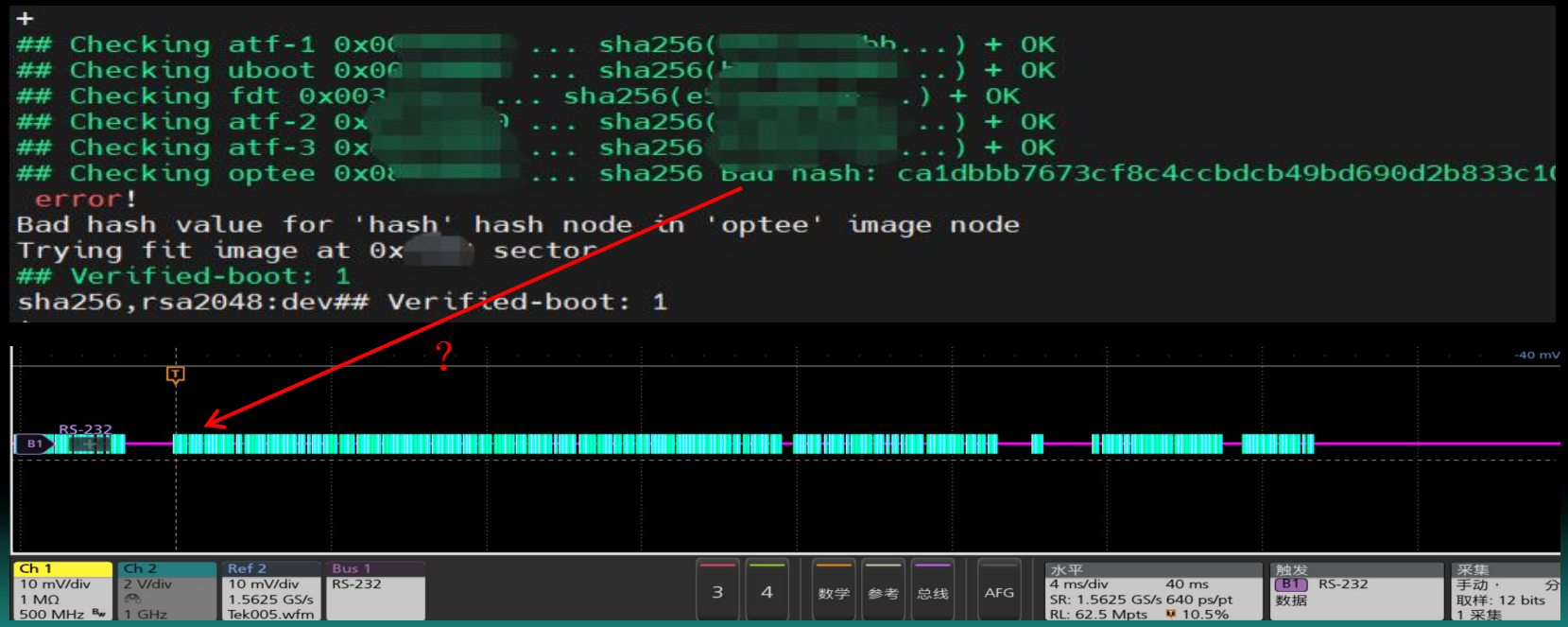




2. Serial Trigger Scheme

2.1 Traditional Trigger Scheme — Serial Trigger

- **Trigger mechanism:**
  - Triggered by software instructions.
  - The software execution process is mapped to externally observable hardware signals.
  - Using the UART serial waveform as the time anchor, fault injection synchronization is realized and the attack window is located.
- **Logic chain:** CPU executes instructions → UART transmits data → TX level changes → Oscilloscope / Glitch injection device captures the trigger point.
- **Core task:** Locate the "golden anchor point" before signature verification.



2.2 Locating the "Golden Anchor Point"

- By comparing logs, locate the verification branch.
- Reverse engineering helps identify a unique string as the golden anchor Point.

```
Trying to boot from MMC1
Trying fit image at 0x[redacted]00 sector
## Verified-boot: 1
sha256,rsa2048:dev## Verified-boot: 1
+
## Checking atf-1 0x0[redacted] ... sha256([redacted] ..) + OK
## Checking uboot 0x[redacted]0 ... sha256(b[redacted]/...) + OK
## Checking fdt 0x0[redacted]8 ... sha256(e[redacted] ..) + OK
## Checking atf-2 0x[redacted]000 ... sha256(a[redacted] ..) + OK
## Checking atf-3 0x[redacted]00 ... sha256(f[redacted] ..) + OK
## Checking optee 0x[redacted]000 ... sha256([redacted] ..) + OK
Jumping to U-Boot(0x0[redacted]000) via ARM Trusted Firmware(0x00[redacted]00)
Total: 179.121 ms

INFO:    Preloader serial: 2
NOTICE:  BL31: v2.3():v2.3-616-gdc1125f48:finley.xiao
NOTICE:  BL31: Built : 16:41:17, Jul 13 2023
INFO:    spec: 0x1
INFO:    ext 32k is not valid
INFO:    ddr: stride-en 4CH
INFO:    GICv3 without legacy support detected.
INFO:    ARM GICv3 driver initialized in EL3
INFO:    valid_cpu_msk=0xff bcore0_rst = 0x0, bcore1_rst = 0x0
INFO:    system boots from cpu-hwid-0
INFO:    idle_st=0x21fff, pd_st=0x11fff9, repair_st=0xfff70001
INFO:    dfs DDR fsp_params[0].freq_mhz= 2112MHz
INFO:    dfs DDR fsp_params[1].freq_mhz= 528MHz
```

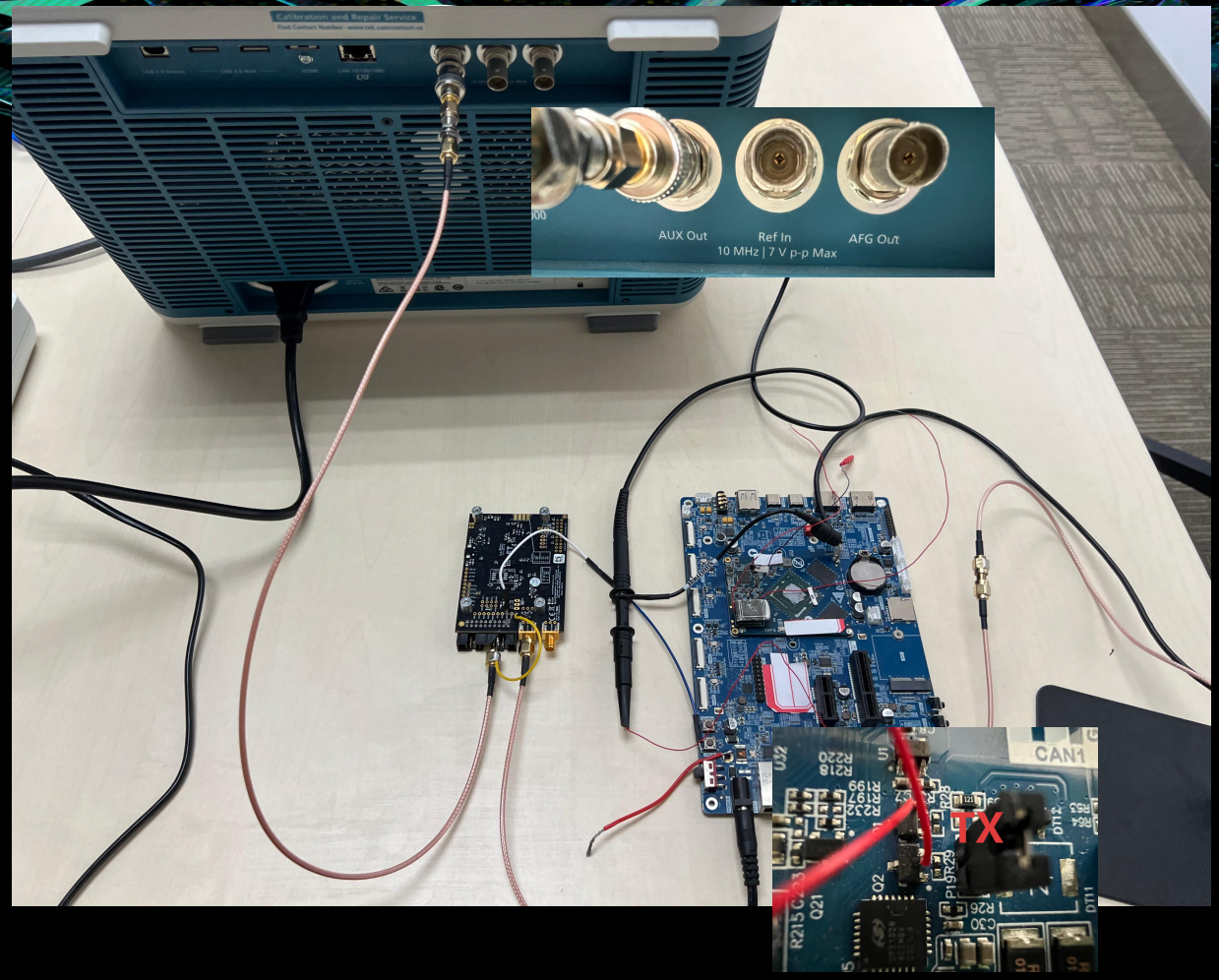
```
+
## Checking atf-[redacted]6([redacted]...) + OK
## Checking uboo[redacted]6([redacted] ..) + OK
## Checking fdt [redacted] .) + OK
## Checking atf-[redacted]6([redacted] ..) + OK
## Checking atf-3 0x[redacted] ... sha256[redacted] ...) + OK
## Checking optee 0x0[redacted] ... sha256 Bad hash: ca1dbbb7673cf8c4ccbdcb49bd690d2b833c1(
 error!
Bad hash value for 'hash' hash node in 'optee' image node
Trying fit image at 0x[redacted] sector
## Verified-boot: 1
sha256,rsa2048:dev## Verified-boot: 1
```

084

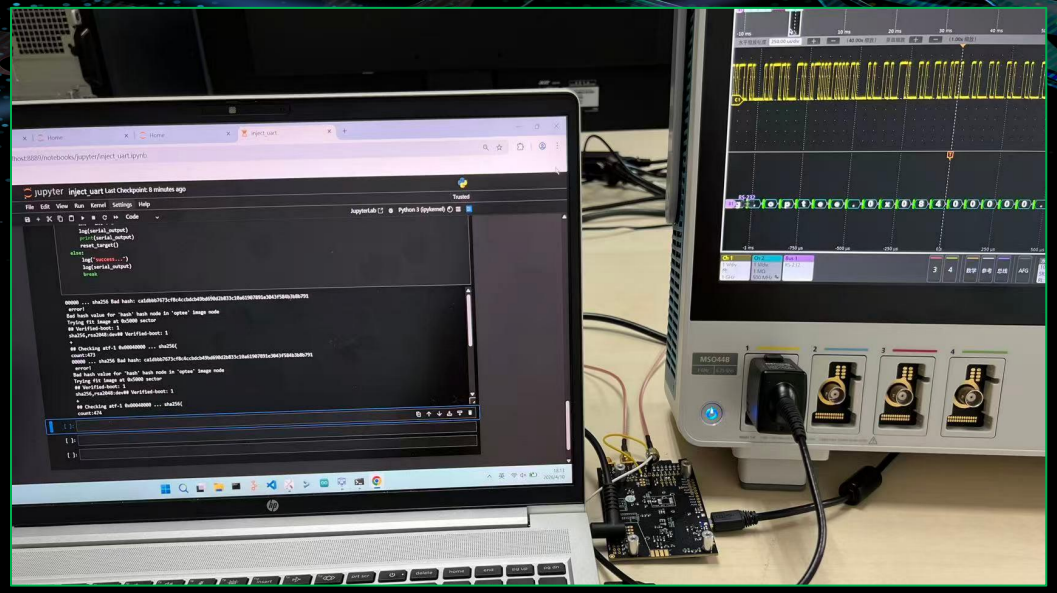
```
41  if ( (unsigned int)sub_554C(a3, a4, v21, v23, &v19) )// calculate_hash
42  {                                                                          ; __int64 __fastcall sub_83B8
43    v10 = "Unsupported hash algorithm";                                      sub_83B8
44    goto LABEL_3;
45  }
46  if ( v19 != v20 )
47  {                                                    E6 03 00 AA  MOV     X6, X0
48    v10 = "Bad hash value len";                        04 00 80 D2  MOV     X4, #0
49    goto LABEL_3;
50  }                                                                          loc_83C0
51  v11 = sub_83B8(v23, v22, v19);              // memcmp 5F 00 04 EB  CMP     X2, X4
52  if ( v11 )                                           61 00 00 54  B.NE    loc_83D0
53  {                                                    00 00 80 52  MOV     W0, #0
54    v15 = 0i64;                                        06 00 00 14  B       locret_83E4
55    sub_8B3C(" Bad hash: ");
56    while ( v19 > (int)v15 )                                                 ; ---------------------------
57    {                                                                        loc_83D0
58      v16 = *((unsigned __int8 *)v23 + v15++);         C3 68 64 38  LDRB    W3, [X6,X4]
59      sub_8B3C("%02x", v16);                           25 68 64 38  LDRB    W5, [X1,X4]
60    }                                                  84 04 00 91  ADD     X4, X4, #1
61    sub_8B3C("\n");                                    60 00 05 6B  SUBS    W0, W3, W5
62    v10 = "Bad hash value";                            00 FF FF 54  B.EQ    loc_83C0
63    goto LABEL_3;
64  }                                                                          locret_83E4
65  v17 = 0i64;                                          C0 03 5F D6  RET
66  sub_8B3C("(", v14);                                                      ; End of function sub_83B8
```

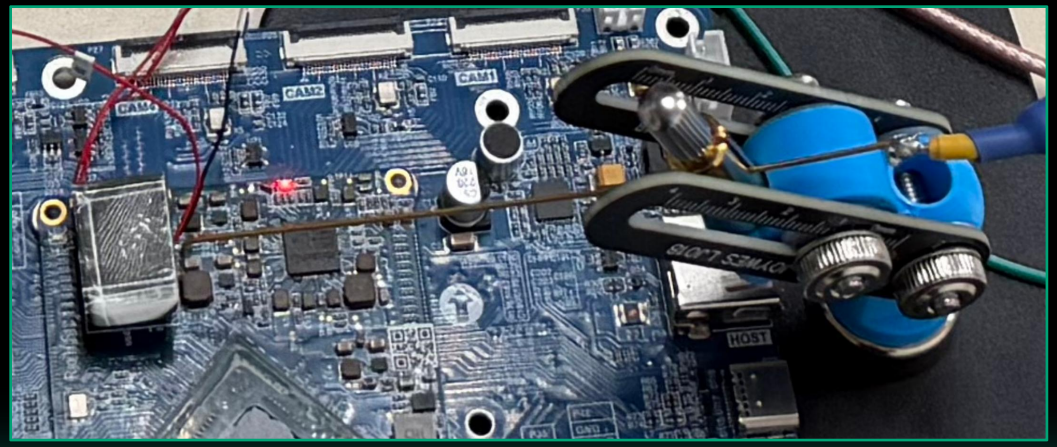
2.3 Specific Fault Attack Process

- The UART TX is connected to the oscilloscope pro be,and the oscilloscope performs waveform featu re matching (golden anchor matching).
- The trigger signal is transmitted to ChipWhisperer -Lite in real time.
- ChipWhisperer receives the trigger, performs rang e scanning, and injects voltage glitches.



2.4 Fault Attack Video with Serial Port Enabled





Home | inject_uart | localhost:8888/notebooks/jupyter/inject_uart.ipynb

jupyter inject_uart Last Checkpoint: 1 minute ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

Run this cell and advance (Shift+Enter)

```
[ ]: import time
     import serial
     import subprocess
     import sys
     import tqdm
```

```
[ ]: SCOPETYPE = 'OPENADC'
     PLATFORM = 'CWLITEARM'
     %run "./Setup_Scripts/Setup_Generic.ipynb"
```

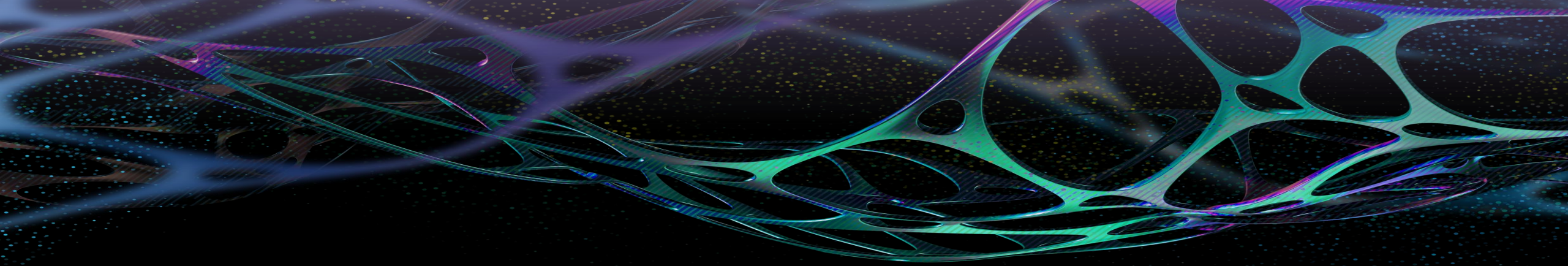
```
[ ]: def enablearm():
         SCOPETYPE = 'OPENADC'
         PLATFORM = 'CWLITEARM'
         %run "./Setup_Scripts/Setup_Generic.ipynb"

         time.sleep(0.5)
         # scope.glitch.clk_src = "clkgen"
         scope.glitch.output = "enable_only"    # "enable_only": Output is high for glitch.repeat cycles.
         scope.glitch.trigger_src = "ext_single"  # one set of glitch events is emitted when the trigger condition is satisfied
         scope.glitch.clkgen_src = "system"
         scope.clock.clkgen_freq = 100E6
         scope.io.glitch_lp = False
         scope.io.glitch_hp = True
         scope.trigger.triggers = "tio4"
```

```
[ ]: scope.io.tio3 = True
```

```
[ ]: com6 = serial.Serial('com6', 115200, timeout=2)
```

```
[ ]: enablearm()
```

```
[ ]: import  time
```



What if there is no serial port output?

3. Cross-Domain Hardware Signal Acquisition and Analysis

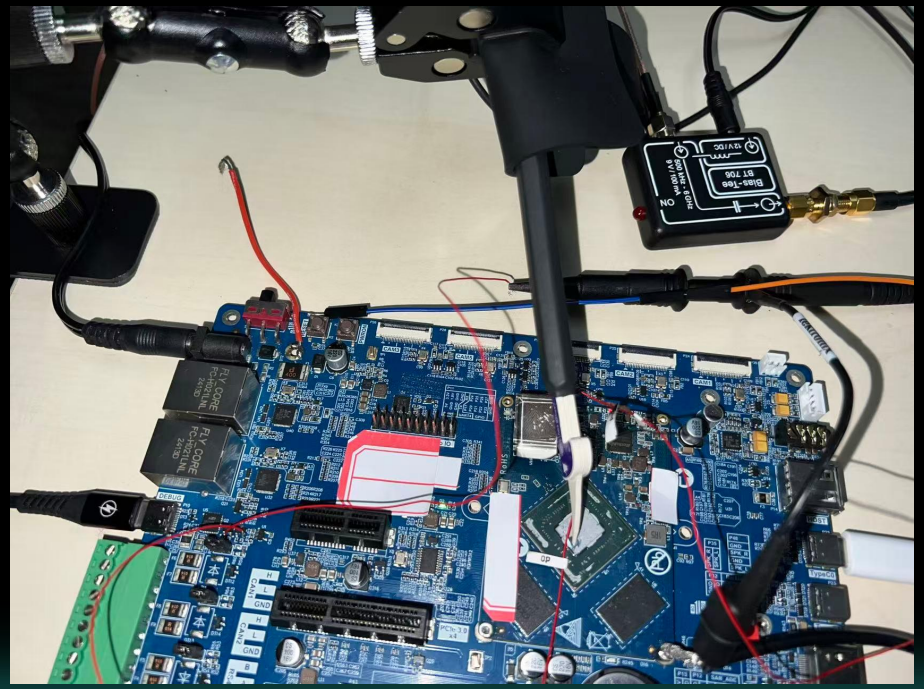
3.1 Analysis of Physical Side-Channel Signal Characteristics During Secure Boot Verification

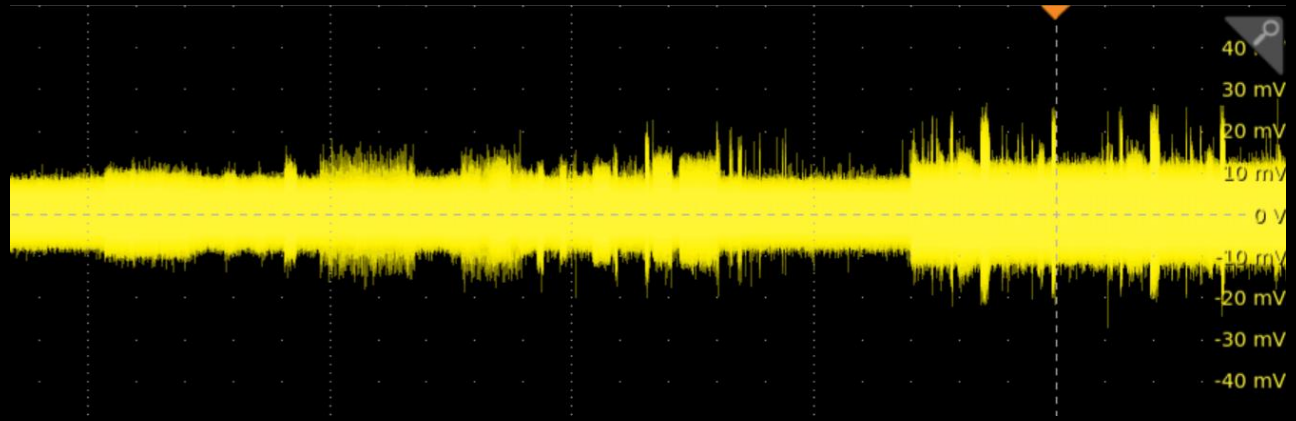
During Secure Boot verification, the CPU performs operations including reading the Bootloader signature, loading the public key, and executing decryption/hashing algorithms such as SHA, RSA, and ECDSA.

- **D0 (data line) characteristics:**
  - Intensive activity during image loading, waveforms show high-frequency toggling.
  - With strong data correlation.
- **CPU electromagnetic leakage characteristics:**
  - Strongly correlated with instruction execution, showing irregular high-frequency disturbances.
  - Energy increases significantly during encryption/hashing stages (e.g., SHA, RSA).
  - Distinguishable pattern differences exist between different code paths (Success / Fail).

3.2 Signal Acquisition – Electromagnetic Signal Acquisition

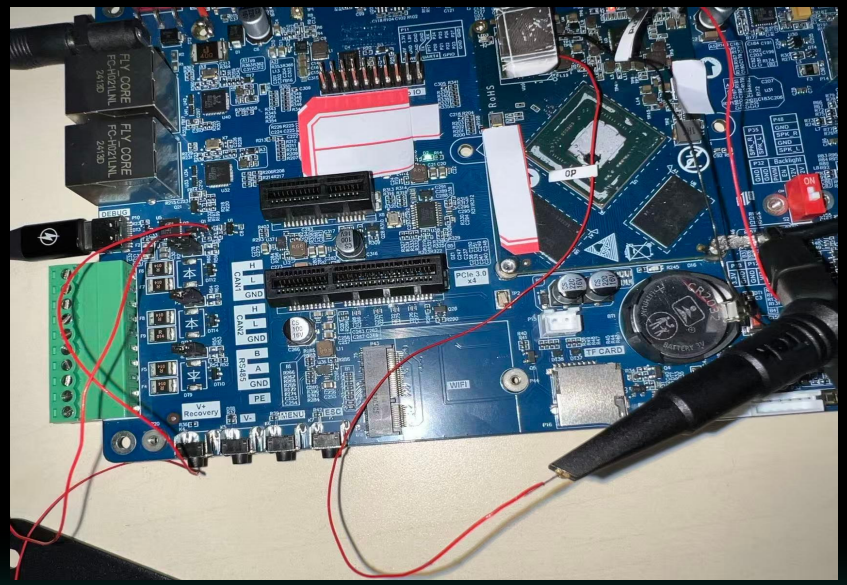
- Remove the CPU metal shield and heatsink to expose the CPU die.
- Use a near-field EM probe (bandwidth 10MHz~3GHz) close to the CPU core to capture fine electromagnetic radiation characteristics during instruction execution.
- Connect the probe output to an oscilloscope channel for synchronous sampling with eMMC signals.

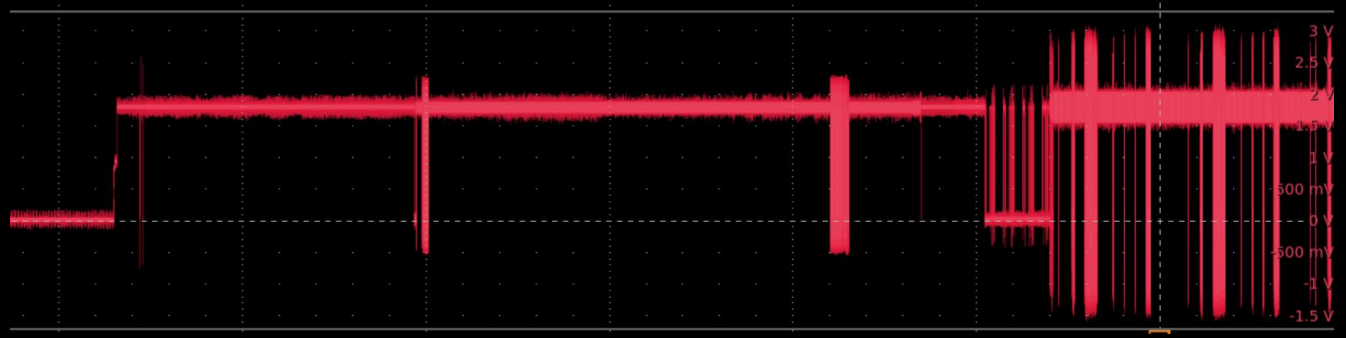




3.2 Signal Acquisition – eMMC Signal Acquisition

- eMMC Bus Signal Extraction.
- Remove the onboard eMMC chip and solder an eMMC adapter to lead out key pins including GND, CLK, CMD, and D0.
- Connect the output of the adapter to an oscilloscope for real-time monitoring of eMMC bus timing.
- Signal path:AXI system bus → eMMC Host controller → eMMC bus (CLK/CMD/D0) → eMMC chip.





3.3 Signal Acquisition Parameter Settings

Collect eMMC bus signals and CPU electromagnetic leakage separately under both normal and abnormal boot conditions.

- Acquisition parameters:
  - Sampling rate: 125 MHz initially for coarse positioning, gradually increased to 1.25 GHz for fine positioning.
  - Sampling depth: No less than 128 MSa, covering the complete boot process of approximately 2 seconds.
  - Rigger mode: Edge trigger, timeout trigger, or other similar methods on the eMMC D0 signal.

3.4 Signal Comparison Algorithm

Principle: Ignore low-amplitude noise and retain valid signals. Calculate cumulative intensity of positive and negative amplitudes separately, and measure differences between two windows using relative ratio.

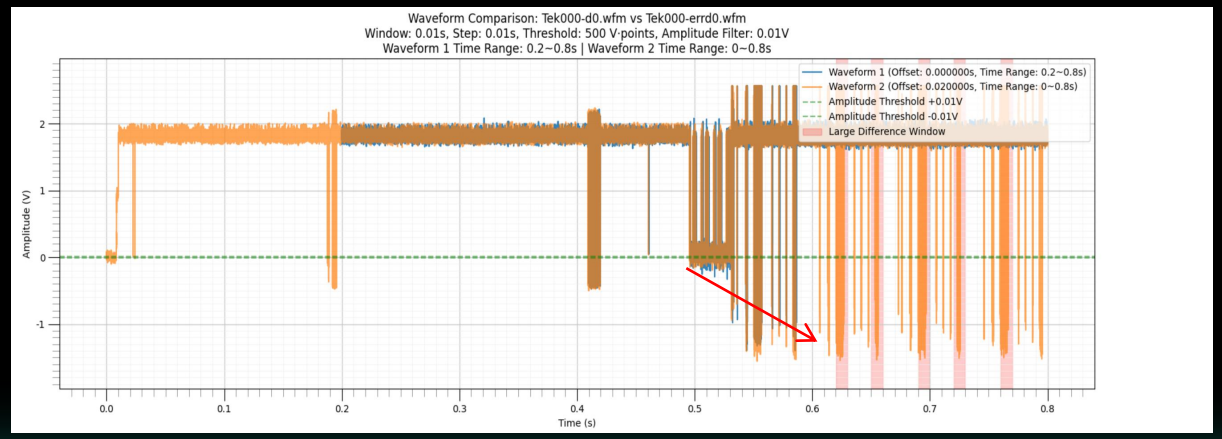
- Align normal and abnormal signals near the power-on reset trigger point.
- Perform comparison tests using the algorithm and adjust the threshold.
- The interval with a significant peak drop corresponds to the Secure Boot verification stage, where the two signal sets show the large st difference.

- *Amplitude Filtering (Noise Reduction)*: $|y| \geq T$
- *Positive-Negative Separation Statistics:*
  - *Sum of positive values:* $Spos = \sum(y > 0)$
  - *Sum of absolute negative values:* $Sneg = \sum|y < 0|$
  - *Difference calculation (ratio):*

$$D = max\left(\frac{max(S_{1,pos}, S_{2,pos})}{min(S_{1,pos}, S_{2,pos})}, \frac{max(S_{1,neg}, S_{2,neg})}{min(S_{1,neg}, S_{2,neg})}\right)$$

3.5 eMMC Signal Comparison

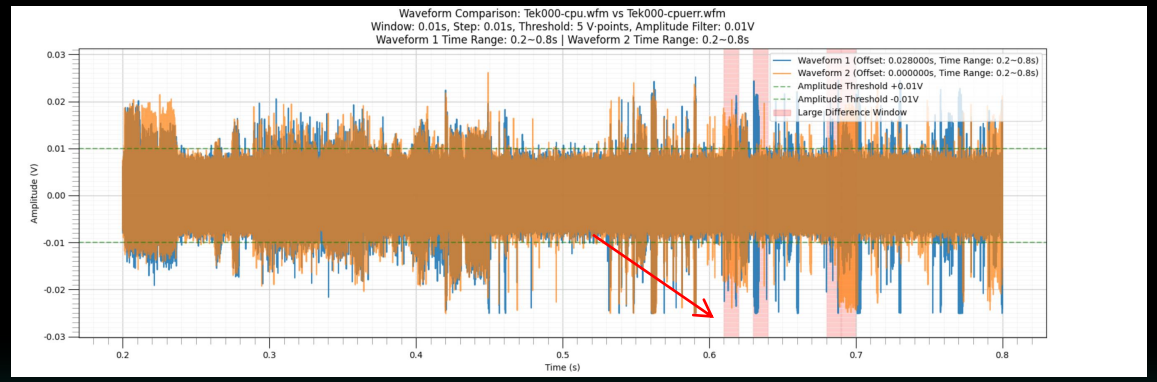
- The eMMC waveform in the baseline environment shows continuous high-level pulses during the period from 0.605s to 0.610s.
- The eMMC waveform in the abnormal environment shows a severe deviation from the normal waveform at 0.610s.
- Significant differences are clearly identified through algorithmic comparison.




```
Original start time: -2.100000 s, Manual offset: 0.020000 s, Final start time: 0.000000 s

【Alignment Information】
Waveform 1 after offset + filtering: 0.200000 ~ 0.800000 s
Waveform 2 after offset + filtering: 0.000000 ~ 0.800000 s
Final aligned time range: 0.200000 s to 0.800000 s
Sampling rate: 6.25e+06 Hz
Data length: 3750000
Amplitude filter threshold: 0.01 V (points with absolute values below this are excluded from calculation)
Total windows: 60
Large difference window 1 start: 0.610000 s, Sum difference: 501.741 V·points
Large difference window 2 start: 0.620000 s, Sum difference: 4072.162 V·points
Large difference window 3 start: 0.650000 s, Sum difference: 1536.606 V·points
Large difference window 4 start: 0.680000 s, Sum difference: 503.251 V·points
Large difference window 5 start: 0.690000 s, Sum difference: 4049.071 V·points
Large difference window 6 start: 0.720000 s, Sum difference: 1534.848 V·points
Large difference window 7 start: 0.760000 s, Sum difference: 4012.438 V·points
Large difference window 8 start: 0.790000 s, Sum difference: 1520.386 V·points

Final output: Amplitude filter threshold=0.01V, all start times with positive/negative amplitude sum difference > 50
  1: 0.610000 s
  2: 0.620000 s
  3: 0.650000 s
  4: 0.680000 s
  5: 0.690000 s
  6: 0.720000 s
  7: 0.760000 s
  8: 0.790000 s
After time range filtering: Number of data points 3750000, time range 0.200000 ~ 0.800000 s
Library estimated sampling rate: 6.25e+06 Hz, Manual sampling rate: 6.25e+06 Hz
Original start time: -5.100000 s, Manual offset: 0.000000 s, Final start time: 0.200000 s
After time range filtering: Number of data points 5000000, time range 0.000000 ~ 0.800000 s
Library estimated sampling rate: 6.25e+06 Hz, Manual sampling rate: 6.25e+06 Hz
Original start time: -2.100000 s, Manual offset: 0.020000 s, Final start time: 0.000000 s

Process finished with exit code 0
```

3.6 Electromagnetic Signal Comparison

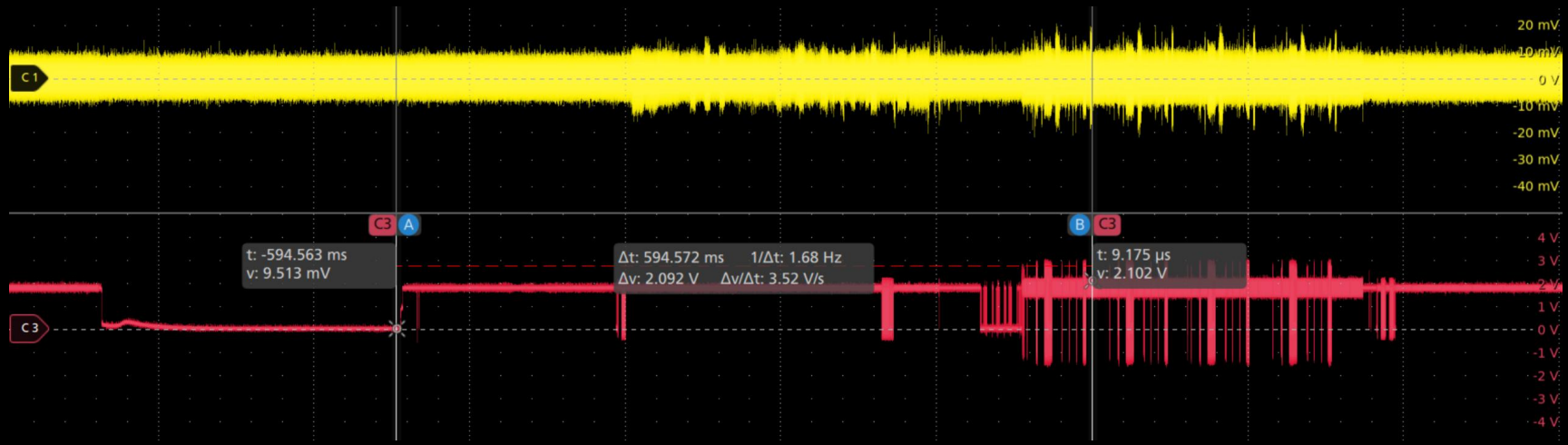
- The CPU electromagnetic signal in the baseline environment exhibits high-frequency fluctuations during 0.6098s–0.6106s.
- The CPU electromagnetic signal in the abnormal environment quickly stabilizes after fluctuating at 0.6098s.
- Algorithmic comparison identifies the difference window as 0.6098s–0.6106s.
- This window is highly similar to that of the eMMC signal, which proves the feasibility of cross-domain signal analysis.




```
After time range filtering: Number of data points 3750000, time range 0.200000 ~ 0.800000 s
Library estimated sampling rate: 6.25e+06 Hz, Manual sampling rate: 6.25e+06 Hz
Original start time: -5.100000 s, Manual offset: 0.000000 s, Final start time: 0.200000 s
After time range filtering: Number of data points 5000000, time range 0.000000 ~ 0.800000 s
Library estimated sampling rate: 6.25e+06 Hz, Manual sampling rate: 6.25e+06 Hz
Original start time: -5.100000 s, Manual offset: 0.028000 s, Final start time: 0.000000 s

【Alignment Information】
Waveform 1 after offset + filtering: 0.200000 ~ 0.800000 s
Waveform 2 after offset + filtering: 0.000000 ~ 0.800000 s
Final aligned time range: 0.200000 s to 0.800000 s
Sampling rate: 6.25e+06 Hz
Data length: 3750000
Amplitude filter threshold: 0.01 V (points with absolute values below this are excluded from calculation)
Total windows: 60
Large difference window 1 start: 0.610000 s, Sum difference: 5.223 V·points
Large difference window 2 start: 0.630000 s, Sum difference: 5.708 V·points
Large difference window 3 start: 0.680000 s, Sum difference: 5.857 V·points
Large difference window 4 start: 0.690000 s, Sum difference: 38.534 V·points

Final output: Amplitude filter threshold=0.01V, all start times with positive/negative amplitude sum difference >
  1: 0.610000 s
  2: 0.630000 s
  3: 0.680000 s
  4: 0.690000 s
After time range filtering: Number of data points 3750000, time range 0.200000 ~ 0.800000 s
Library estimated sampling rate: 6.25e+06 Hz, Manual sampling rate: 6.25e+06 Hz
Original start time: -5.100000 s, Manual offset: 0.000000 s, Final start time: 0.200000 s
After time range filtering: Number of data points 5000000, time range 0.000000 ~ 0.800000 s
Library estimated sampling rate: 6.25e+06 Hz, Manual sampling rate: 6.25e+06 Hz
Original start time: -5.100000 s, Manual offset: 0.028000 s, Final start time: 0.000000 s

Process finished with exit code 0
```

3.7 Comparison Summary

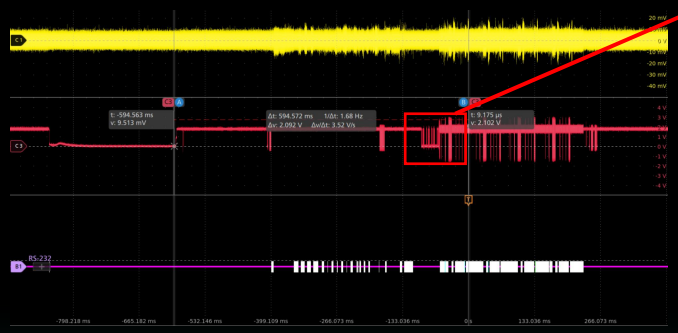
- By comparing firmware, both CPU EM and eMMC waveforms show differences after 600 ms from power-on, reducing the glitch window to about 10 ms.
- In the following steps, we will capture weak hash calculation signals to further narrow the range.

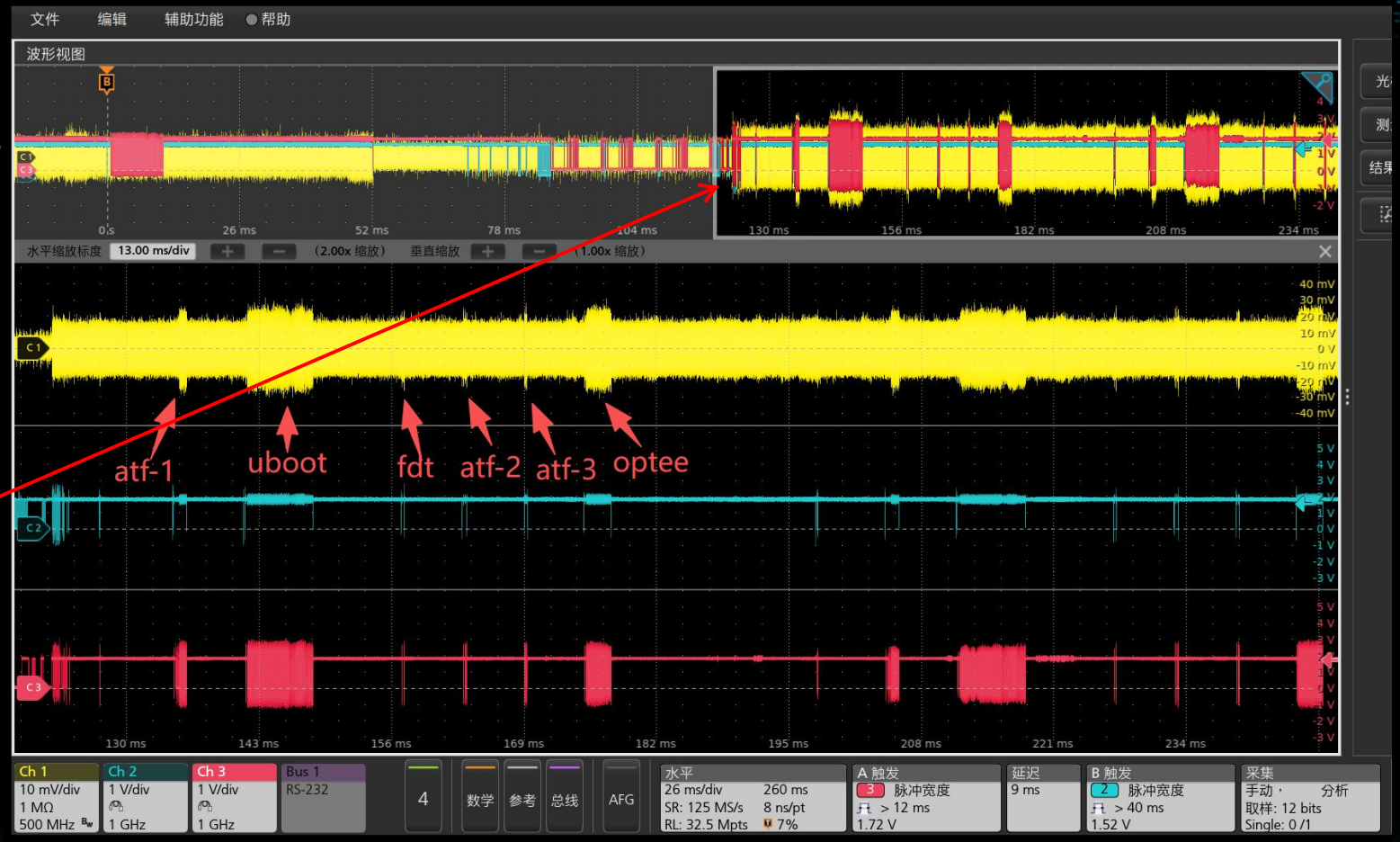


4. Serial-Free Trigger Scheme

4.1 Glitch Attack Window Without Serial Port

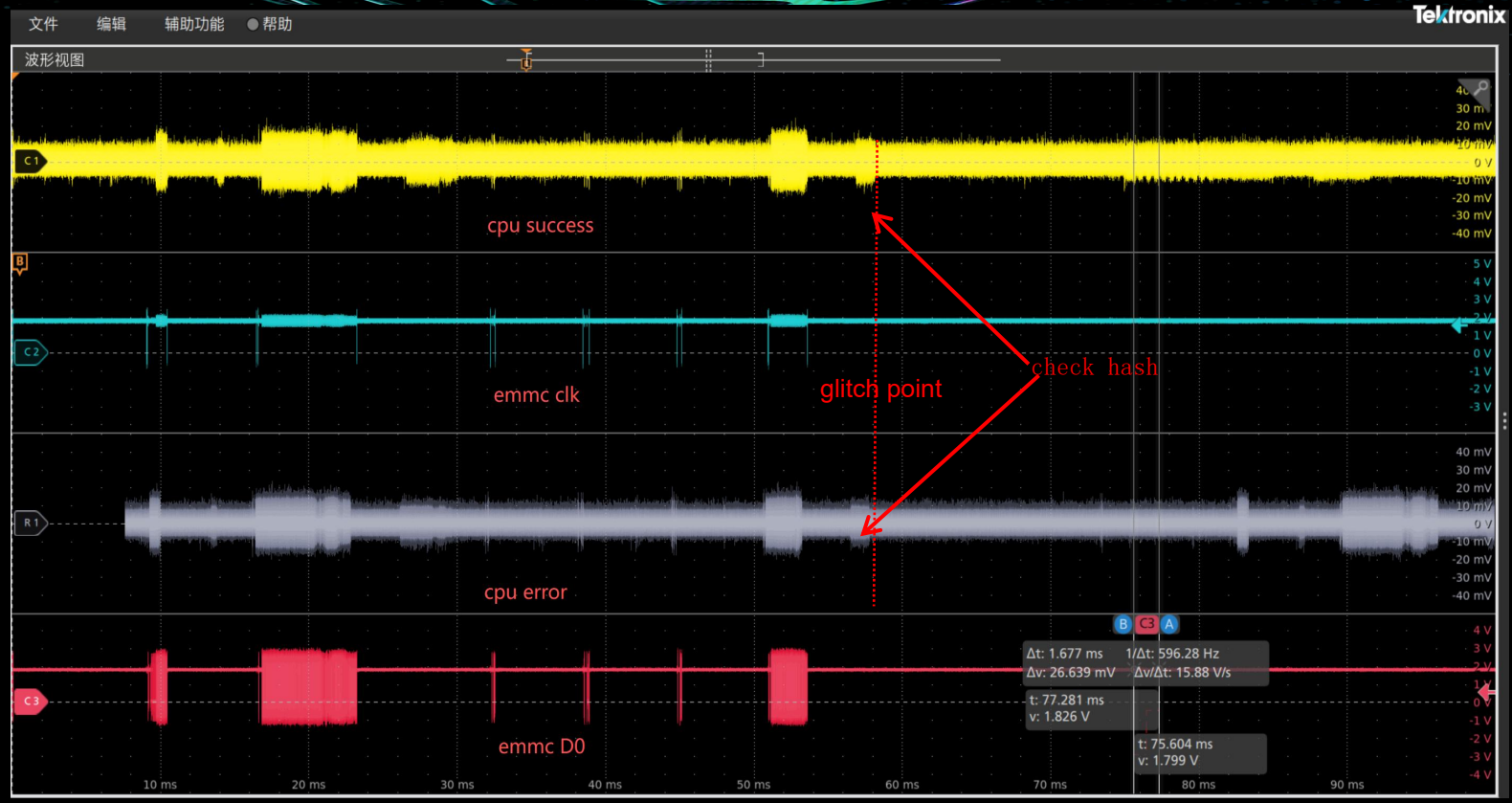
- CPU and D0 show clear synchronization during data verification.
- Obvious signal features occur when reading ATF-1/uboot/FDT/ATF-2/ATF-3/optee.
- Data reading is followed by hash checksum verification of the block.
- Glitch points are located by matching D0 and EM signal characteristics.



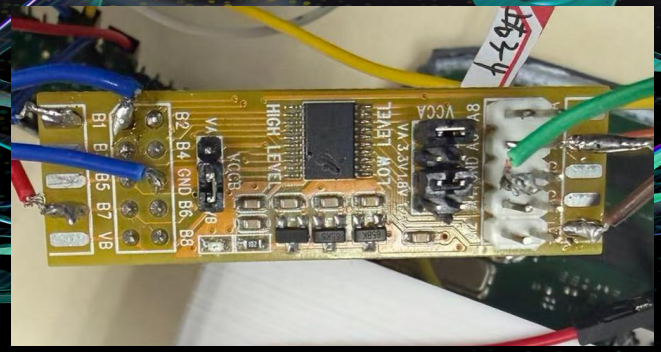


4.2 Narrowing the Attack Window Further

- Early CPU boot stage,High-frequency electromagnetic leakage：
  - eMMC data reading
  - Cryptographic algorithm calculation
- The difference between the two types of signals:
  - EM signals from eMMC data reading correspond to data fetch signals on the D0 pin.
  - EM signals from cryptographic computation show no toggling on the D0 pin.
- Signature verification typically occurs immediately after cryptographic computation.

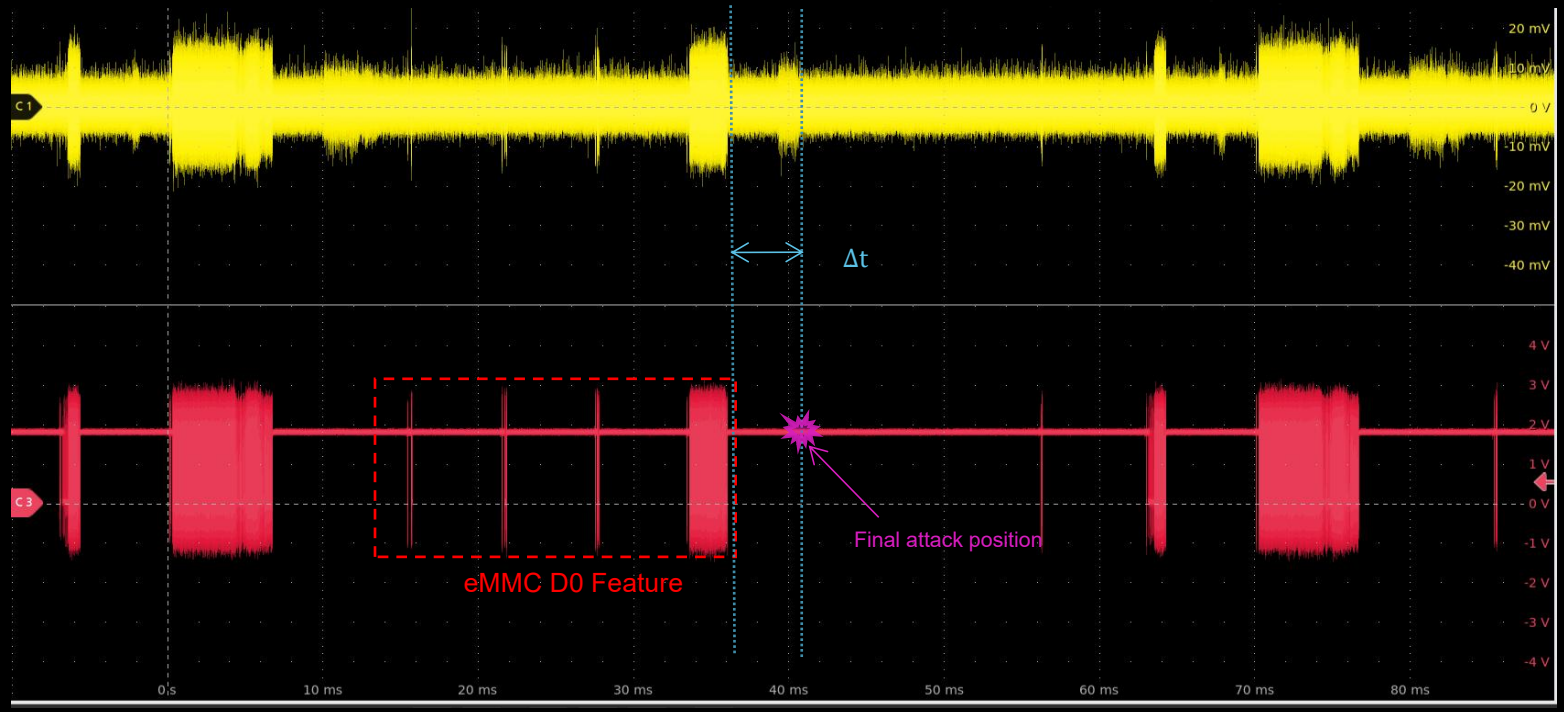


4.3 Detailed Fault Attack Procedure

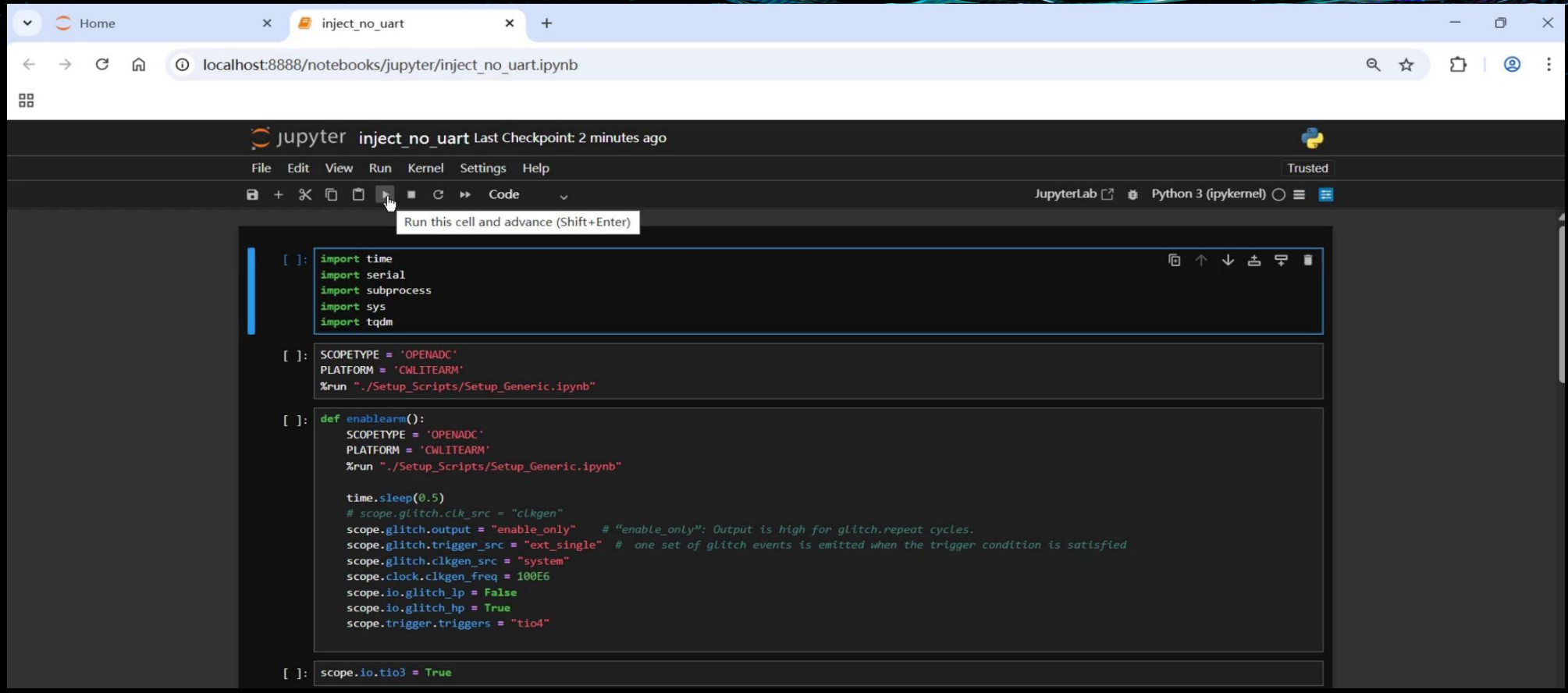


- Convert the eMMC D0 signal to a CW-compatible I/O level via a level shifter.
- Use three short pulses plus one wide pulse on the D0 signal, and calculate the approximate pulse width as the trigger point.
- Based on the end of the last eMMC D0 toggle, add the relative time of hash computation as the trigger offset.

$$\text{glitchpoint} = \text{lastD0} + \Delta t$$



4.4 Video of Successful Fault Attack with Serial Port Disabled



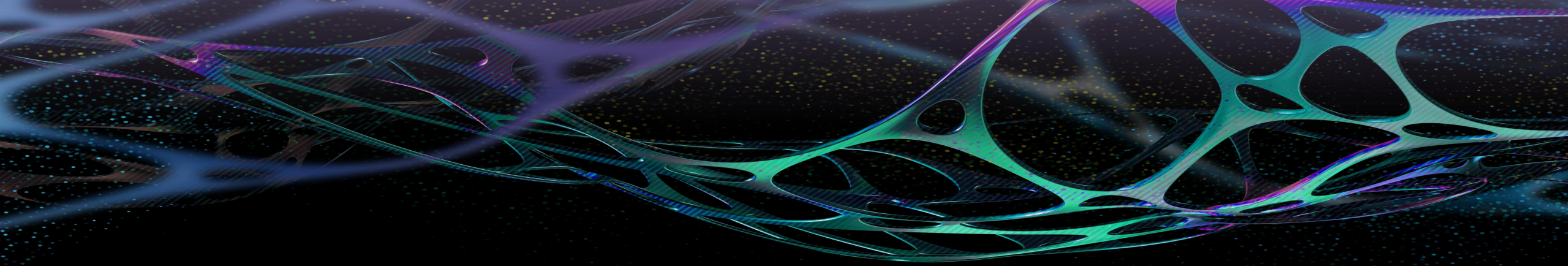
Conclusion

- Cross-domain hardware signal correlation analysis method.
- By combining CPU EM micro-features and eMMC timing, it precisely locates Secure Boot verification in a serial-free environment.
- The fault attack window is compressed from second scale to millisecond scale.
- Repeatable and targeted precise fault attacks are realized.

https://github.com/xcatx9527/wfm_cmp.git

Future Work

- Algorithm optimization
- Multi-signal fusion
- Defense mechanism research



Thanks!