



COSIC

Computer Security and Industrial Cryptography

JAIF 2026
Benedikt Gierlichs &
Lennert Woutes

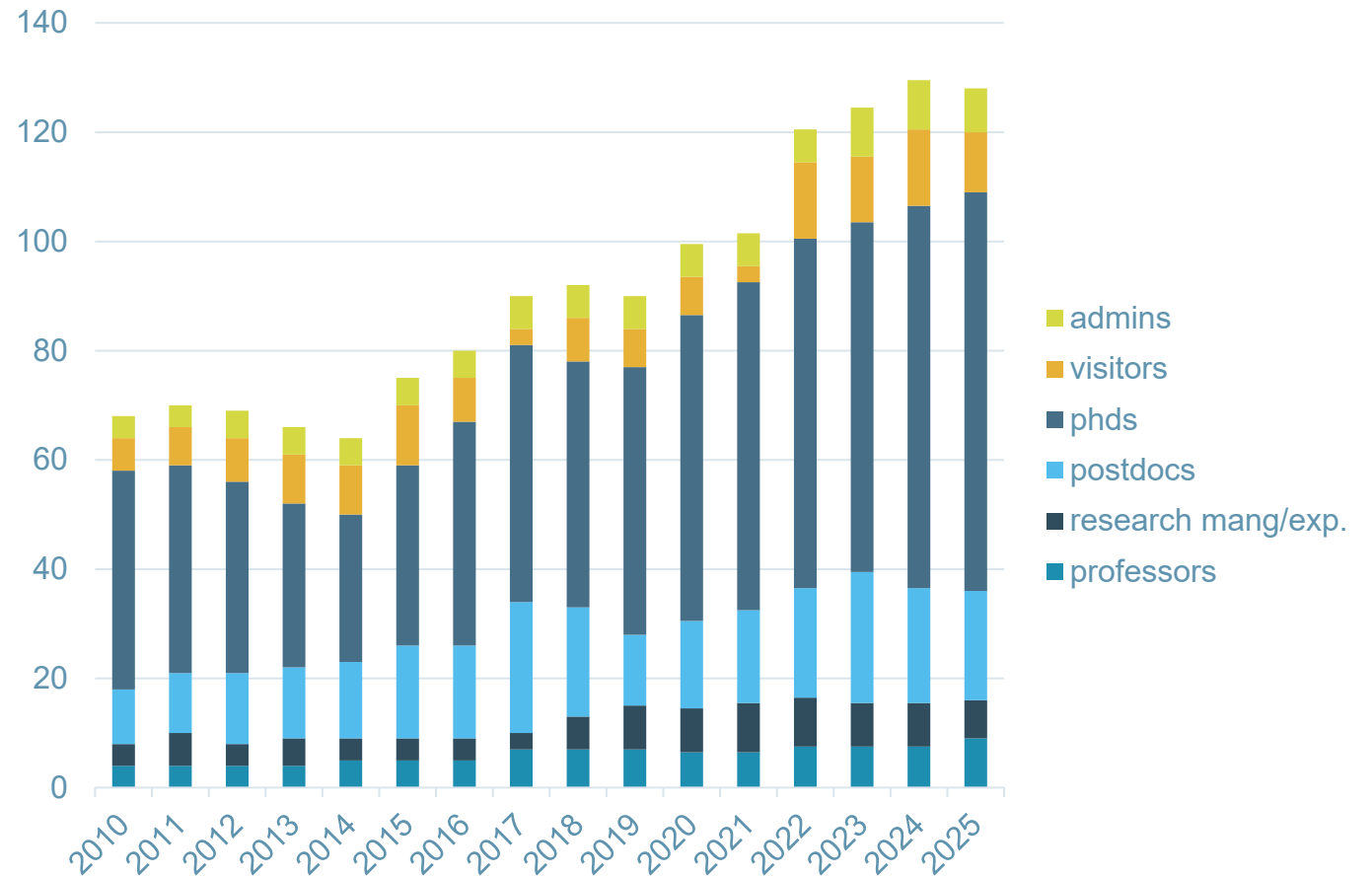
In short

- **C**OMputer **S**ecurity and **I**ndustrial **C**ryptography
- Established in 1978
- A research group within the Department Electrical Engineering (ESAT) at KU Leuven
- Broad expertise in digital security and innovative security solutions
- COSIC is headed by Professor Bart Preneel

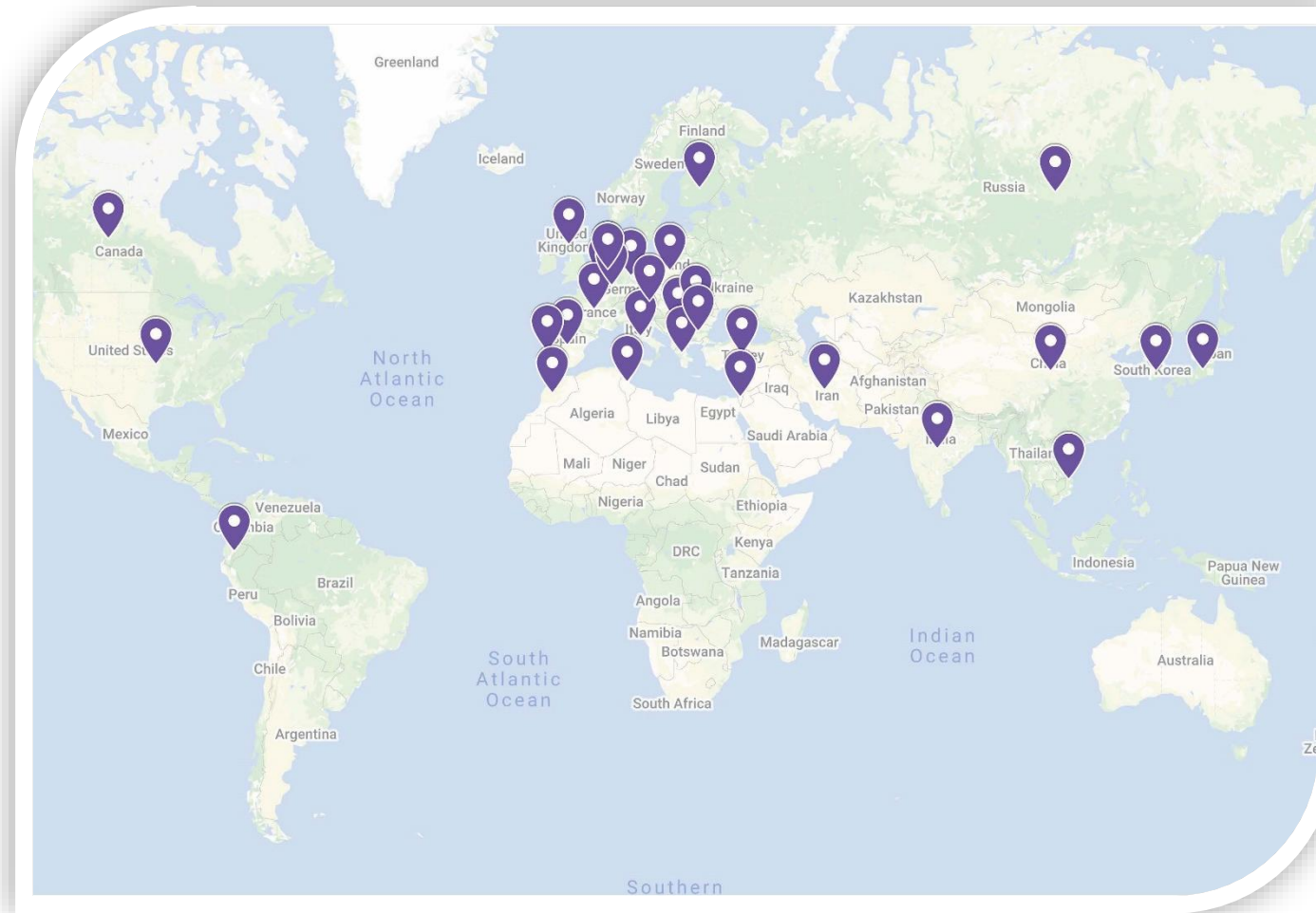


Largest applied cryptography research group worldwide

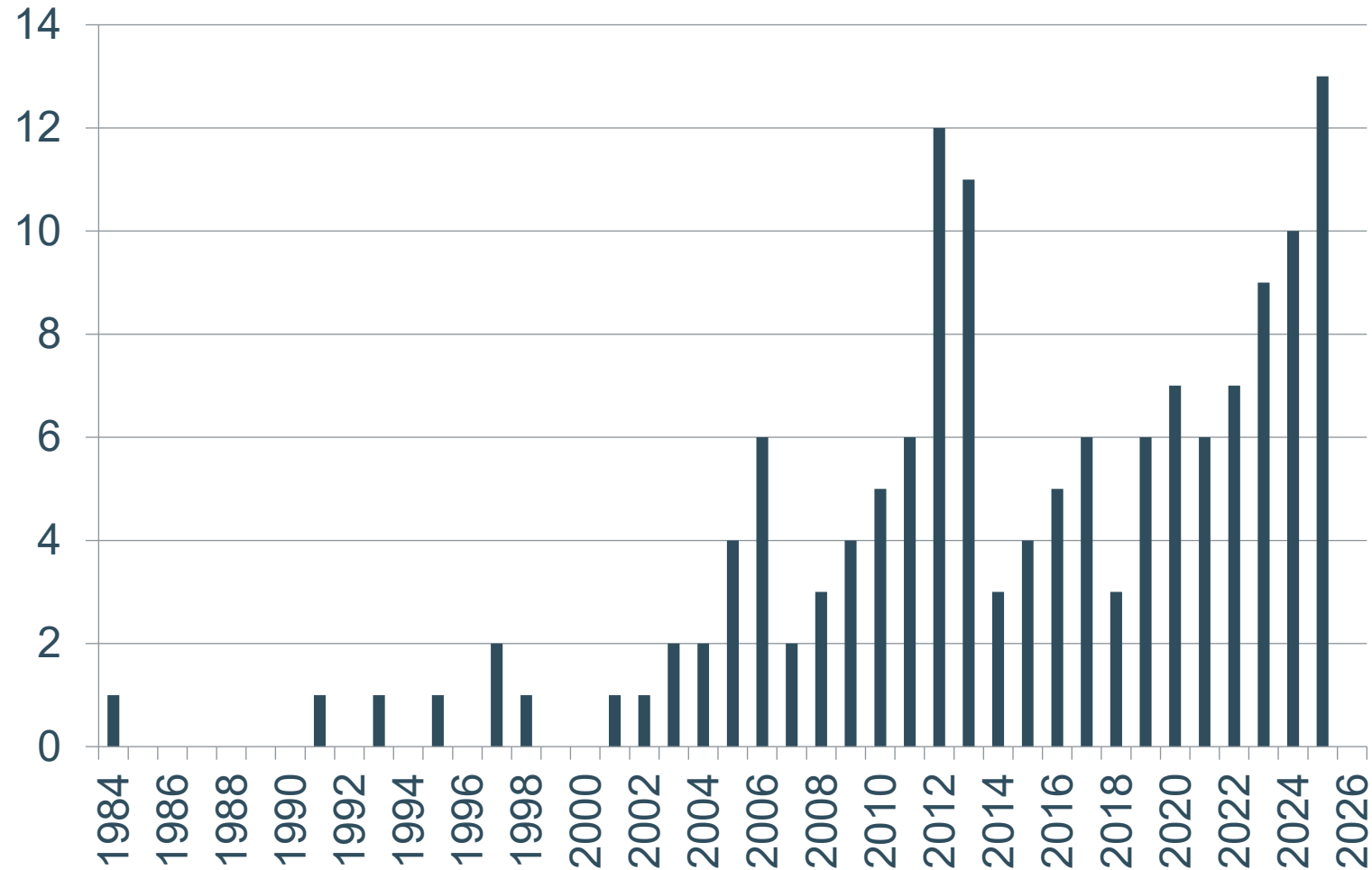
- 7+2 professors
- 7 research experts
- 25 postdocs
- 62 PhD students
- 8 support staff
- 10+ visitors/interns



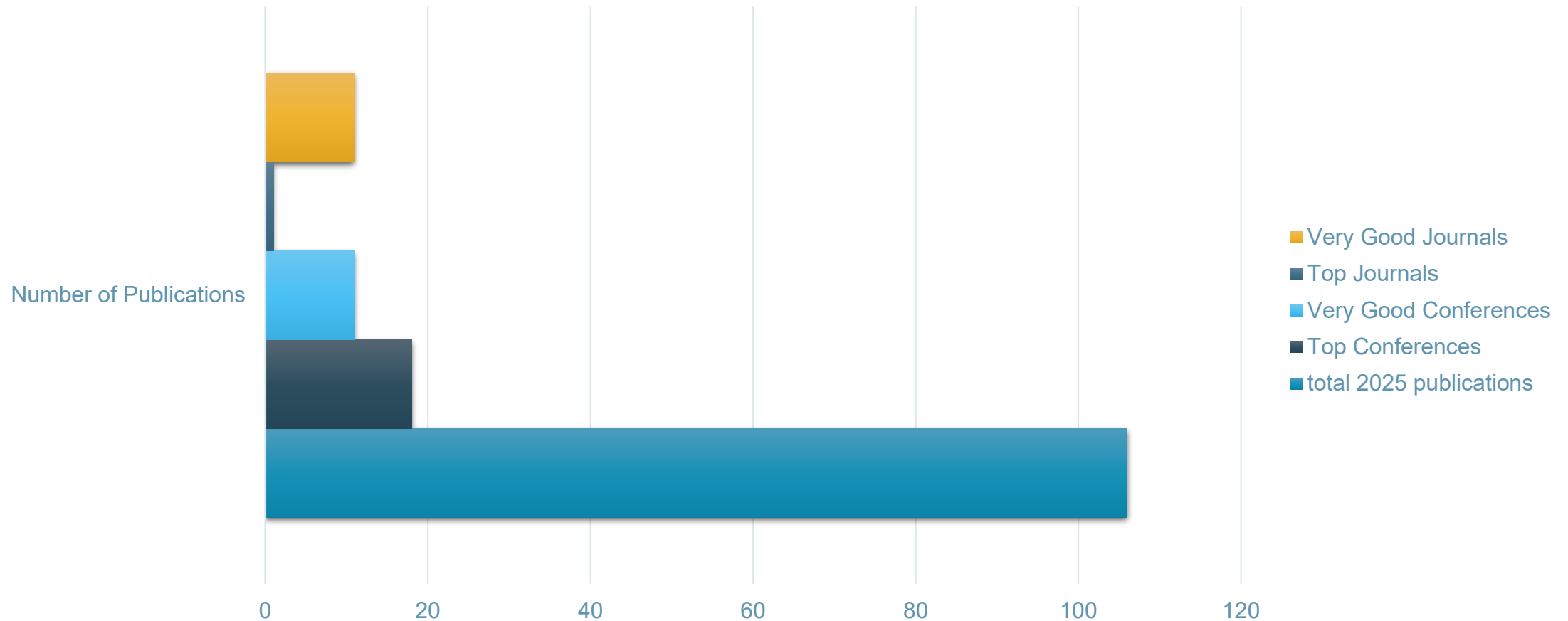
COSIC international members



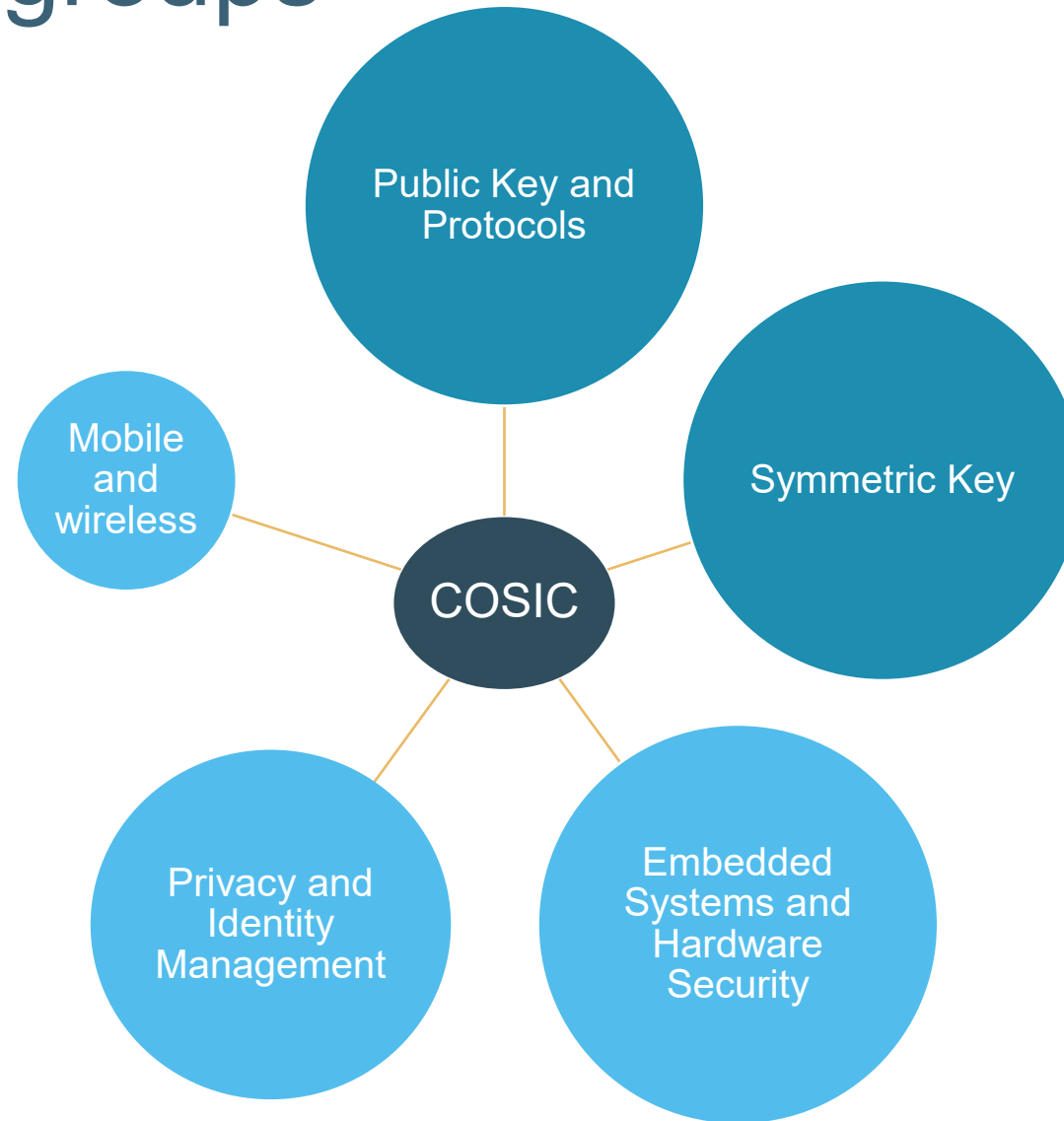
More than 90 graduated PhD students



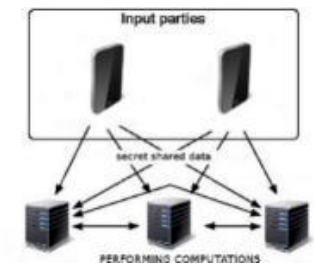
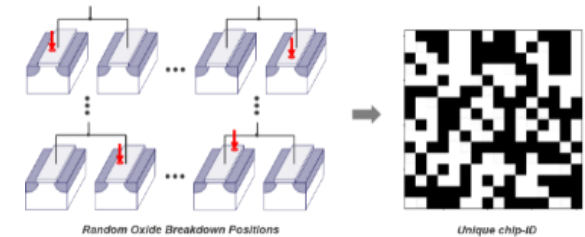
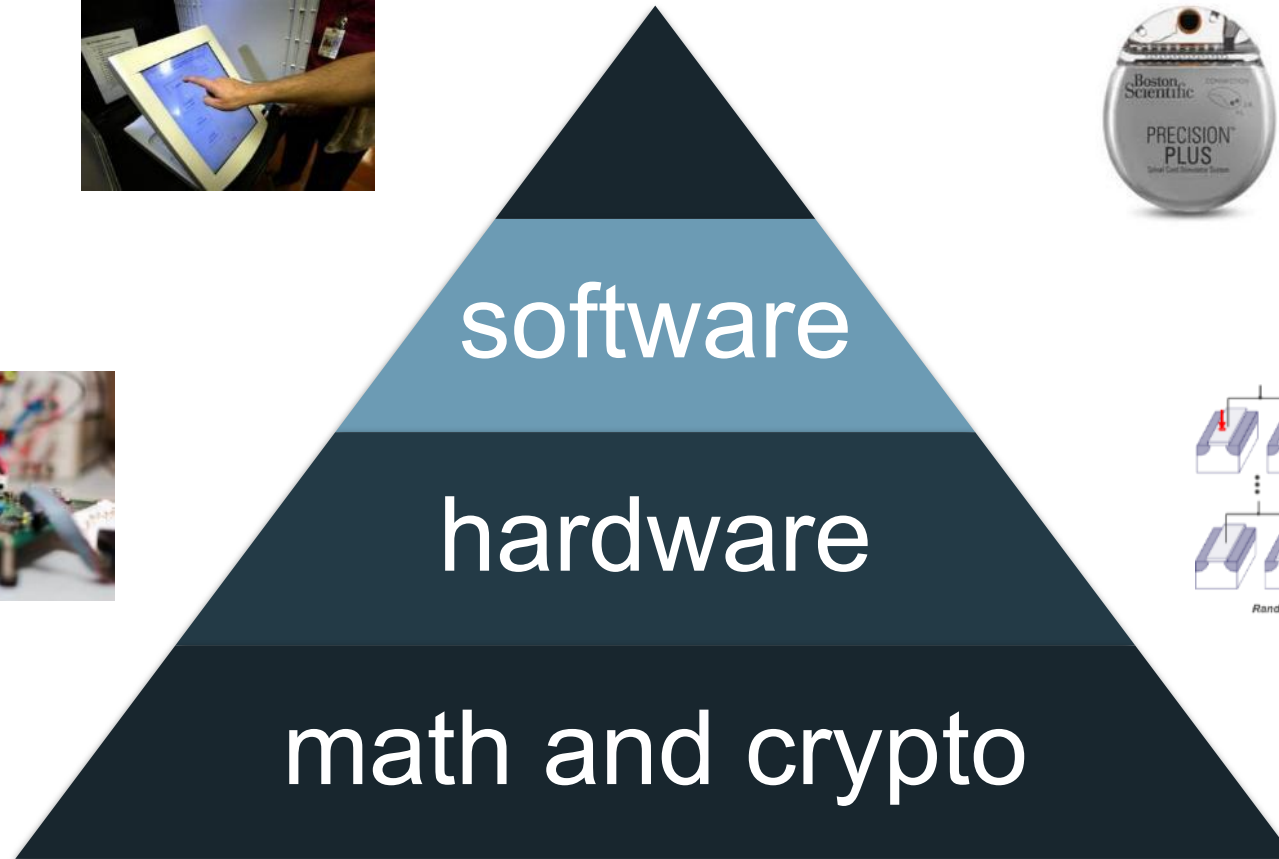
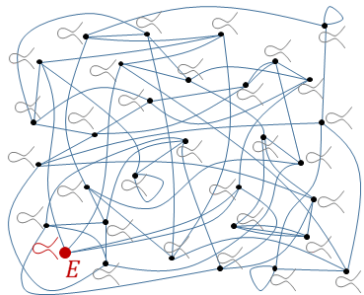
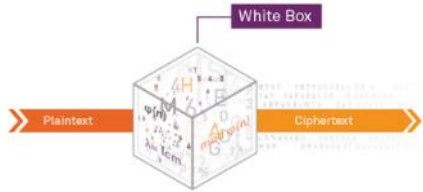
Total publications 2025 in COSIC Database



COSIC - Subgroups



Transversal research



COSIC Research

Efficient and secure implementations: FPGA and ASIC, Side-channel attacks: power, timing, EM analysis, fault attacks

Post-quantum cryptography

Computing on Encrypted data: MPC, FHE, ZK

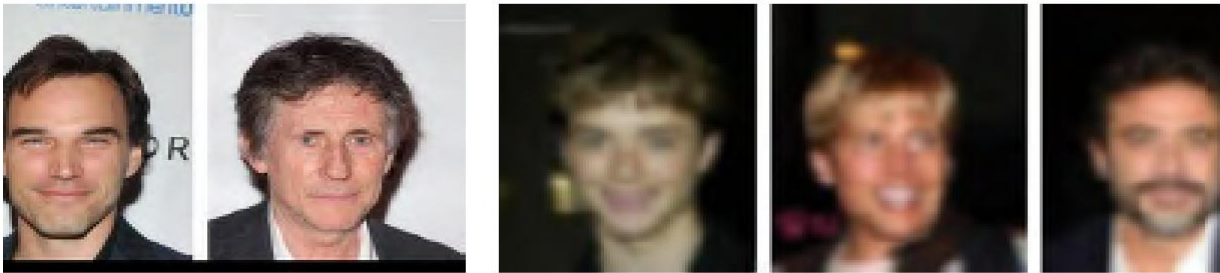
Cryptographic protocols: entity authentication, anonymous credentials, oblivious transfer

Cryptographic algorithms: AES, RIPEMD-160, Whirlpool, Trivium, FAEST

Number theoretic algorithms, Boolean functions, secret sharing, secure multi-party computation

Perceptual hashing: false positives

Detection of copyright violation and
CSAM (Child Sexual Abuse Material)

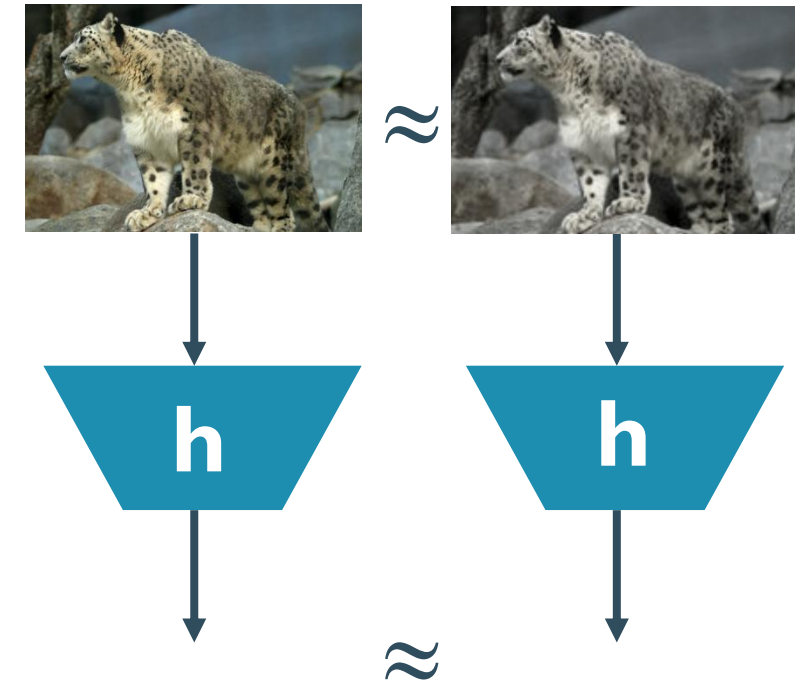


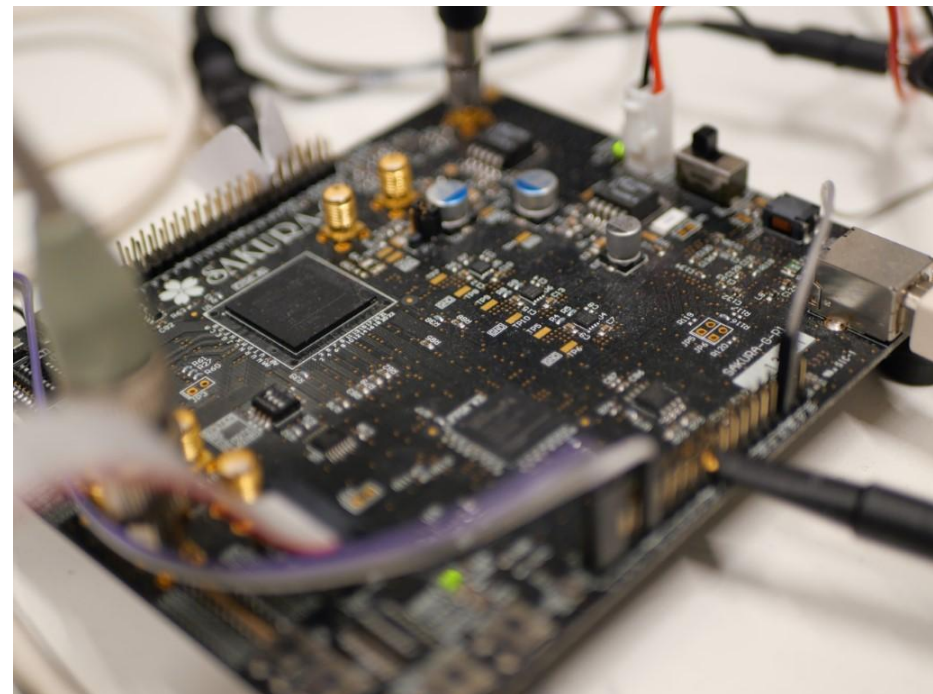
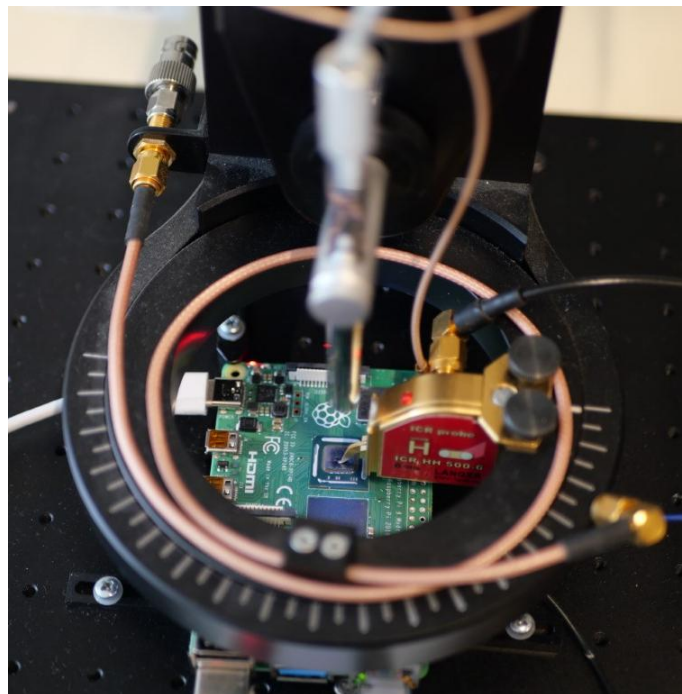
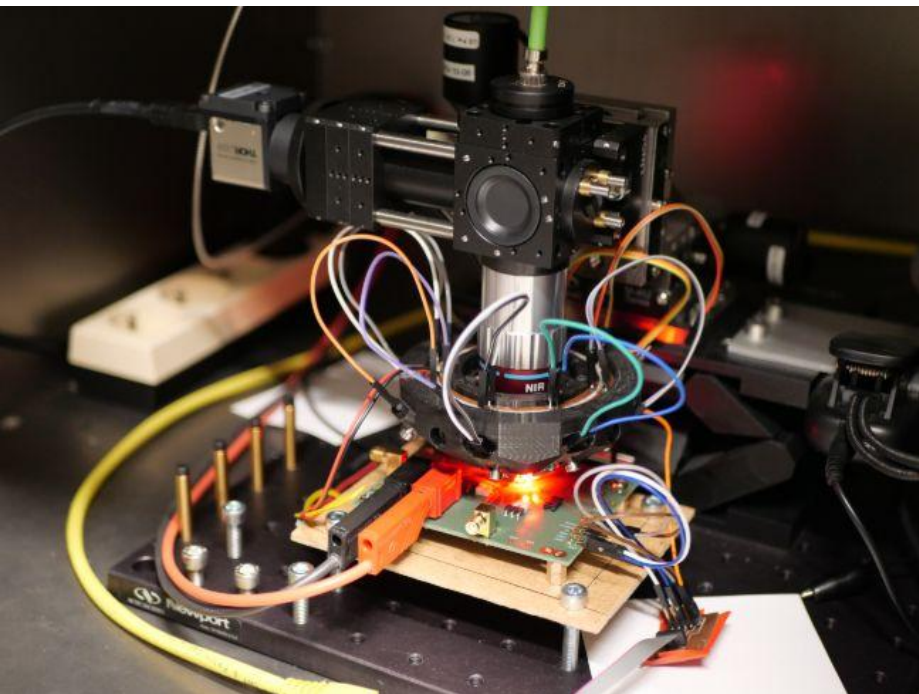
(a) Non BF

(b) Light BF

[NCMEC, Thorn] Current CSAM databases contain 17.5 to 19 million images
EU citizen has taken on average 597 selfies per year (2019)
Assume that a citizen is reported only after x alerts

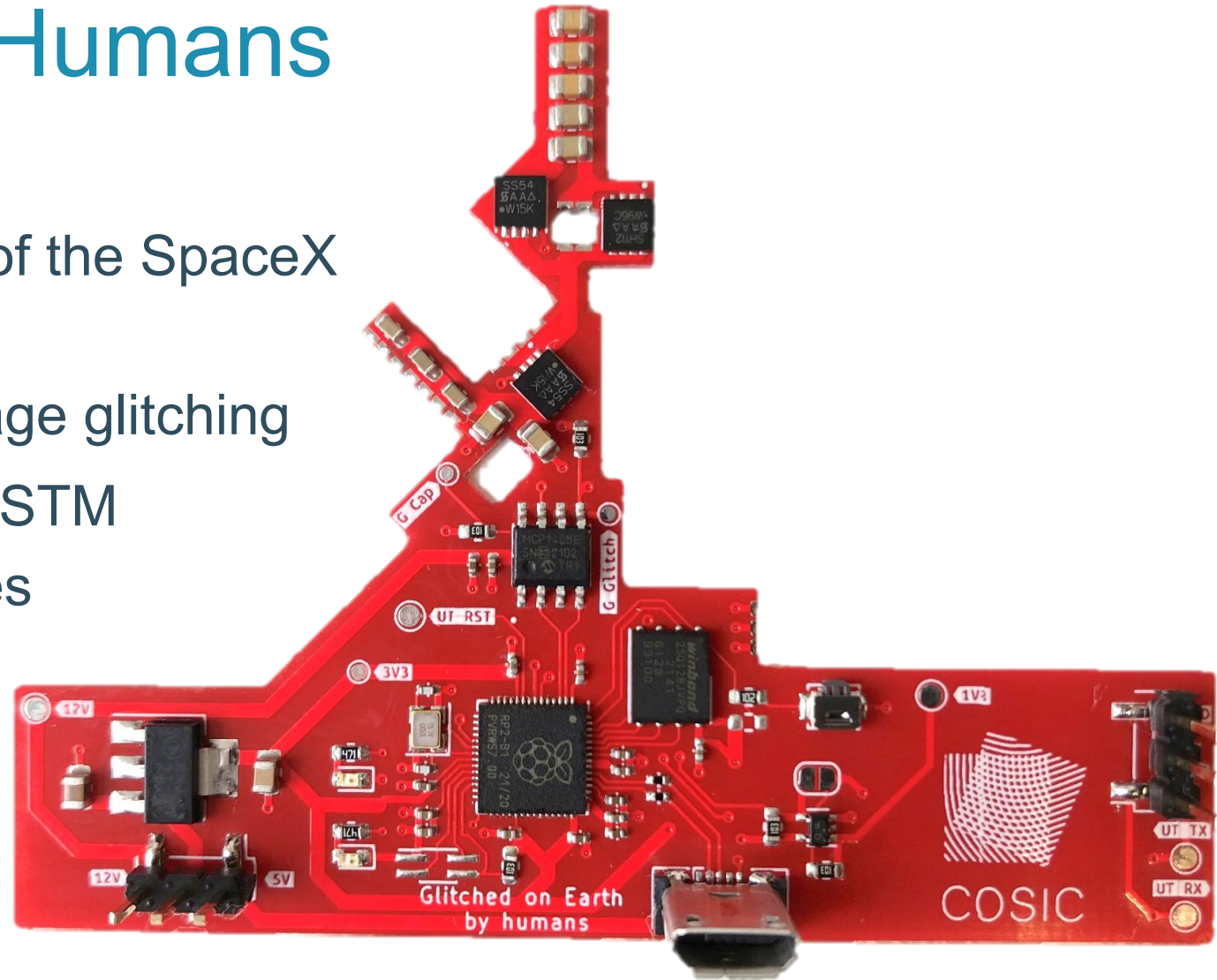
# alerts x before reporting	1	2	3	5	8	10
# falsely reported citizens reported in year 1	406M	304M	185M	39M	1.2M	69K





Glitched on Earth by Humans

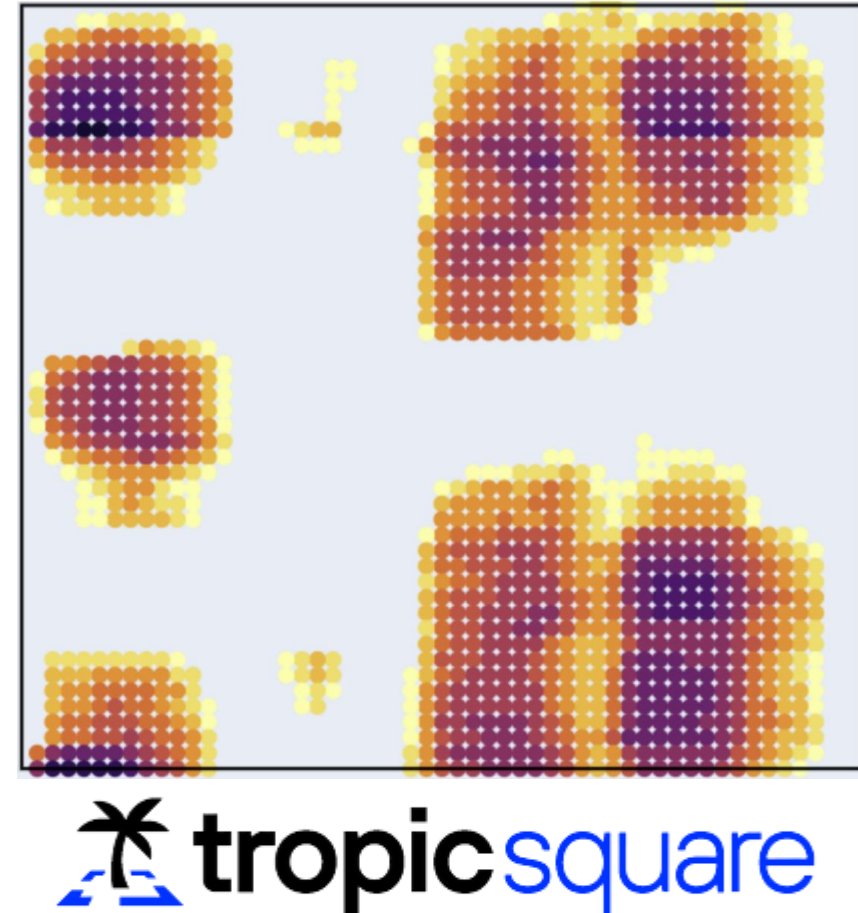
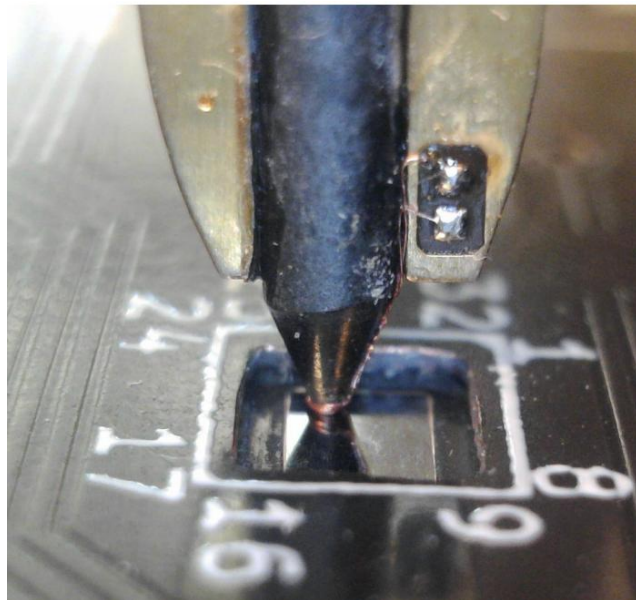
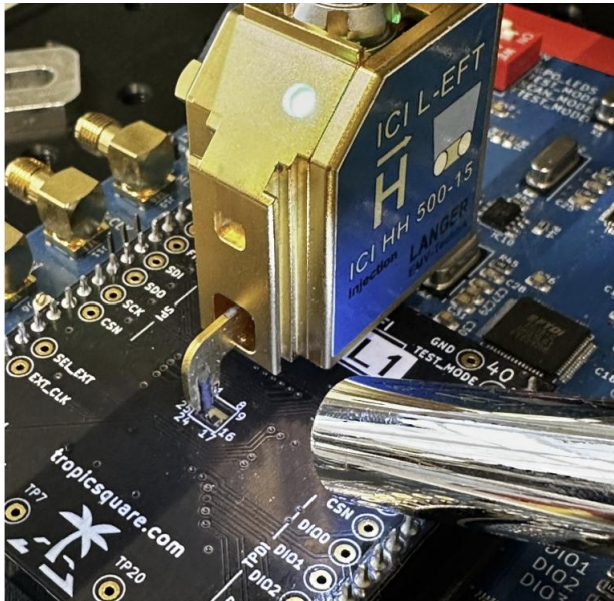
- A Black-Box Security Evaluation of the SpaceX Starlink User Terminal
- Secure boot bypass through voltage glitching
- Custom quad core SoC made by STM
 - With some FI countermeasures
- Custom open-source modchip
- Black Hat USA 2022:
<https://youtu.be/NXqLMmGwJm0>



github.com/KULeuven-COSIC/Starlink-FI

Fault injection evaluation of TROPIC01

- Collaboration between Tropic Square and KU Leuven
- Evaluating resistance against VFI, EMFI, and LFI
- Security sensor characterization
- Detailed talk during OPTIMIST @ CHES 2026

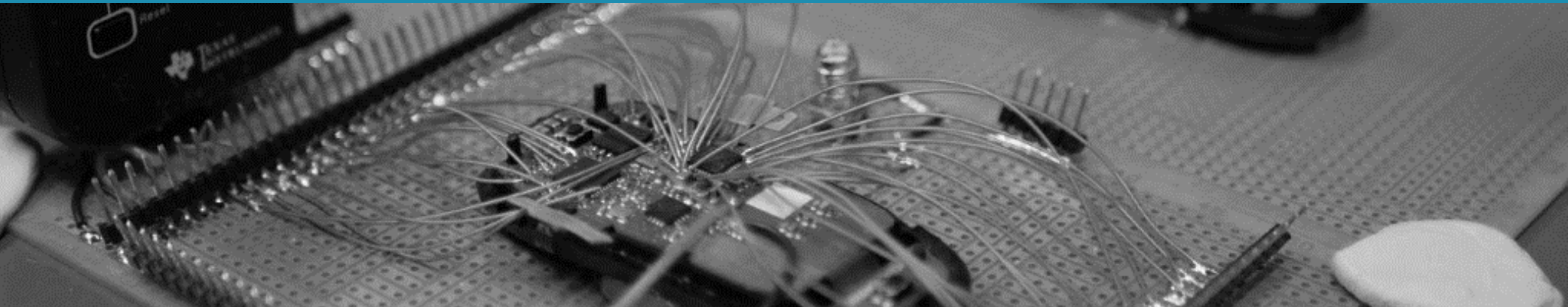


Failure is Not an Option:

A Reliable Process to Exploit STM32F2/F4 Microcontrollers

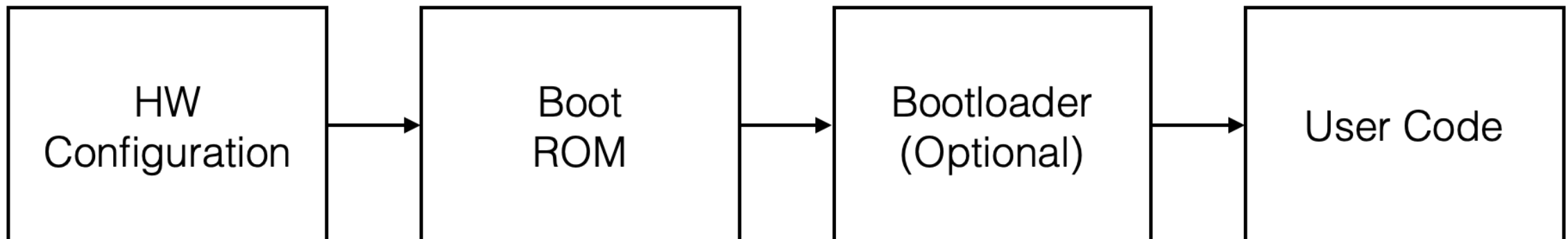
Lennert Wouters

Joe Grand



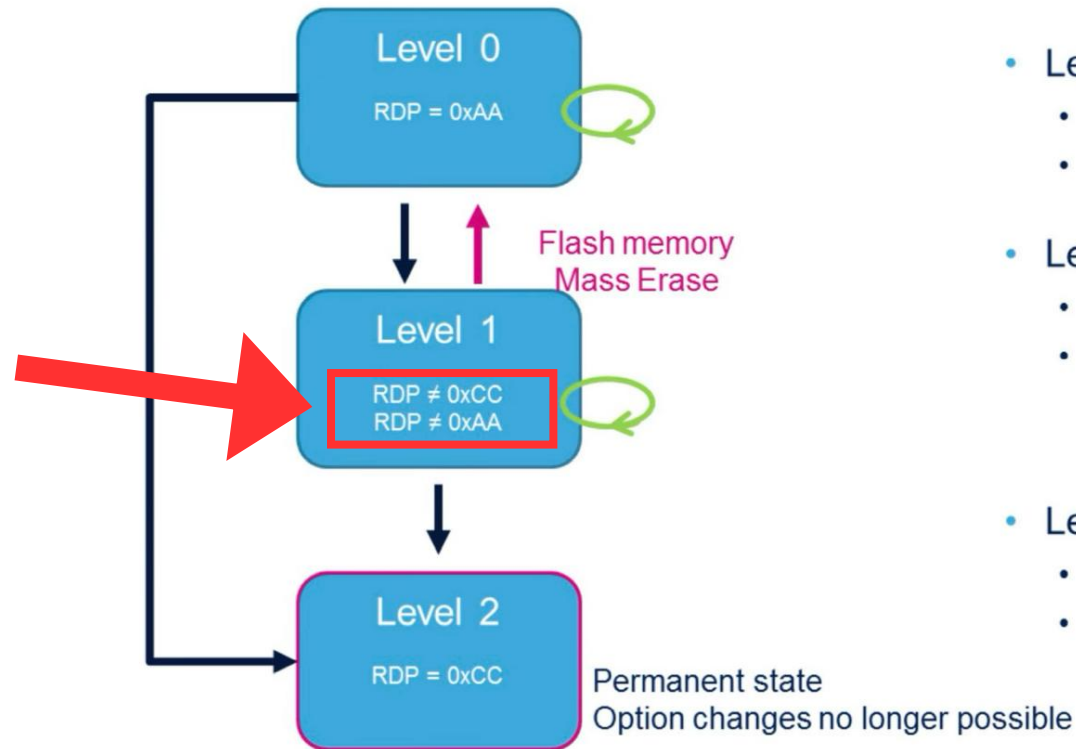
MICROCONTROLLER (MCU) SECURITY 101

- MCUs have internal memories that may contain secrets (e.g. firmware / IP or key material)
- Vendor-specific implementations try to prevent adversaries from reading those secrets
 - May require fuse/register setting, password, challenge/response
 - Reduce access (allow subset of functionality)
 - "Permanently" disable access
- Configured/checked during chip boot process



RDP transition scheme

10



- Level 0
 - Option byte mods are allowed
 - Can transition to Level 1 or Level 2
- Level 1
 - Option byte mods are allowed.
 - Can transition to Level 0 or Level 2
 - Level 0 → Mass erase of user Flash memory, backup regs and SRAM2
- Level 2
 - Option bytes are frozen
 - No transition possible

STM32 BOOT MODES

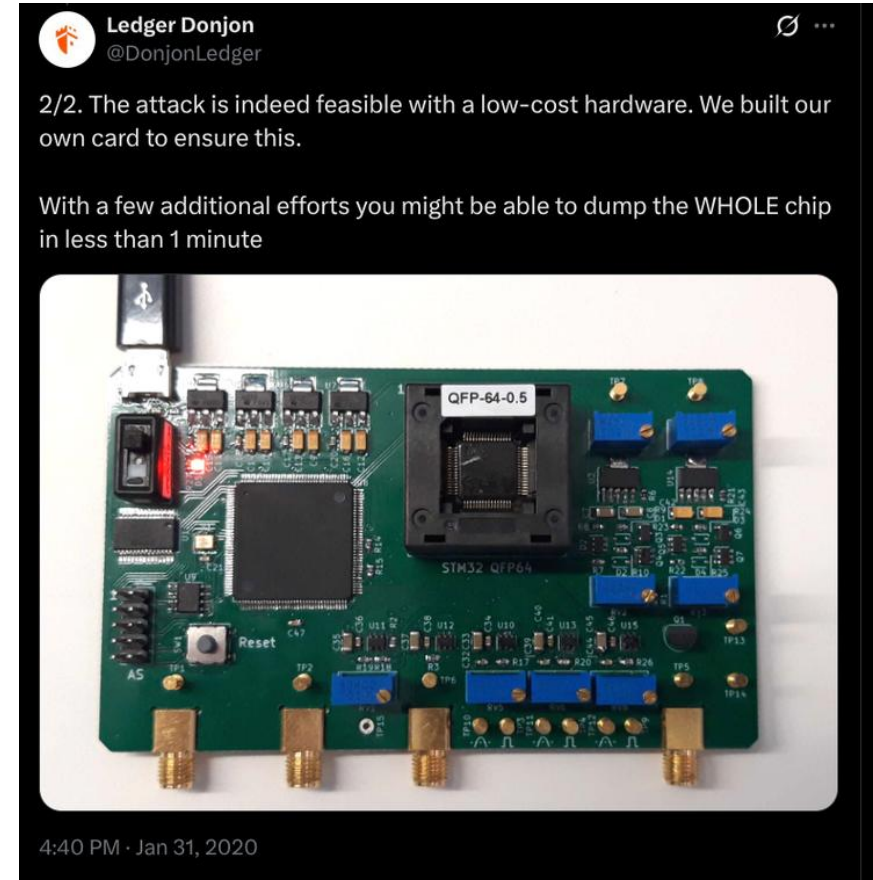
- [RM0033 STM32F205xx Reference Manual](#)
- The main flash memory boot mode is the only available option if the chip is in RDP2

Table 4. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

EXISTING ATTACK TECHNIQUES

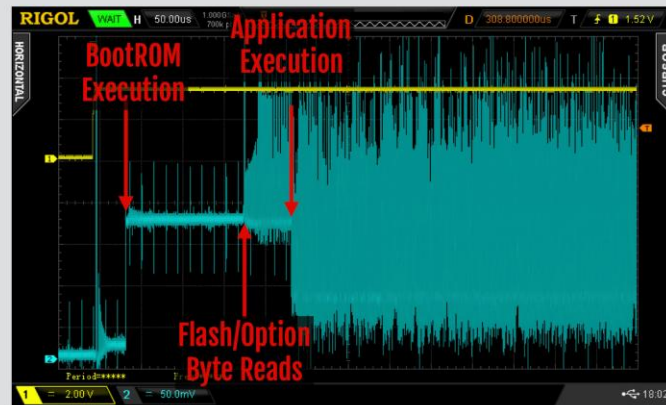
- RDP Downgrade
 - [wallet.fail](#) / [chip.fail](#) (2018/2019)
- Bootloader Read Memory Glitch (STM32F1)
 - [Shaping the Glitch](#), Bozzato, et al. (2019)
 - [Prog.World](#) (2020)
- Combined
 - [Kraken Security Labs](#) (2019)
 - [Wrongbaud](#) (Downgrade + Bootloader Go) (2022)
- Other
 - [ROM bootloader USB DFU bug](#), Goodspeed (2012)
 - [Ledger unknown attack technique](#) (2020)
 - [FPB \(Flash Patch and Breakpoint\)](#), Obermaier, et al. (2020)
 - [Flash erase suppression](#), Schink, et al. (2024)



PROBLEMS W/ EXISTING TECHNIQUES

- Can lead to device corruption or permanent loss of data
- We weren't successful in reproducing some attacks
- Some attacks did not have any public details
- Previous research does not cover a large enough sample size to account for device variability when performing fault injection on multiple chips

Power consumption after reset (200 μ s)



WISH LIST

- Complete Flash extraction from an RDP2-secured MCU
- Near-zero risk of unintentional corruption or data loss
- Reliable and repeatable
- Limit the number and type of faults
- Use accessible/available hardware
- No semi-invasive or invasive attacks

EMFI

CNC 3018 Pro Max

ChipSHOUTER

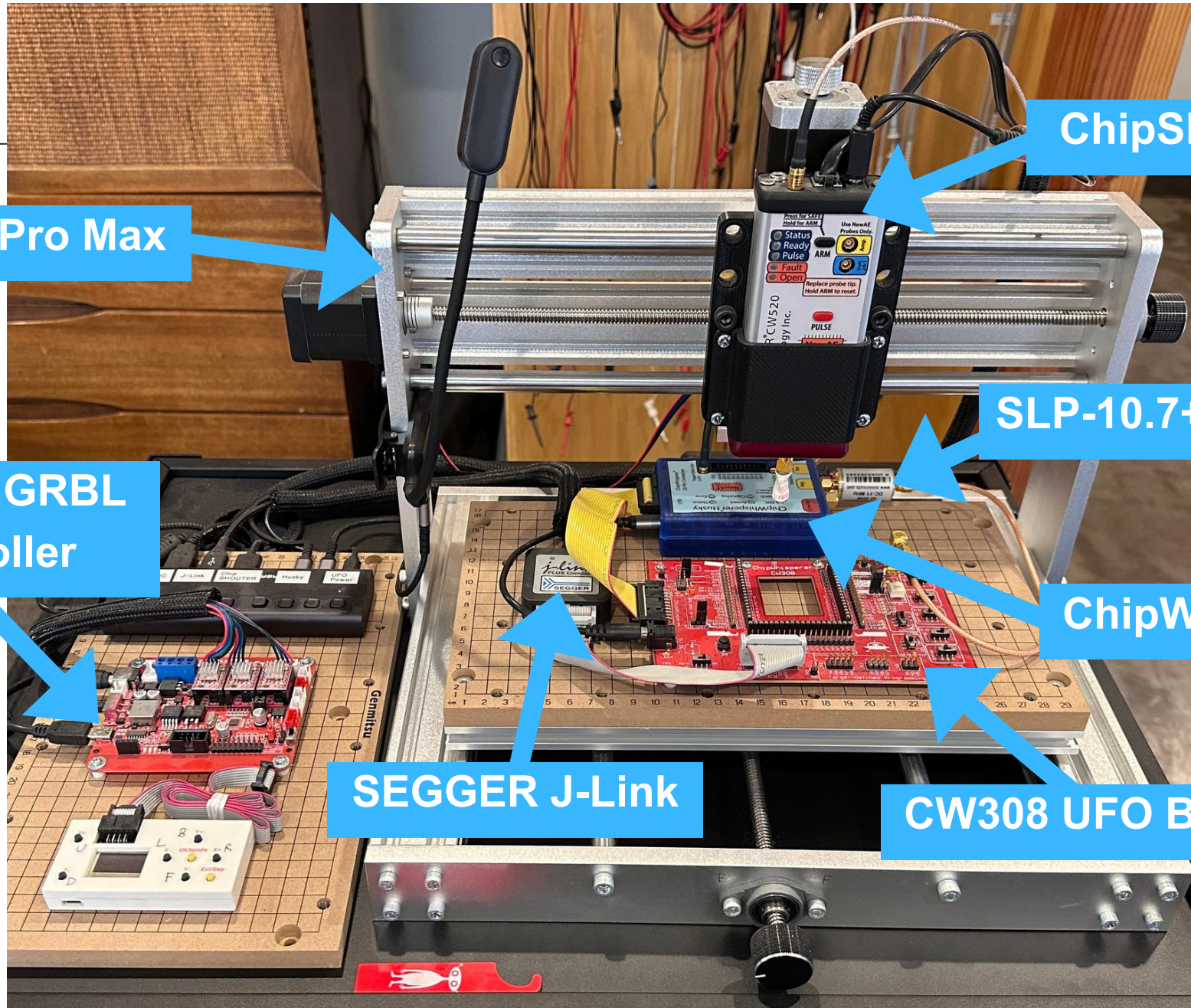
Doesbot GRBL
Controller

SLP-10.7+ Low Pass Filter

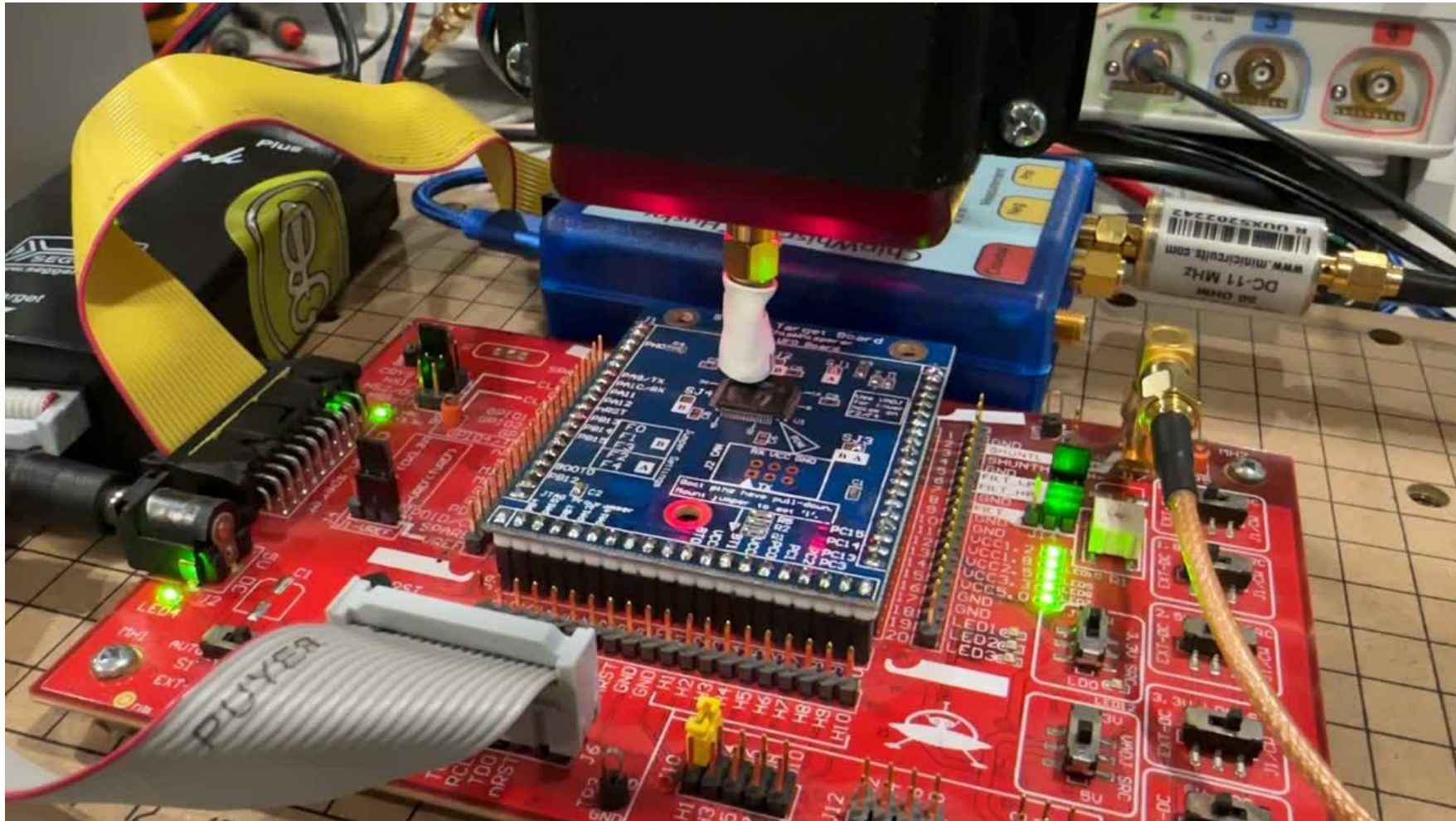
ChipWhisperer Husky

SEGGER J-Link

CW308 UFO Board



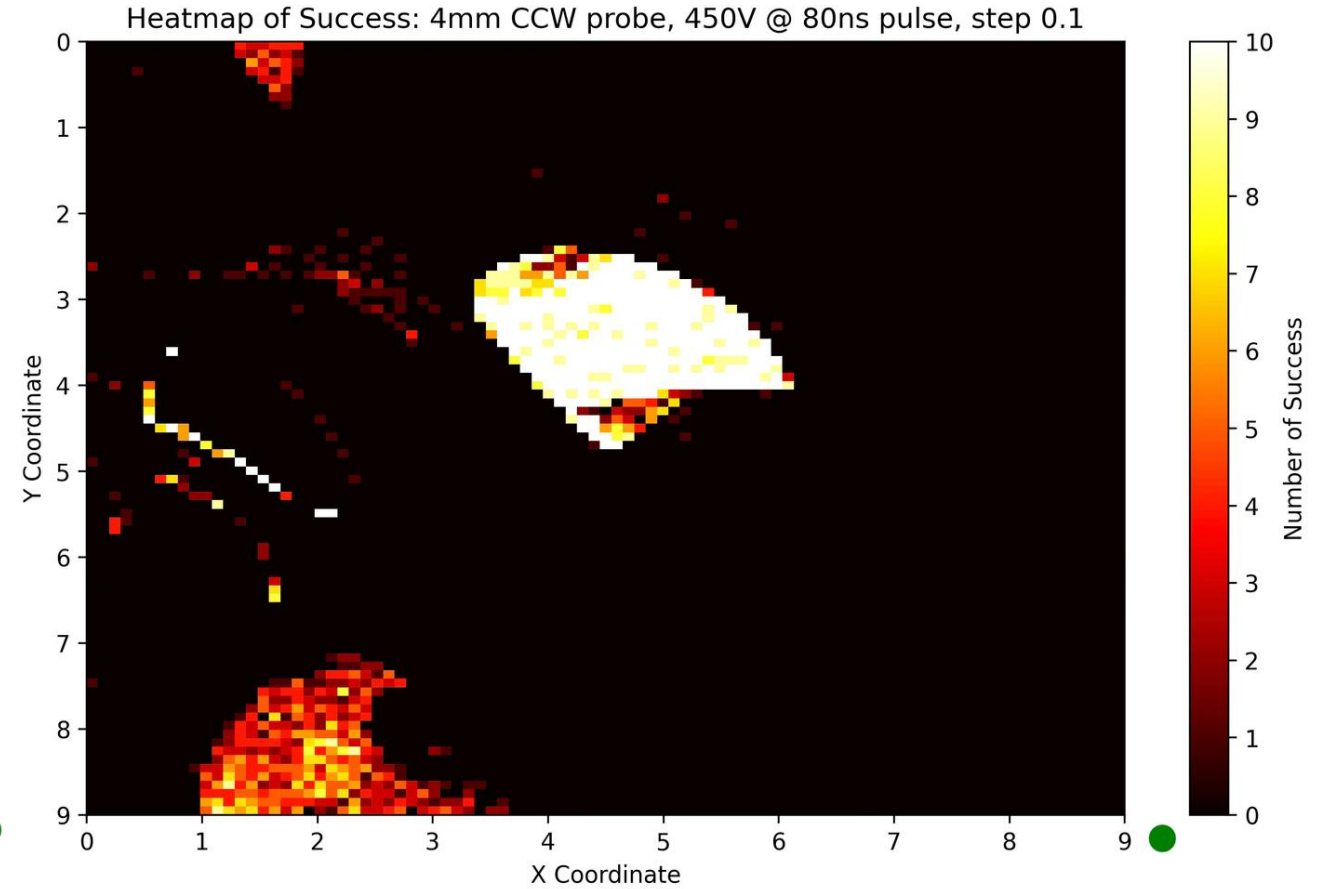
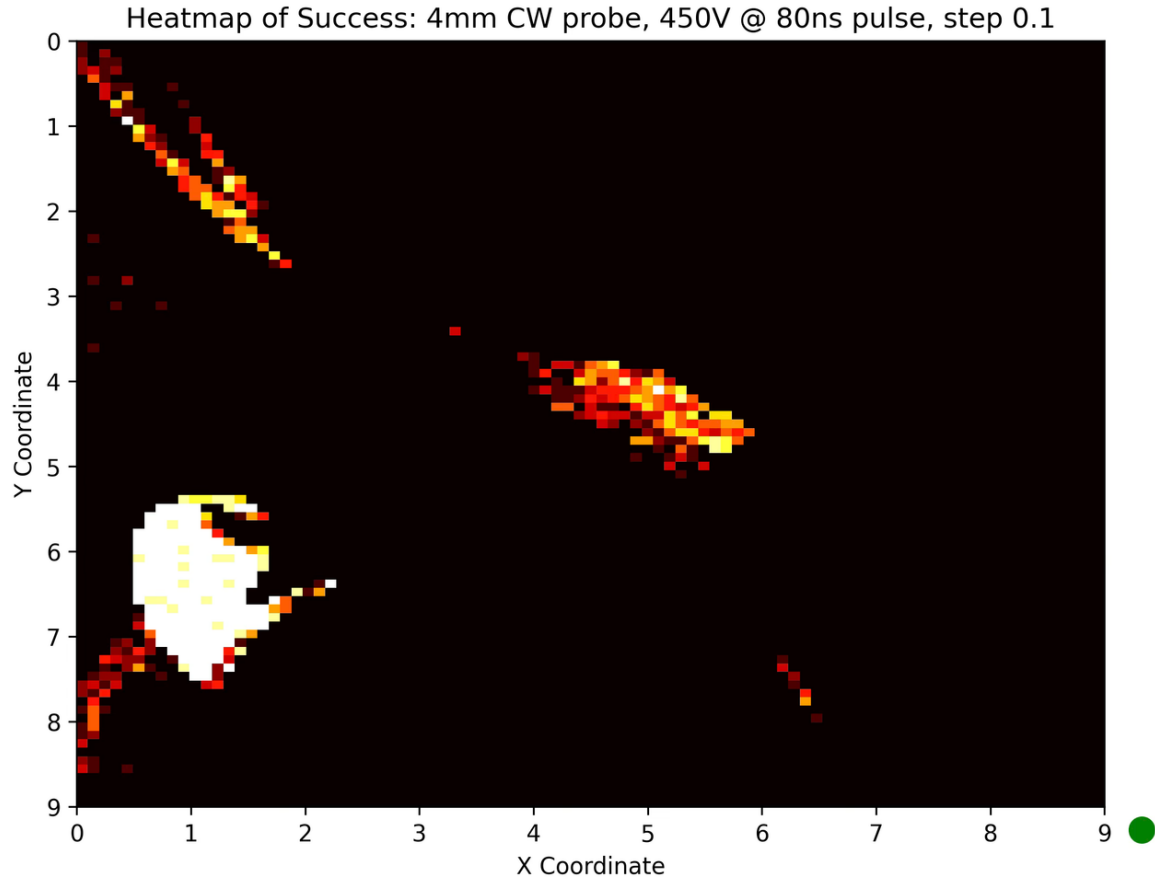
- Identify vulnerable areas of the chip



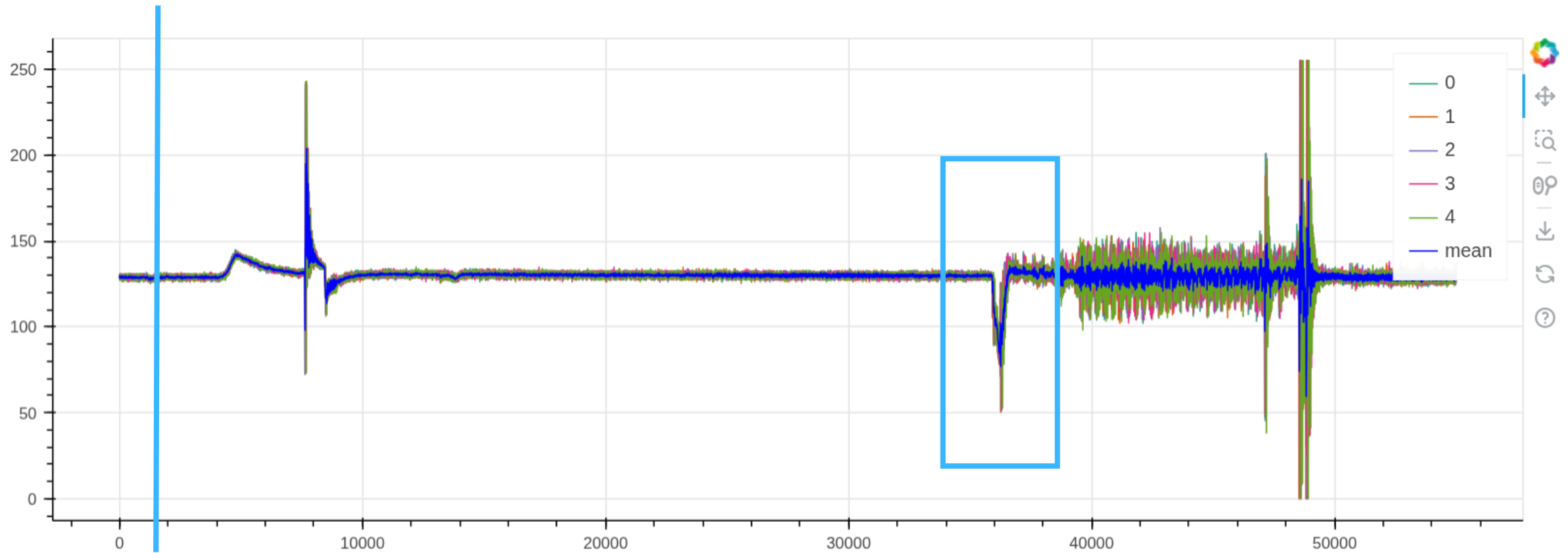
DEVICE CHARACTERIZATION

- Goal is to determine parameters across target device
 - Probe type, probe position, glitch settings (voltage, offset, trigger)
- Different parameters can cause different results
 - No change, reset, option byte corruption, permanent downgrade, Flash wipe/erase
- Use test code first to ensure device is vulnerable, then increase to specific attacks
- Can be manual or automated positioning
- Visualize results w/ plots/graphs

F2 CHARACTERIZATION: TEST CODE

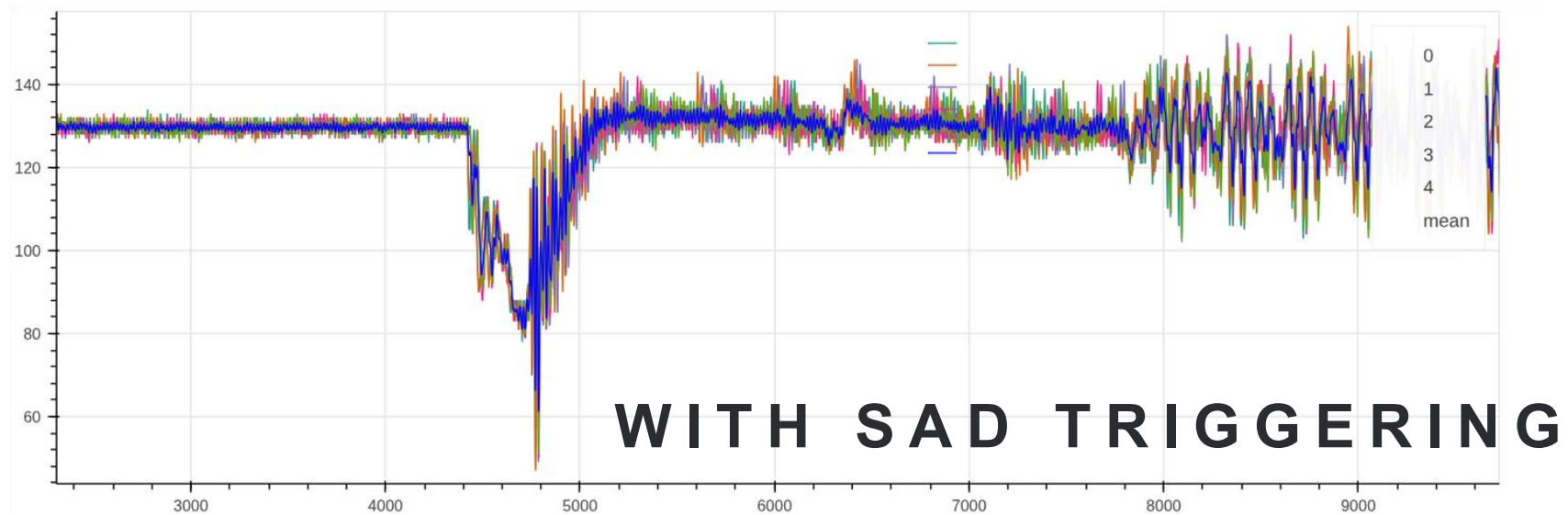


F2 RDP DOWNGRADE: POWER ANALYSIS

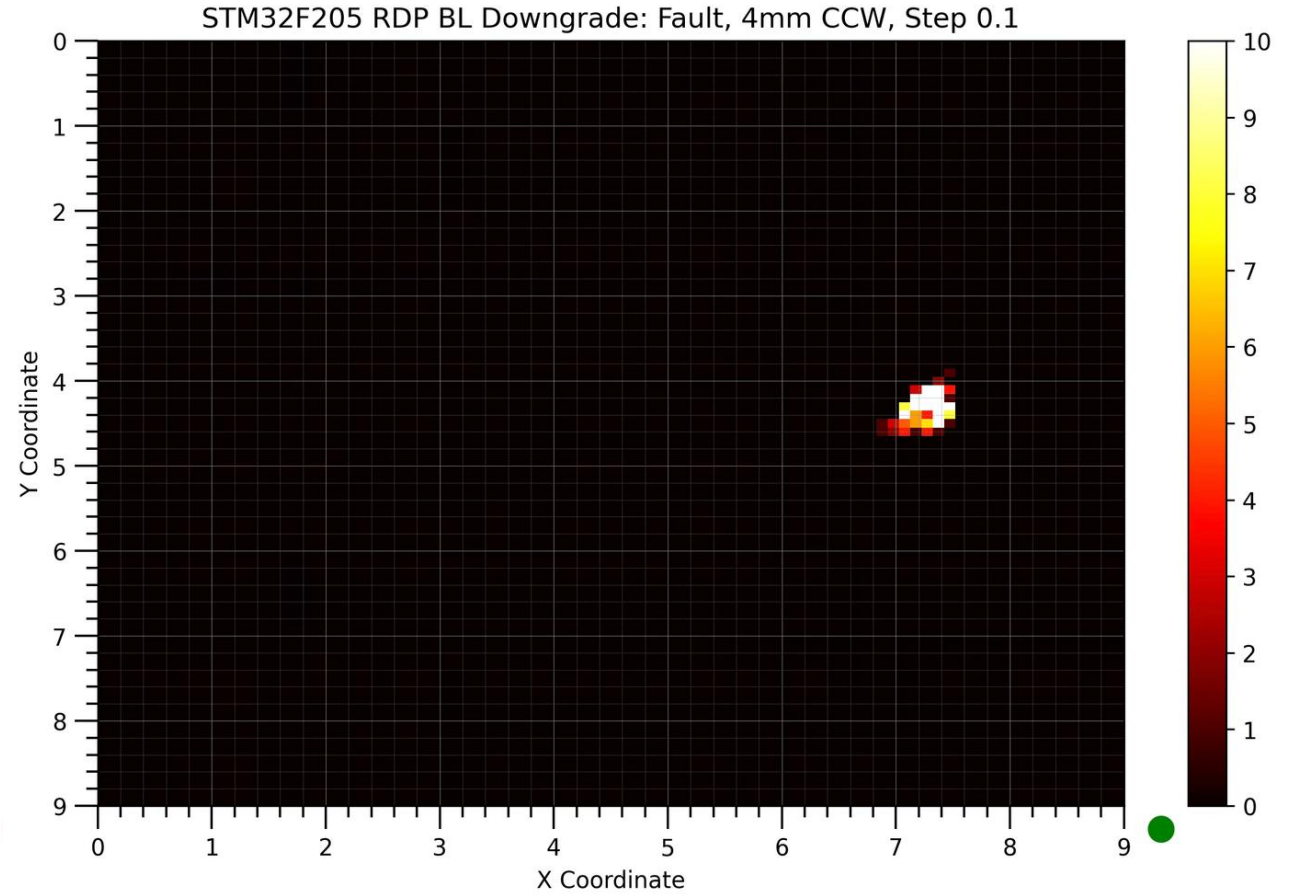
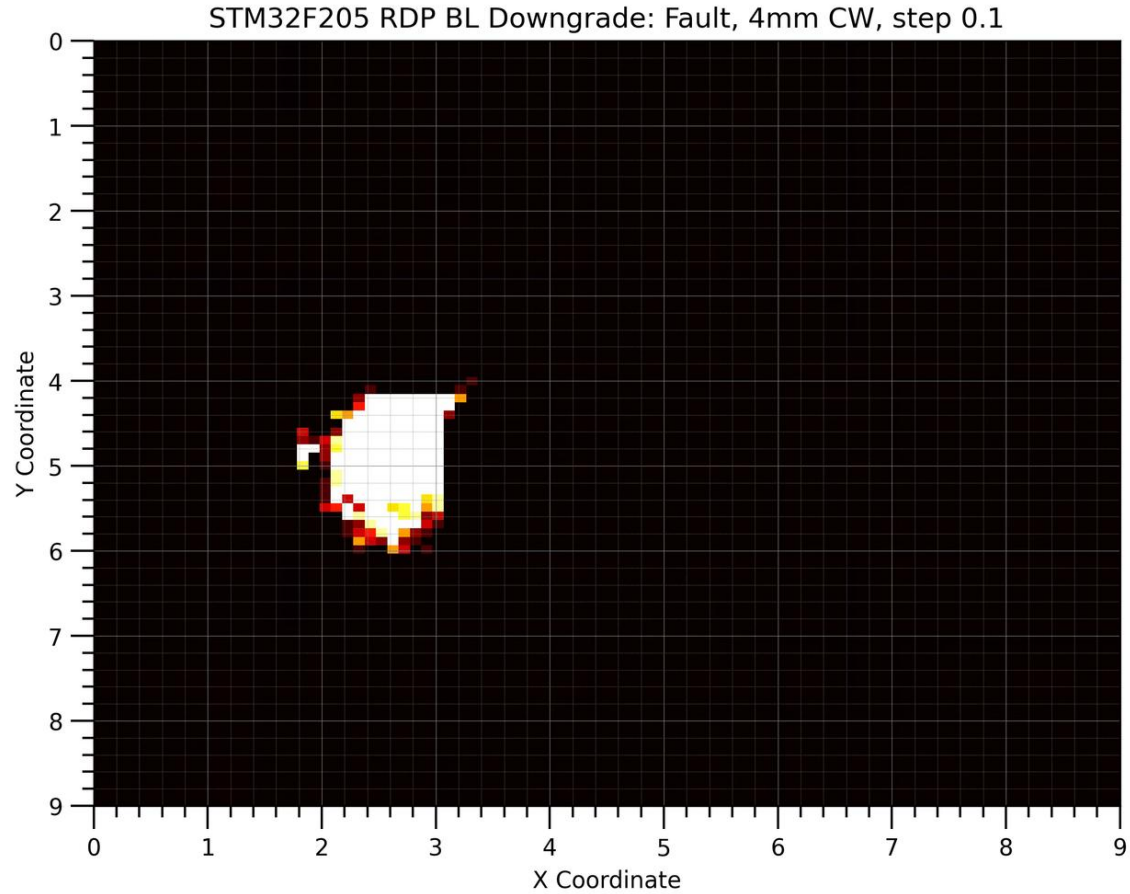


NRST TRIGGER

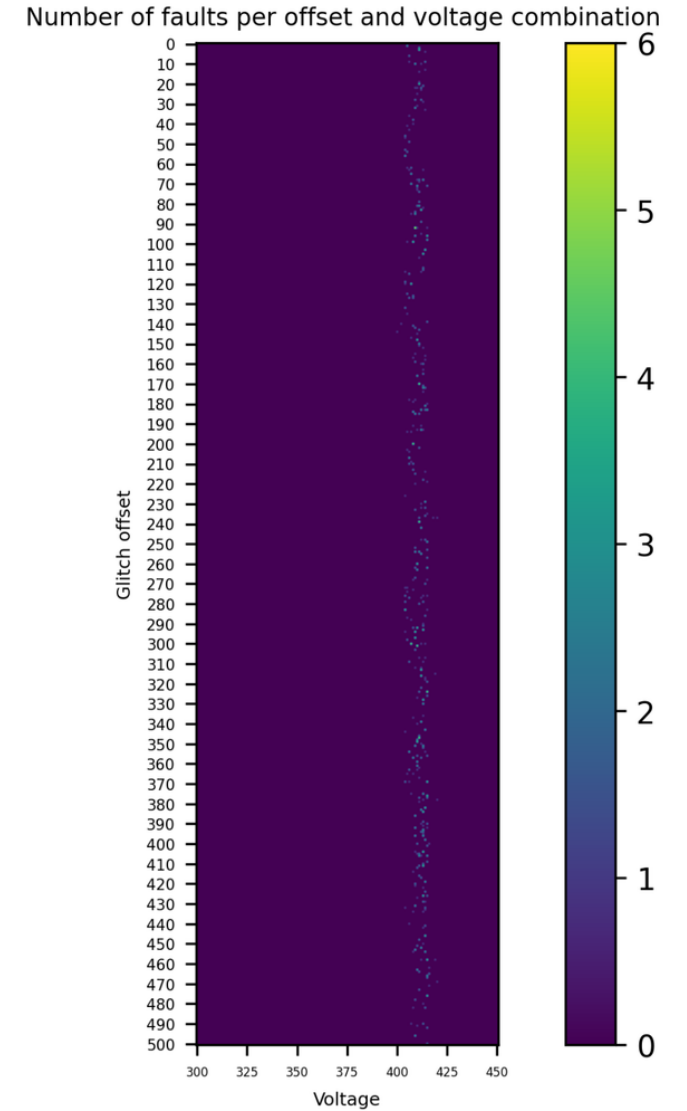
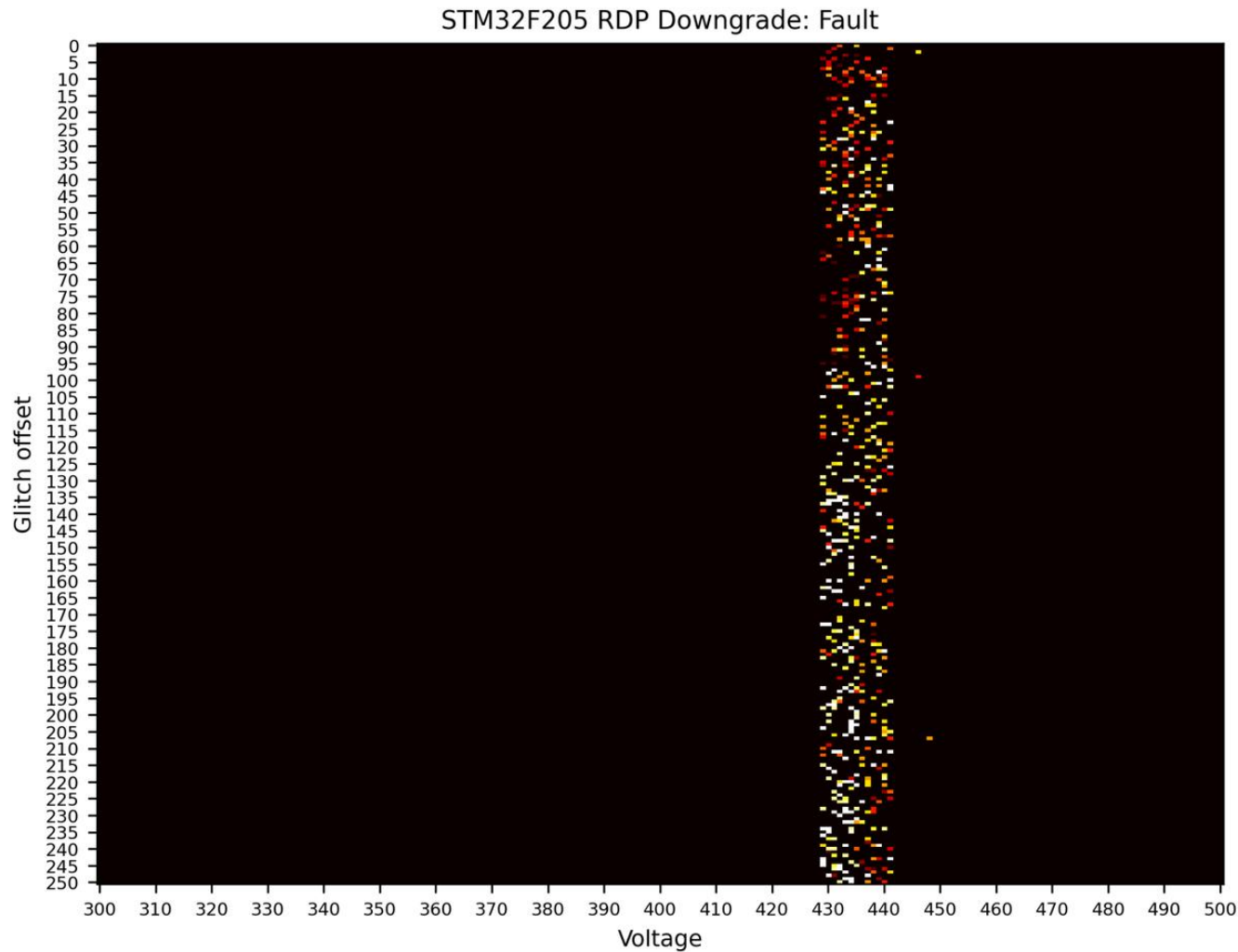
F2 RDP DOWNGRADE: POWER ANALYSIS



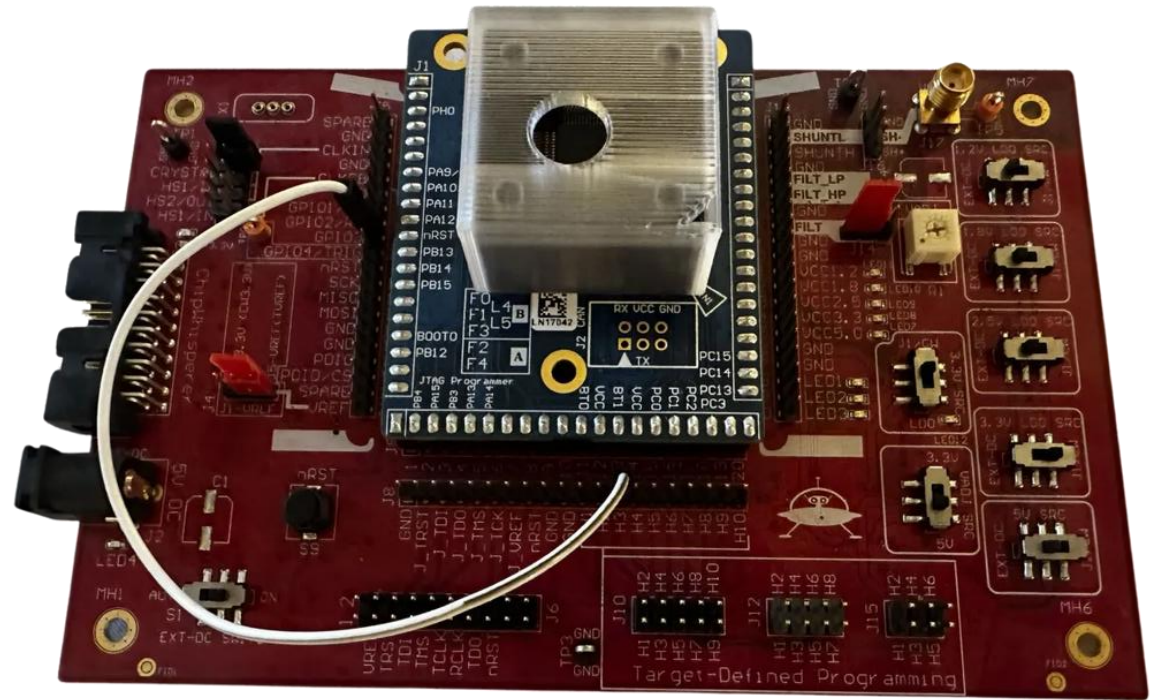
F2 RDP DOWNGRADE: POSITION (CW)



F2 RDP DOWNGRADE: OFFSET INDEPENDENT



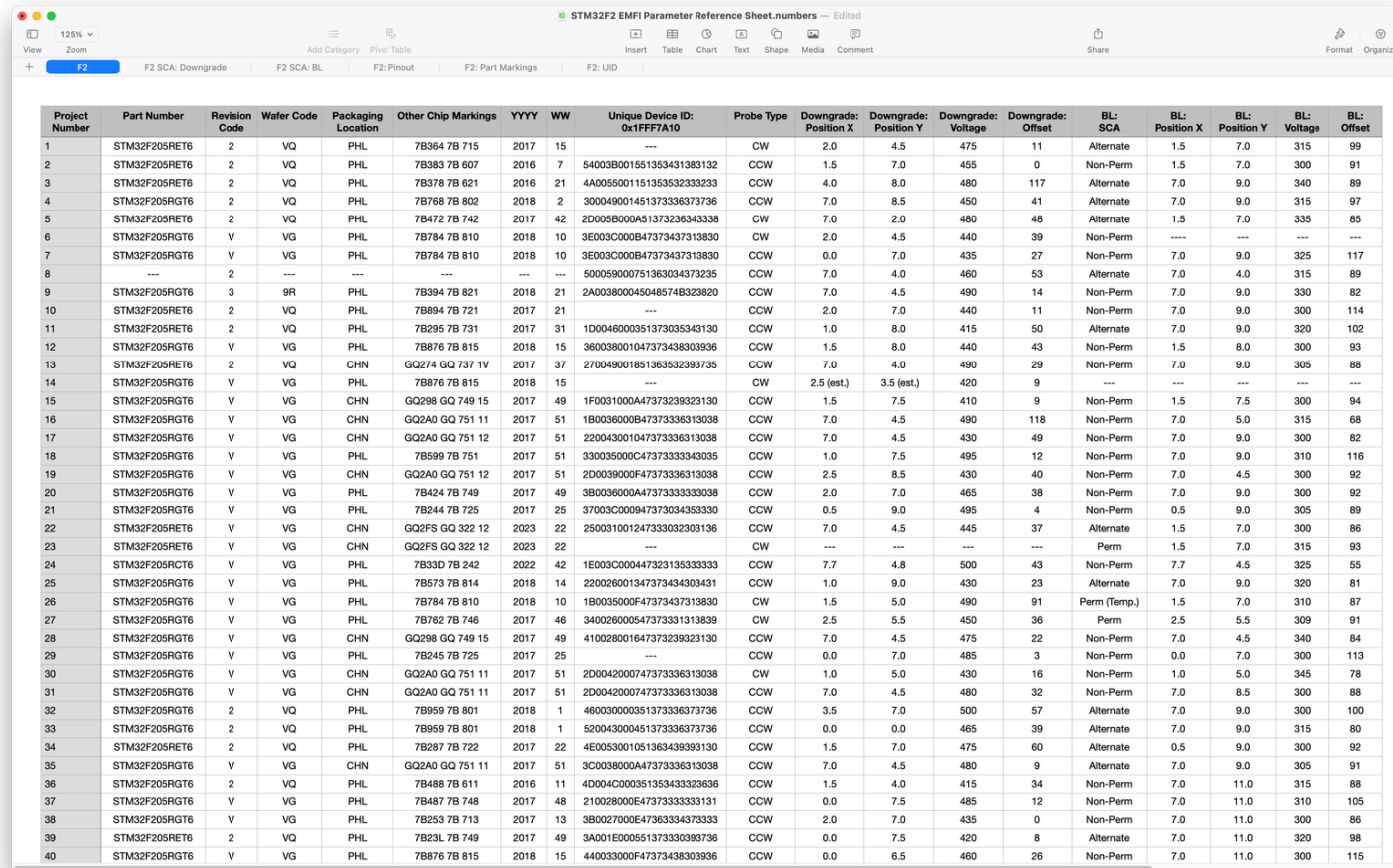
EMFI ALIGNMENT HATS



Worked for ~5 samples...
...then things got weird!

F2 RDP DOWNGRADE: PROBE POSITIONING

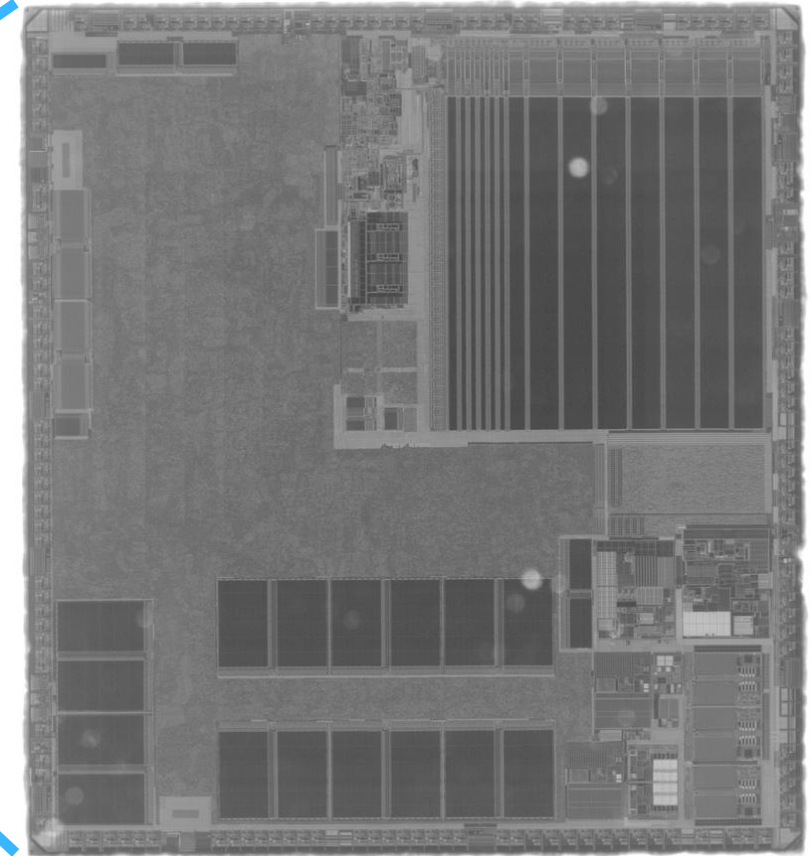
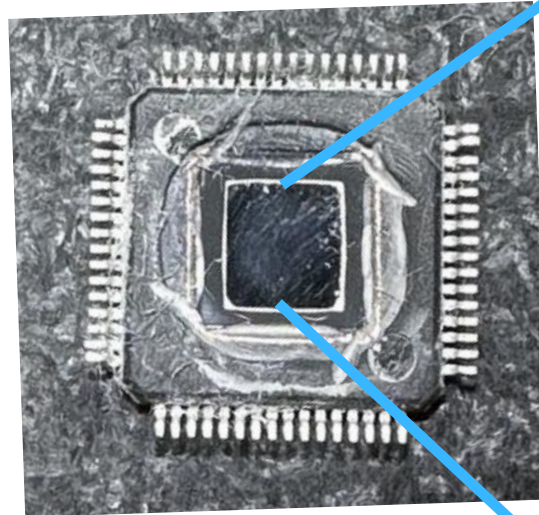
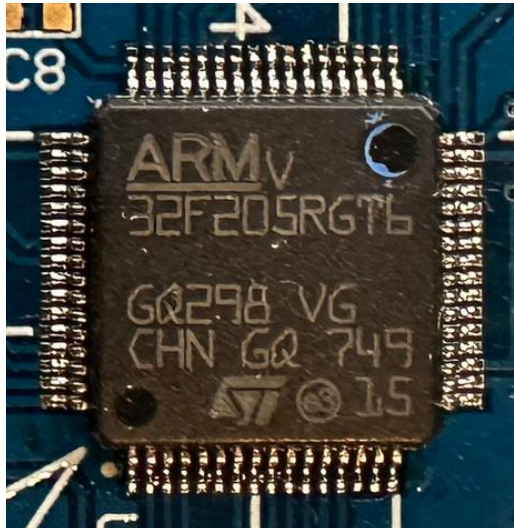
- Different devices resulted in different probe location for RDP downgrade
 - Based on fab location, packaging location, other manufacturing parameters?



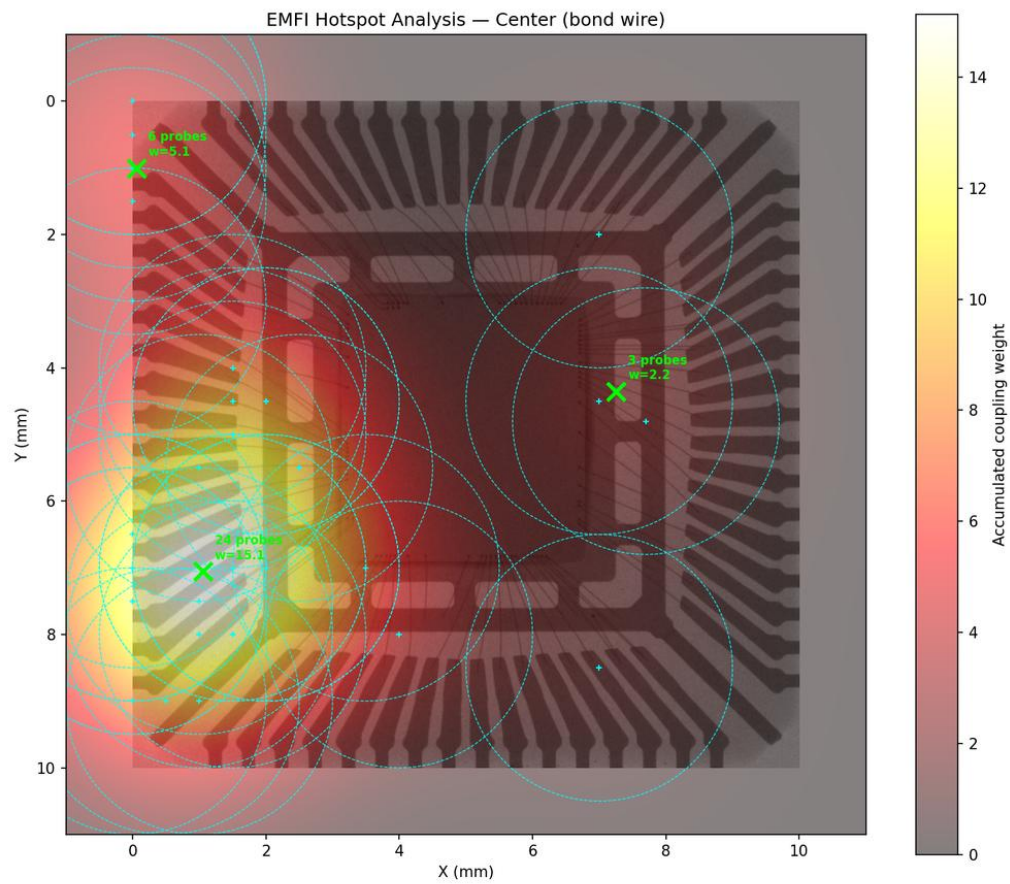
Project Number	Part Number	Revision Code	Wafer Code	Packaging Location	Other Chip Markings	YYYY	WW	Unique Device ID: 0x1FFF7A10	Probe Type	Downgrade: Position X	Downgrade: Position Y	Downgrade: Voltage	Downgrade: Offset	BL: SCA	BL: Position X	BL: Position Y	BL: Voltage	BL: Offset
1	STM32F205RET6	2	VQ	PHL	7B364 7B 715	2017	15	---	CW	2.0	4.5	475	11	Alternate	1.5	7.0	315	99
2	STM32F205RET6	2	VQ	PHL	7B383 7B 607	2016	7	540038001551353431383132	CCW	1.5	7.0	455	0	Non-Perm	1.5	7.0	300	91
3	STM32F205RET6	2	VQ	PHL	7B378 7B 621	2016	21	4A0055001151353532333233	CCW	4.0	8.0	480	117	Alternate	7.0	9.0	340	89
4	STM32F205RGTE	2	VQ	PHL	7B768 7B 802	2018	2	300049001451373336373736	CCW	7.0	8.5	450	41	Alternate	7.0	9.0	315	97
5	STM32F205RET6	2	VQ	PHL	7B472 7B 742	2017	42	2D005B000A51373236343338	CW	7.0	2.0	480	48	Alternate	1.5	7.0	335	85
6	STM32F205RGTE	V	VG	PHL	7B784 7B 810	2018	10	3E003C000B47373437313830	CW	2.0	4.5	440	39	Non-Perm	----	----	----	----
7	STM32F205RGTE	V	VG	PHL	7B784 7B 810	2018	10	3E003C000B47373437313830	CCW	0.0	7.0	435	27	Non-Perm	7.0	9.0	325	117
8	---	2	---	---	---	---	---	500059000751363034373235	CCW	7.0	4.0	460	53	Alternate	7.0	4.0	315	89
9	STM32F205RGTE	3	9R	PHL	7B394 7B 821	2018	21	2A003800045048574B323820	CCW	7.0	4.5	490	14	Non-Perm	7.0	9.0	330	82
10	STM32F205RET6	2	VQ	PHL	7B894 7B 721	2017	21	---	CCW	2.0	7.0	440	11	Non-Perm	7.0	9.0	300	114
11	STM32F205RET6	2	VQ	PHL	7B295 7B 731	2017	31	1D0046000351373035343130	CCW	1.0	8.0	415	50	Alternate	7.0	9.0	320	102
12	STM32F205RGTE	V	VG	PHL	7B876 7B 815	2018	15	360038001047373438303936	CCW	1.5	8.0	440	43	Non-Perm	1.5	8.0	300	93
13	STM32F205RET6	2	VQ	CHN	GQ274 GQ 737 1V	2017	37	270049001851363532393735	CCW	7.0	4.0	490	29	Non-Perm	7.0	9.0	305	88
14	STM32F205RGTE	V	VG	PHL	7B876 7B 815	2018	15	---	CW	2.5 (est.)	3.5 (est.)	420	9	---	---	---	---	---
15	STM32F205RGTE	V	VG	CHN	GQ298 GQ 749 15	2017	49	1F0031000A47373239323130	CCW	1.5	7.5	410	9	Non-Perm	1.5	7.5	300	94
16	STM32F205RGTE	V	VG	CHN	GQ2A0 GQ 751 11	2017	51	1B0036000B47373336313038	CCW	7.0	4.5	490	118	Non-Perm	7.0	5.0	315	68
17	STM32F205RGTE	V	VG	CHN	GQ2A0 GQ 751 12	2017	51	220043001047373336313038	CCW	7.0	4.5	430	49	Non-Perm	7.0	9.0	300	82
18	STM32F205RGTE	V	VG	PHL	7B599 7B 751	2017	51	330035000C47373333343035	CCW	1.0	7.5	495	12	Non-Perm	7.0	9.0	310	116
19	STM32F205RGTE	V	VG	CHN	GQ2A0 GQ 751 12	2017	51	2D0039000F47373336313038	CCW	2.5	8.5	430	40	Non-Perm	7.0	4.5	300	92
20	STM32F205RGTE	V	VG	PHL	7B424 7B 749	2017	49	3B0036000A47373333333038	CCW	2.0	7.0	465	38	Non-Perm	7.0	9.0	300	92
21	STM32F205RGTE	V	VG	PHL	7B244 7B 725	2017	25	37003C000947373034353330	CCW	0.5	9.0	495	4	Non-Perm	0.5	9.0	305	89
22	STM32F205RET6	V	VG	CHN	GQ2FS GQ 322 12	2023	22	250031001247333032303136	CCW	7.0	4.5	445	37	Alternate	1.5	7.0	300	86
23	STM32F205RET6	V	VG	CHN	GQ2FS GQ 322 12	2023	22	---	CW	---	---	---	---	Perm	1.5	7.0	315	93
24	STM32F205RCT6	V	VG	PHL	7B33D 7B 242	2022	42	1E003C000447323135333333	CCW	7.7	4.8	500	43	Non-Perm	7.7	4.5	325	55
25	STM32F205RGTE	V	VG	PHL	7B573 7B 814	2018	14	220026001347373434303431	CCW	1.0	9.0	430	23	Alternate	7.0	9.0	320	81
26	STM32F205RGTE	V	VG	PHL	7B784 7B 810	2018	10	1B0035000F47373437313830	CW	1.5	5.0	490	91	Perm (Temp.)	1.5	7.0	310	87
27	STM32F205RGTE	V	VG	PHL	7B762 7B 746	2017	46	340026000547373331313839	CW	2.5	5.5	450	36	Perm	2.5	5.5	309	91
28	STM32F205RGTE	V	VG	CHN	GQ298 GQ 749 15	2017	49	410028001647373239323130	CCW	7.0	4.5	475	22	Non-Perm	7.0	4.5	340	84
29	STM32F205RGTE	V	VG	PHL	7B245 7B 725	2017	25	---	CCW	0.0	7.0	485	3	Non-Perm	0.0	7.0	300	113
30	STM32F205RGTE	V	VG	CHN	GQ2A0 GQ 751 11	2017	51	2D0042000747373336313038	CW	1.0	5.0	430	16	Non-Perm	1.0	5.0	345	78
31	STM32F205RGTE	V	VG	CHN	GQ2A0 GQ 751 11	2017	51	2D0042000747373336313038	CCW	7.0	4.5	480	32	Non-Perm	7.0	8.5	300	88
32	STM32F205RGTE	2	VQ	PHL	7B959 7B 801	2018	1	460030000351373336373736	CCW	3.5	7.0	500	57	Alternate	7.0	9.0	300	100
33	STM32F205RGTE	2	VQ	PHL	7B959 7B 801	2018	1	520043000451373336373736	CCW	0.0	0.0	465	39	Alternate	7.0	9.0	315	80
34	STM32F205RET6	2	VQ	PHL	7B287 7B 722	2017	22	4E0053001051363439393130	CCW	1.5	7.0	475	60	Alternate	0.5	9.0	300	92
35	STM32F205RGTE	V	VG	CHN	GQ2A0 GQ 751 11	2017	51	3C0038000A47373336313038	CCW	7.0	4.5	480	9	Alternate	7.0	9.0	305	91
36	STM32F205RGTE	2	VQ	PHL	7B486 7B 611	2016	11	4D004C000351353433323636	CCW	1.5	4.0	415	34	Non-Perm	7.0	11.0	315	88
37	STM32F205RGTE	V	VG	PHL	7B487 7B 748	2017	48	210028000E47373333333131	CCW	0.0	7.5	485	12	Non-Perm	7.0	11.0	310	105
38	STM32F205RGTE	V	VG	PHL	7B253 7B 713	2017	13	3B0027000E47363334373333	CCW	2.0	7.0	435	0	Non-Perm	7.0	11.0	300	86
39	STM32F205RET6	2	VQ	PHL	7B23L 7B 749	2017	49	3A001E000551373330393736	CCW	0.0	7.5	420	8	Alternate	7.0	11.0	320	98
40	STM32F205RGTE	V	VG	PHL	7B876 7B 815	2018	15	440033000F47373438303936	CCW	0.0	6.5	460	26	Non-Perm	7.0	11.0	300	115

CHIP PACKAGING

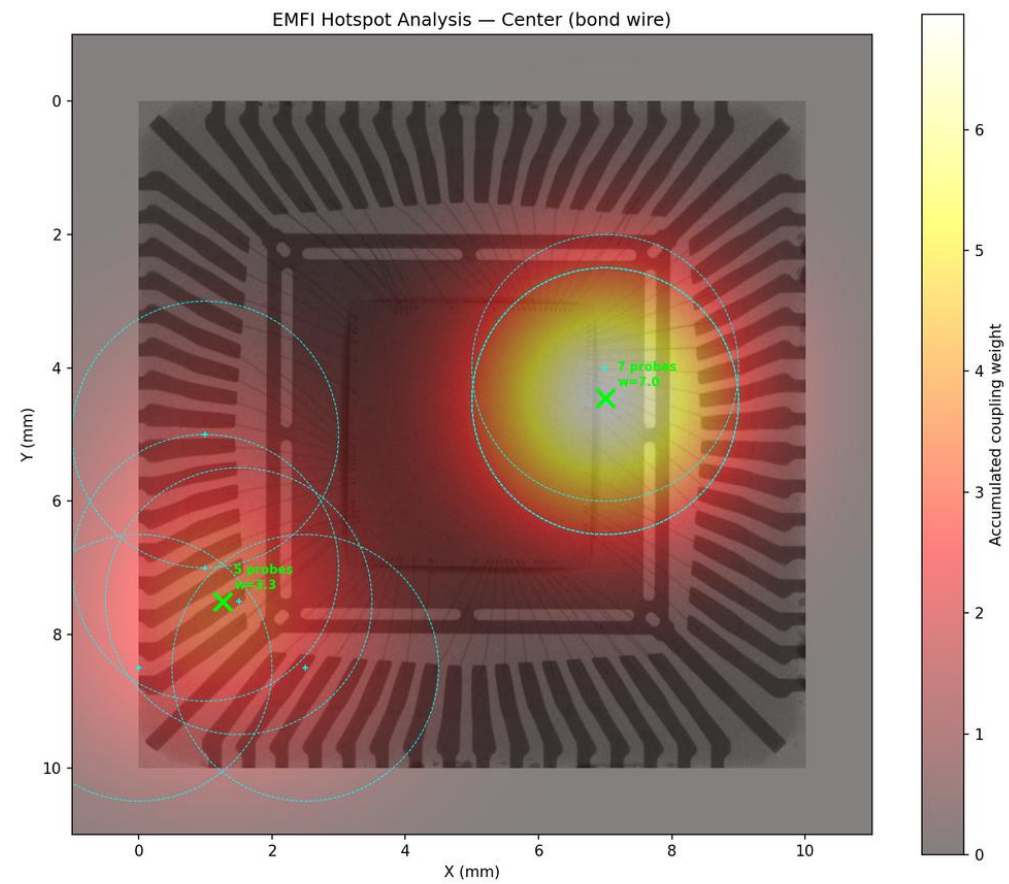
- EMFI can be more localized than VFI, but often couples to bond wires
- For many package types, the actual chip/die is much smaller than the package



X-RAY + EMFI LOCATION

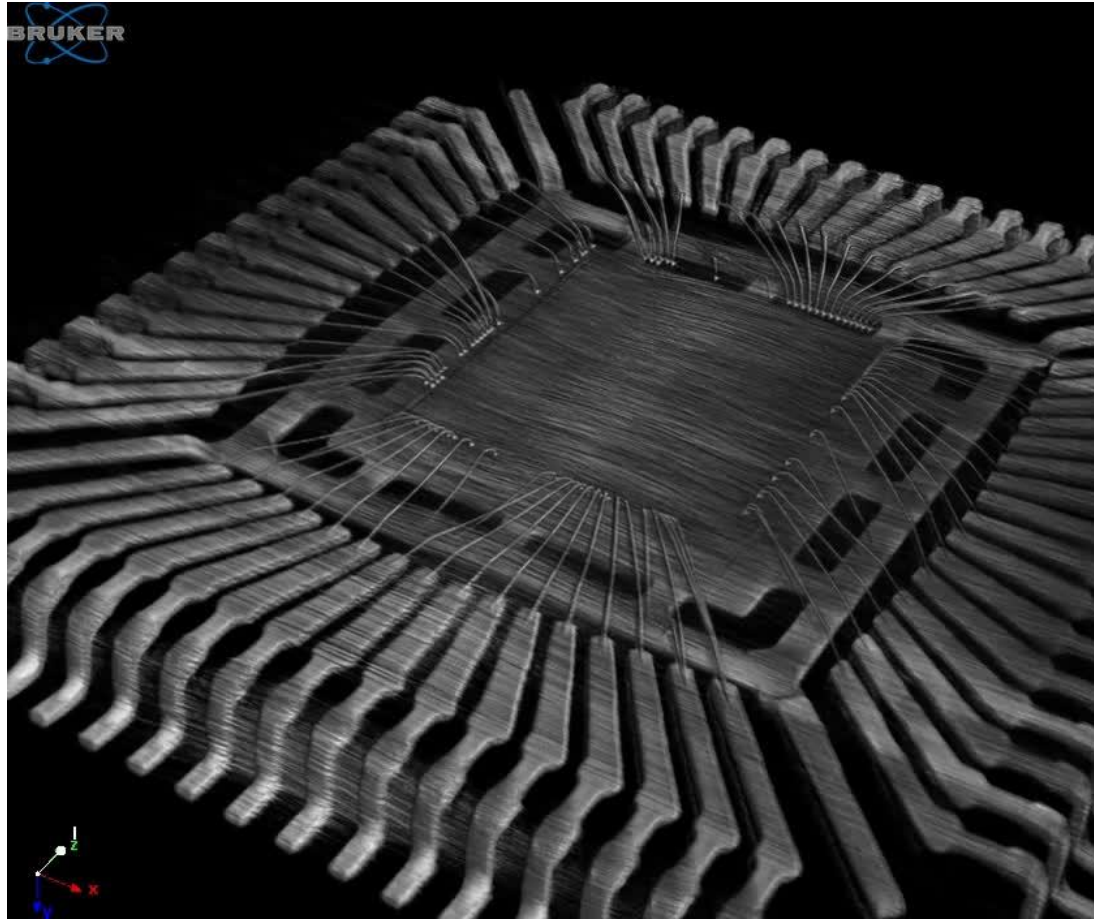


PHL (Phillipines)

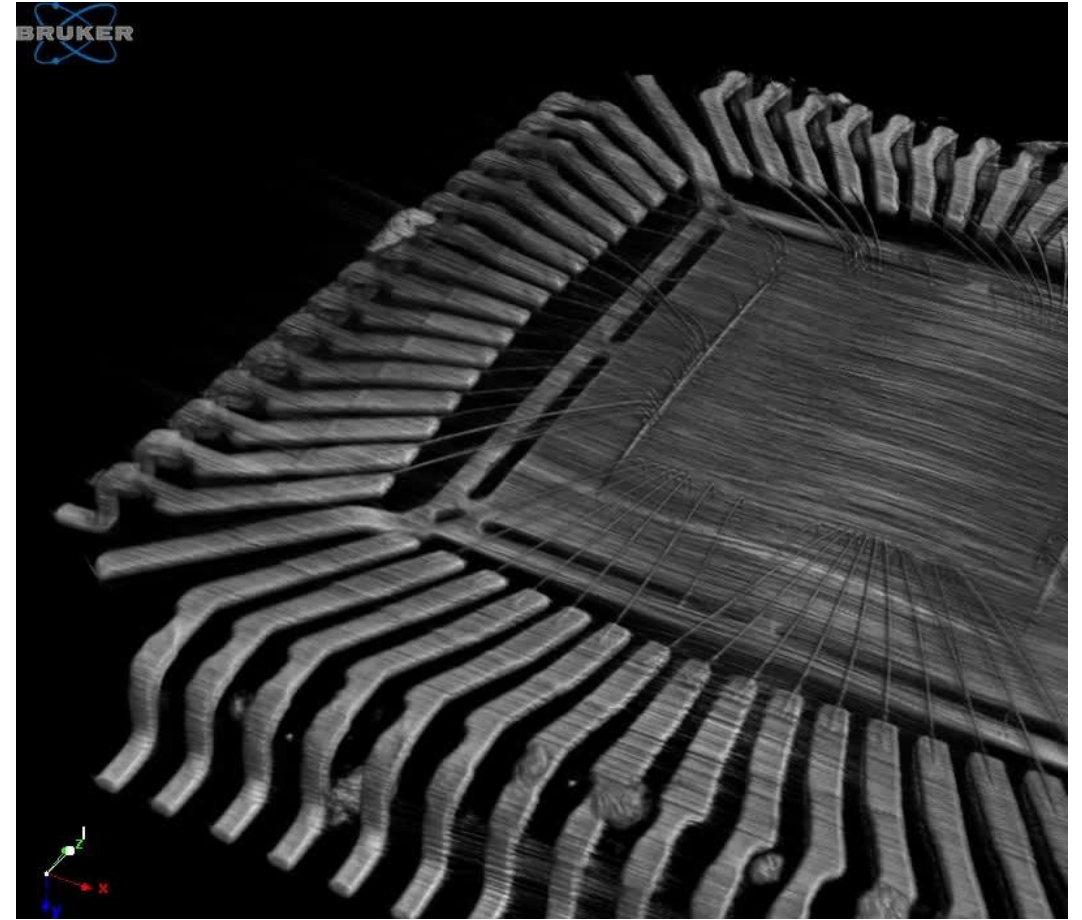


CHN (China)

WIREBOND ANALYSIS (CT)

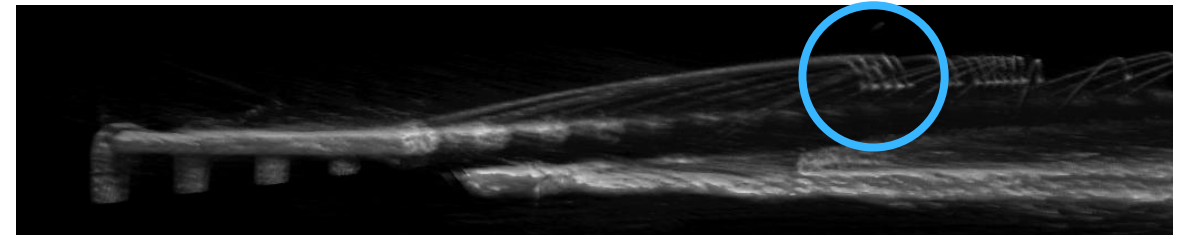
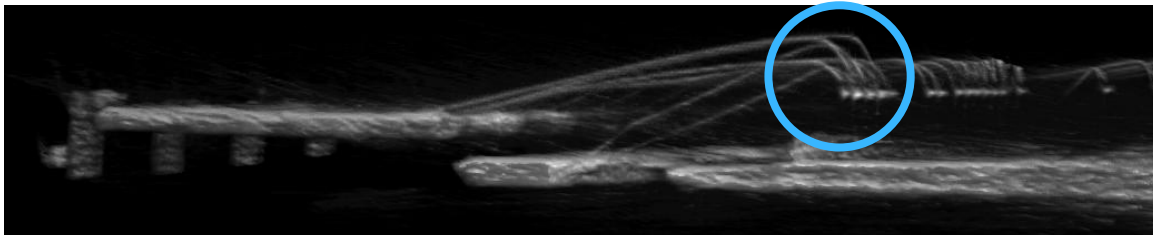
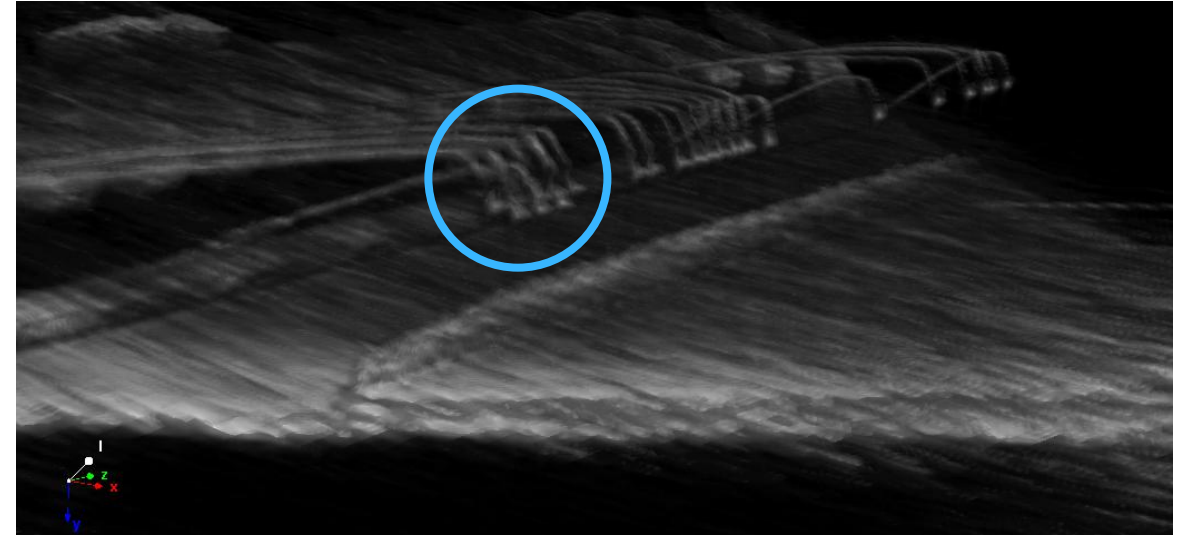
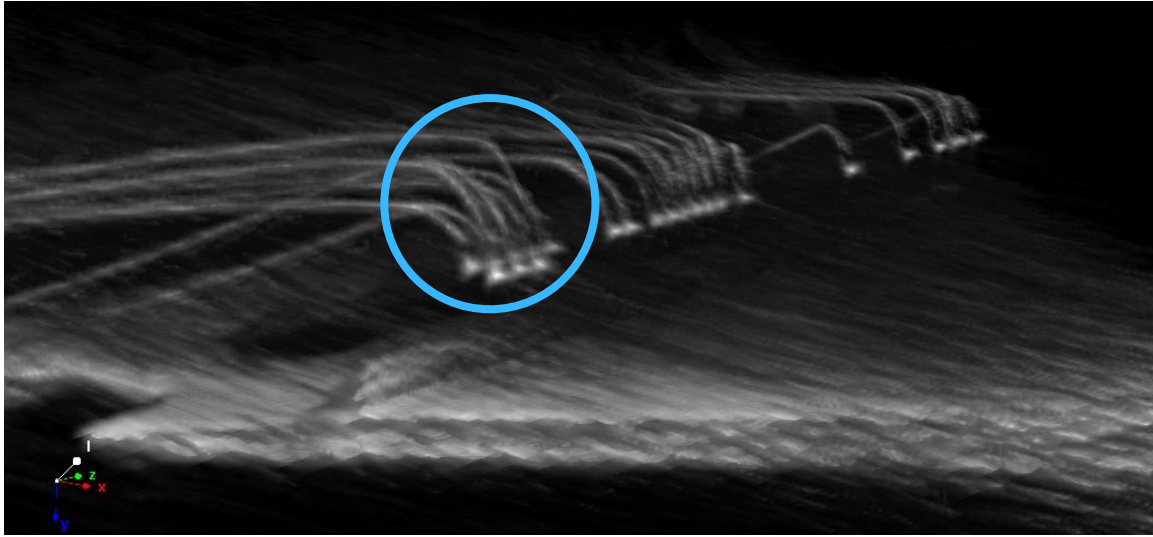


PHL (Phillipines)



CHN (China)

WIREBOND ANALYSIS (CT)

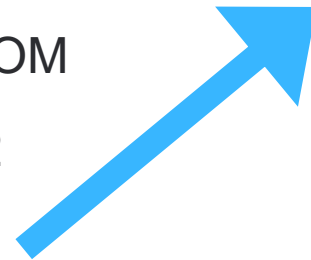


PHL (Phillipines)

CHN (China)

STM32 BOOTLOADER

- After downgrading to RDP1 we can access the bootloader serial interface
- Provides a low-level, command-based interface to perform programming-related operations on the STM32
- Supports multiple peripherals depending on version
 - UART, CAN, USB, I2C, SPI
- Stored in the internal (unchangeable) Boot ROM
- AN3155: USART protocol used in the STM32 bootloader

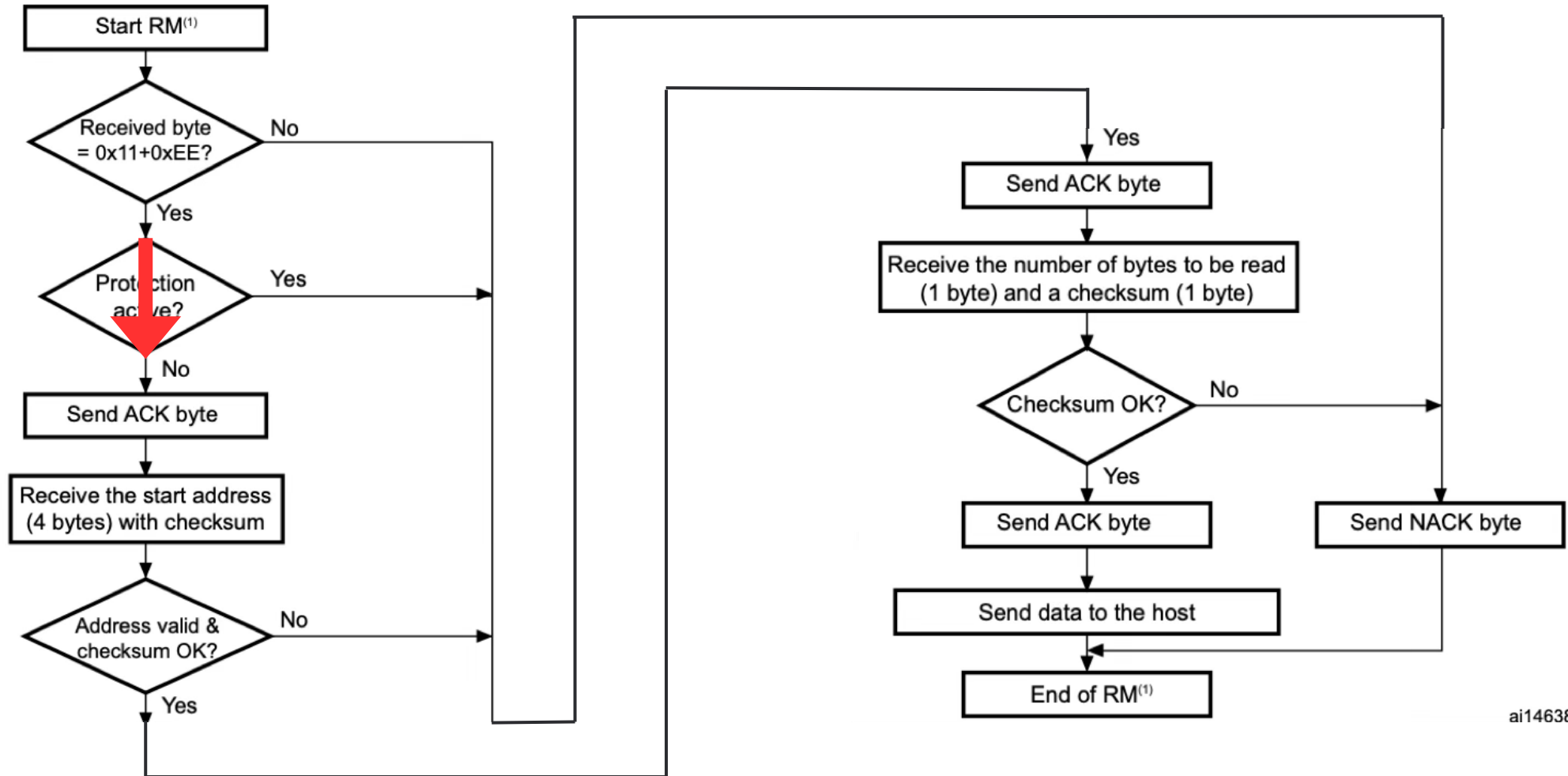


Command ⁽¹⁾	Code	Description
Get ⁽²⁾	0x00	Gets the version and the allowed commands supported by the current version of the protocol.
Get Version ⁽²⁾	0x01	Gets the protocol version.
Get ID ⁽²⁾	0x02	Gets the chip ID.
Read Memory ⁽³⁾	0x11	Reads up to 256 bytes of memory starting from an address specified by the application.
Go ⁽³⁾	0x21	Jumps to user application code located in the internal flash memory or in the SRAM.
Write Memory ⁽³⁾	0x31	Writes up to 256 bytes to the RAM or flash memory starting from an address specified by the application.
Erase ⁽³⁾⁽⁴⁾	0x43	Erases from one to all the flash memory pages.
Extended Erase ⁽³⁾⁽⁴⁾⁽⁵⁾	0x44	Erases from one to all the flash memory pages using two-byte addressing mode (available only for USART bootloader v3.0 and higher).
Special ⁽⁵⁾	0x50	Generic command that allows to add new features depending on the product constraints, without adding a new command for every feature.
Extended Special ⁽⁵⁾	0x51	Generic command that allows the user to send more data compared to the Special command.
Write Protect ⁽⁵⁾	0x63	Enables the write protection for some sectors.
Write Unprotect ⁽⁵⁾	0x73	Disables the write protection for all flash memory sectors.
Readout Protect	0x82	Enables the read protection.
Readout Unprotect ⁽²⁾	0x92	Disables the read protection.
Get Checksum ⁽⁵⁾	0xA1	Computes a CRC value on a given memory area with a size multiple of 4 bytes.

1. If a denied command is received or an error occurs during the command execution, the bootloader sends an NACK byte, and goes back to command checking.

2. Protection. When active, only this limited subset of commands is available, all others are NACK-ed, and have no effect on the device. The protection depends upon the product.

STM32 BOOTLOADER: READ MEMORY



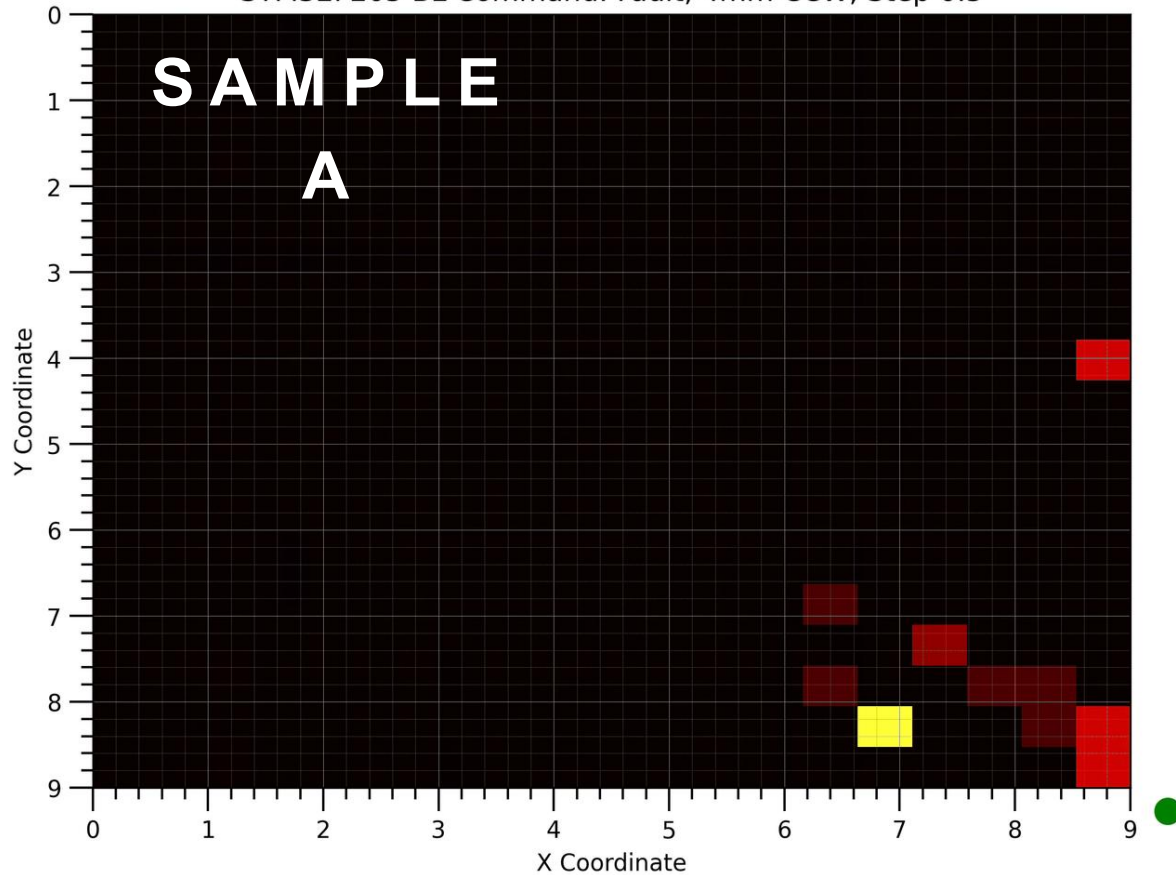
STM32 BOOTLOADER: TRIGGERING

- Jitter due to UART, clocks not synchronized
- Risk of option byte corruption and/or wipe!

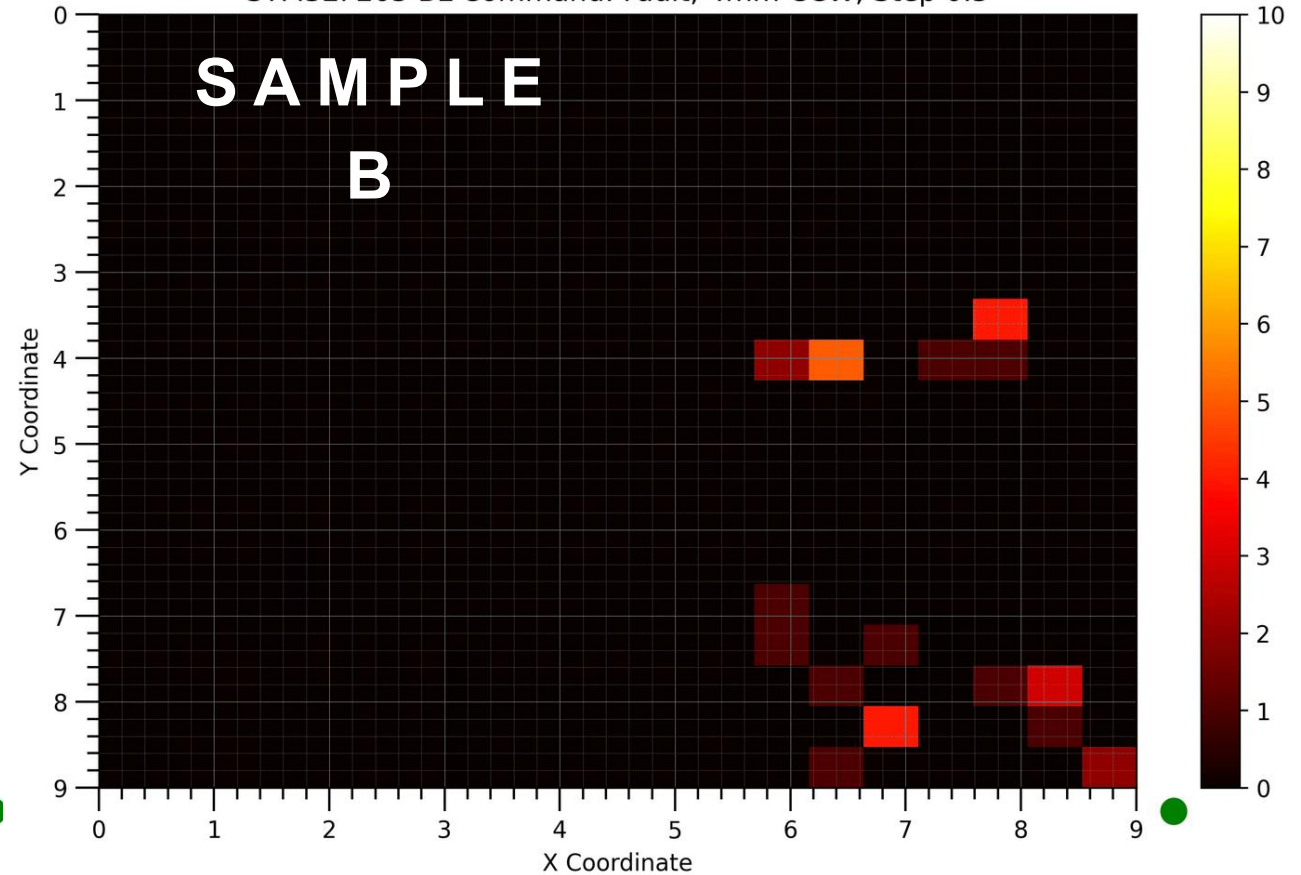


READ MEMORY GLITCH: POSITION (CCW)

STM32F205 BL Command: Fault, 4mm CCW, Step 0.5



STM32F205 BL Command: Fault, 4mm CCW, Step 0.5



STM32F2 TAKEAWAYS

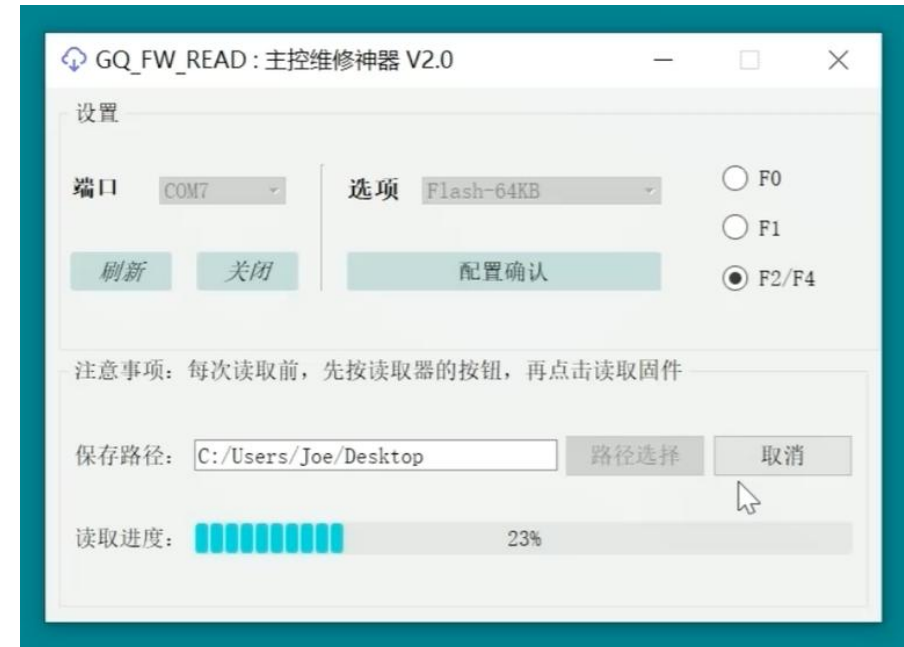
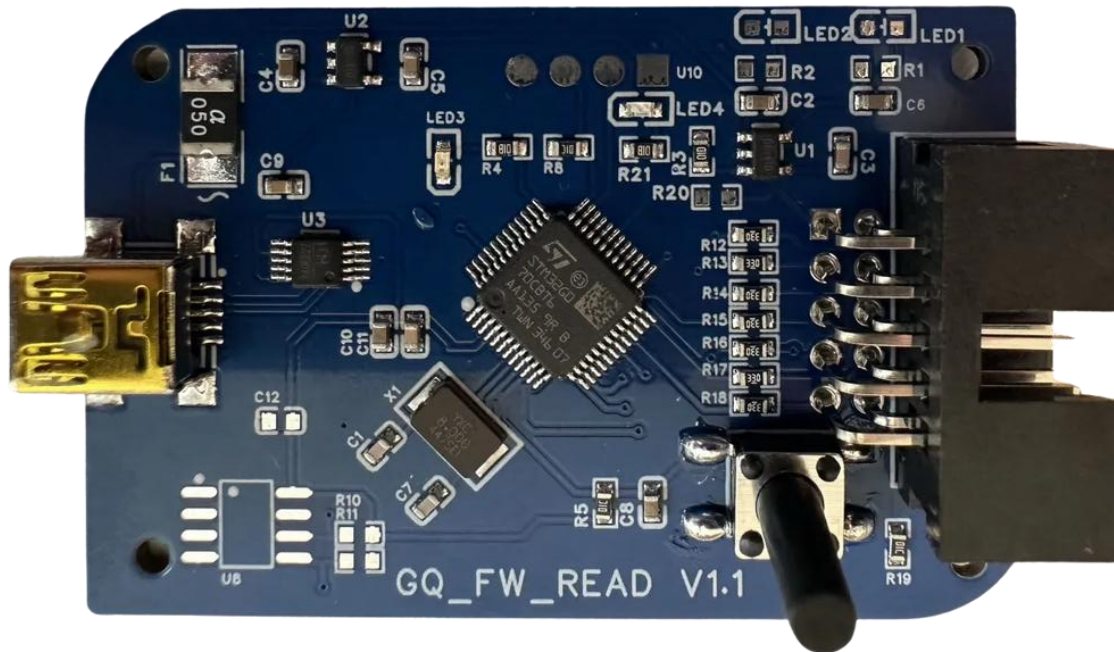
- Downgrading the F2 with EMFI
 - Probe location and voltage are important, offset from trigger is not
 - Downgrade offset range is very early, likely before the ROM bootloader even starts
 - We never observed any corruption or wipe while using this technique
- Glitching the read memory command can cause data loss
 - ADC level or SAD triggering is a requirement for consistent timing
 - Careful characterization over multiple target samples
 - For a hardware wallet, we only have to glitch the read memory function a few times to extract the needed data

STM32F205 → F427

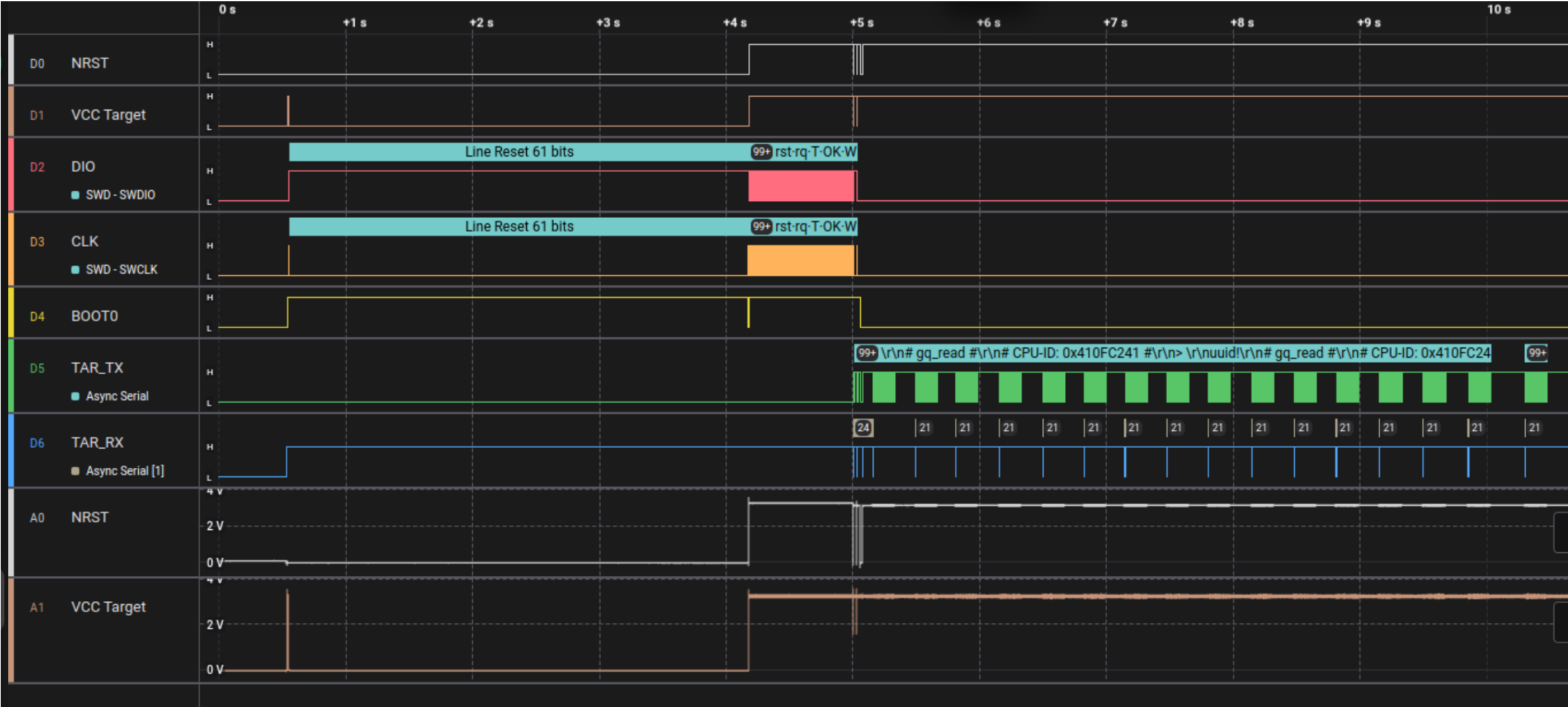
- Difficulties with downgrading the STM32F427
 - Early downgrade didn't work
 - Permanent Option Byte corruption
 - Wipes during Read Memory
 - Additional security features: Proprietary Code Read Out Protection (PCROP)
- Developing a reliable downgrade strategy and the move back to VFI
- Creating our previously unpublished attack that allows to extract the full Flash memory using only two faults

SIDE QUEST: AN STM32 “DECRYPTOR”

- We purchased a commercial STM32 "decryptor" from China (Xianyu / Goofish) advertised to extract flash from RDP1-protected devices
 - We learned about these devices through a talk from Chen Guannan (OSR Security)
 - Reverse engineering this tool gave us an idea for a new approach!



SIDE QUEST: AN STM32 “DECRYPTOR”



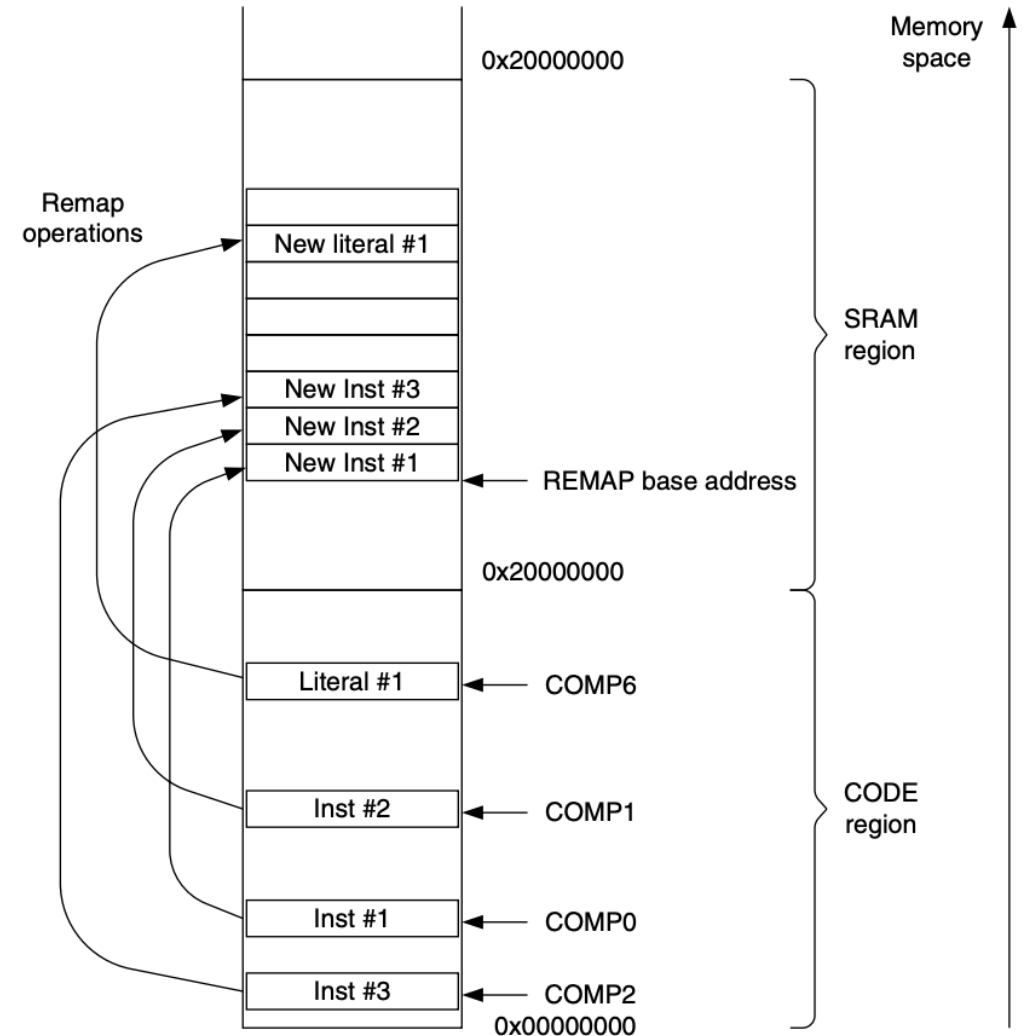
SIDE QUEST: AN STM32 “DECRYPTOR”

The screenshot displays a debugger interface with the following components:

- Program Trees:** Shows the file structure of `payload.bin`, including `sram`, `TIM2_TIM3_TIM4_TIM5_TIM6_TIM7_TIM12_TIM13`, `RTC_WWDG_IWDG_I2S2ext_SPI2_SPI3_I2S4`, `CAN1_CAN2`, `PWR_DAC_UART7_UART8`, `TIM1_TIM8`, `USART1_USART6`, `ADC1`, and `ADC2`.
- Symbol Tree:** Lists imports, exports, and functions. Functions include `command_parser`, `exception_handler`, `FUN_20000184`, `FUN_2000019c`, `FUN_200001dc`, `FUN_20000434`, and `infinite_loop`.
- Data Type M...:** Shows data types, including `BuiltInTypes`, `payload.bin`, and `generic_clib`.
- Listing: payload.bin:** Displays assembly code with addresses, hex values, and mnemonics. Key instructions include `ldr r2, [r3, #0x0] => Peripherals::RCC`, `ldr r2, [r3, #0x19]`, `beq LAB_200004e2`, `ldr r3, [r1, #0x0] => RCC.CFGR`, `orr r3, r3, #0x2`, `str r3, [r1, #0x0] => RCC.CFGR`, `movs r4, #0x0`, `ldr r3, [sp, #0x0] => local_18`, `cmp r3, #0x2`, `bne LAB_200005c4`, `ldr r2, [DAT_20000614]`, `ldr r4, [DAT_20000618]`, `ldr r3, [r2, #0x0] => RCC.AHB1ENR`, `ldr r5, [DAT_2000061c]`, `orr r3, r3, #0x1`, `str r3, [r2, #0x0] => RCC.AHB1ENR`, `ldr r3, [r2, #0x14] => RCC.APB2ENR`, `orr r3, r3, #0x10`, `str r3, [r2, #0x14] => RCC.APB2ENR`, `ldr r3, [DAT_20000620]`, `str r3, [r0, #0x0] => DAT_20003004`, `ldr r3, [DAT_20000624]`, `ldr r0 = s_gg_read_#_CPU-ID: 0x_2000080e, [D...`, `ldr r2, [r3, #0x0] => Peripherals::GPIOA`, `bic r2, r2, #0xc0000`, `str r2, [r3, #0x0] => Peripherals::GPIOA`, `ldr r2, [r3, #0x0] => Peripherals::GPIOA`, `orr r2, r2, #0x80000`, `str r2, [r3, #0x0] => Peripherals::GPIOA`, `bic r2, r2, #0x300000`, `str r2, [r3, #0x0] => Peripherals::GPIOA`, `ldr r2, [r3, #0x0] => Peripherals::GPIOA`, `orr r2, r2, #0x200000`, `str r2, [r3, #0x0] => Peripherals::GPIOA`, `ldr r2, [DAT_2000062c]`, `ldr r3, [r2, #0x0] => GPIOA.OTYPER`, `bic r3, r3, #0x200`, `str r3, [r2, #0x0] => GPIOA.OTYPER`, `ldr r3, [r2, #0x4] => GPIOA.OSPEEDR`, `orr r3, r3, #0xc0000`, `str r3, [r2, #0x4] => GPIOA.OSPEEDR`, `ldr r3, [r2, #0x8] => GPIOA.PUPDR`, `bic r3, r3, #0x300000`, `str r3, [r2, #0x8] => GPIOA.PUPDR`.
- Decompile: main - (payload.bin):** Shows the decompiled C code, including the initialization of `Peripherals::RCC.AHB1ENR`, `Peripherals::RCC.APB2ENR`, `Peripherals::RCC.APB2ENR = uVar1 | 0x10`, `*(&undefined4 *)uVar2 = 0x5555aab6`, `uVar1 = Peripherals::GPIOA.MODER`, `Peripherals::GPIOA.MODER = uVar1 & 0xffff3fff`, `uVar1 = Peripherals::GPIOA.MODER`, `Peripherals::GPIOA.MODER = uVar1 | 0x80000`, `uVar1 = Peripherals::GPIOA.MODER`, `Peripherals::GPIOA.MODER = uVar1 & 0xffcfffff`, `uVar1 = Peripherals::GPIOA.MODER`, `Peripherals::GPIOA.MODER = uVar1 | 0x2000000`, `uVar1 = Peripherals::GPIOA.OTYPER`, `Peripherals::GPIOA.OTYPER = uVar1 & 0xffffdfff`, `uVar1 = Peripherals::GPIOA.OSPEEDR`, `Peripherals::GPIOA.OSPEEDR = uVar1 | 0xc0000`, `uVar1 = Peripherals::GPIOA.PUPDR`, `Peripherals::GPIOA.PUPDR = uVar1 & 0xffcfffff`, `uVar1 = Peripherals::GPIOA.AFRH`, `Peripherals::GPIOA.AFRH = uVar1 & 0xfffffff0f`, `uVar1 = Peripherals::GPIOA.AFRH`, `Peripherals::GPIOA.AFRH = uVar1 | 0x70`, `uVar1 = Peripherals::GPIOA.AFRH`, `Peripherals::GPIOA.AFRH = uVar1 & 0xfffff0ff`, `uVar1 = Peripherals::GPIOA.AFRH`, `Peripherals::GPIOA.AFRH = uVar1 | 0x700`, `Peripherals::USART1.BRR = 0xd0`, `Peripherals::USART1.CRL = 0x200c`, `printf("\r\n# gg_read #\r\n# CPU-ID: 0x");`, `uVar1 = Peripherals::SCB.CPUID`, `printWordHexBigEndian(uVar1)`, `printf("#\r\n");`, `printf("> ");`, `printf("\r\n");`, `printf("uuid");`, `printWordHexBigEndian(_DAT_1fff7a10)`, `do {`, `do {`, `FUN_20000184();`, `uVar1 = Peripherals::USART1.SR;`, `uVar1 = Peripherals::USART1.SR;`, `} while (-1 < (int)(uVar1 << 0x1a));`, `uVar1 = Peripherals::USART1.DR;`, `FUN_20000434(uVar1 & 0xff);`, `} while( true );`, `}`.

FPB

- Flash Patch and Breakpoint
- ARM Cortex-M3 Technical Reference Manual
- Implements hardware breakpoints, patches instruction or data from code space to SRAM
- Allows for in-the-field, live code patches



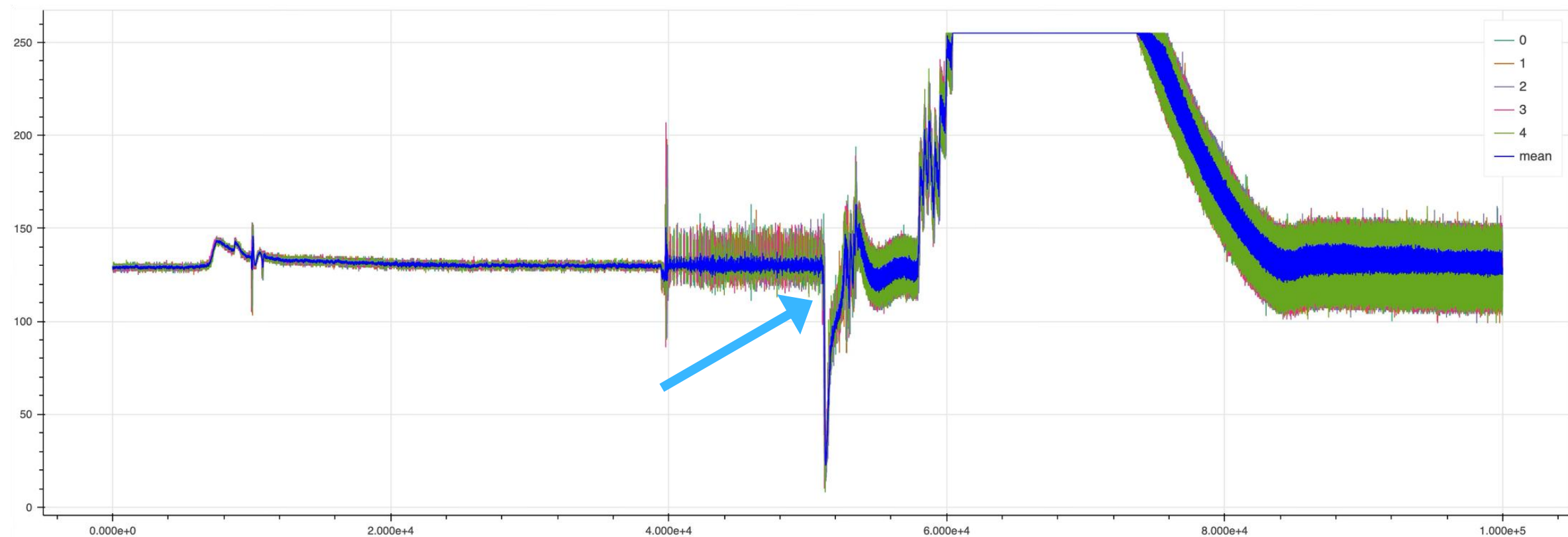
NEW RDP2 ATTACK METHODOLOGY

1. Downgrade RDP2-secured MCU into RDP1
2. Load payload into the now-accessible SRAM
3. Power cycle the chip with BOOT0 = 1 and BOOT1 = 1 to execute code from SRAM
 - Downgrade RDP2-secured MCU into RDP1 (again)
 - SRAM will need to persist!
 - Note: If MCU uses internal oscillator, user code may execute after an unsuccessful downgrade attempt, which could affect the payload in RAM
4. First stage of payload that was previously loaded into SRAM will execute
 - Payload will setup the FPB peripheral to redirect the reset vector to the second stage of the payload in SRAM

NEW RDP2 ATTACK METHODOLOGY

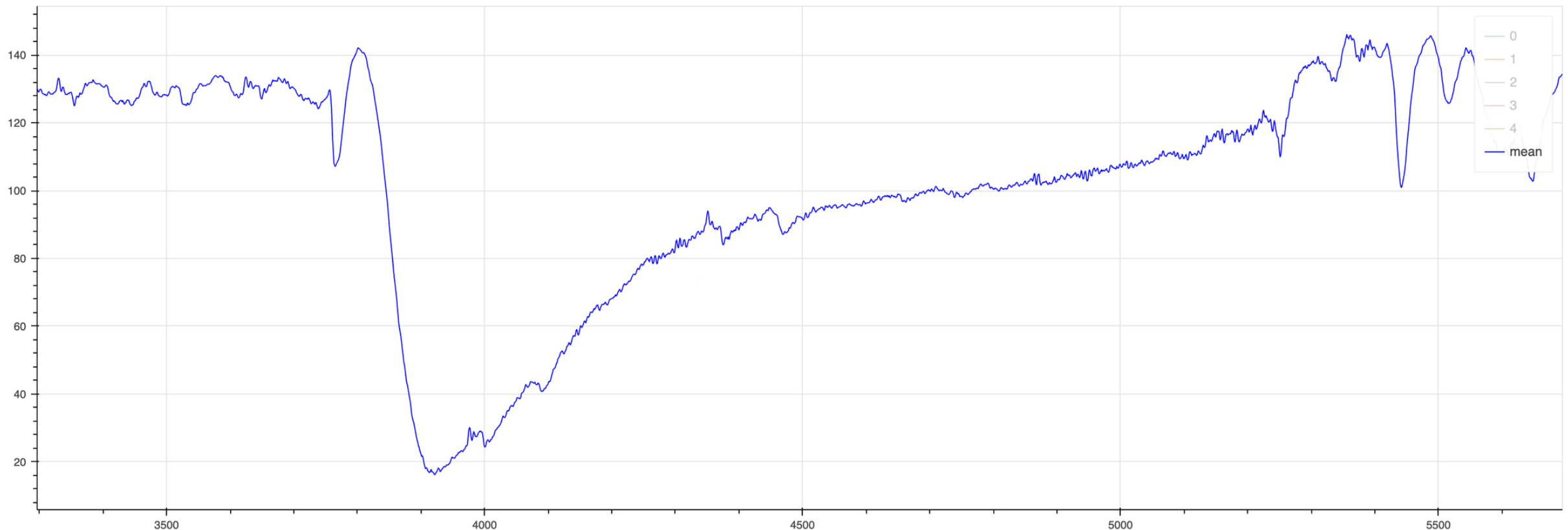
5. Perform a soft reset with $BOOT0 = 0$ and $BOOT1 = X$ to execute code from Flash
6. MCU fetches the reset vector and is redirected to the second stage of the payload in SRAM
 - MCU still thinks it's executing from Flash!
 - It's now in an RDP1 state with Flash memory accessible and the payload executing from SRAM
7. The second payload stage can read all Flash memory

F4 RDP DOWNGRADE: POWER ANALYSIS



F4 RDP DOWNGRADE: POWER ANALYSIS

- After NRST trigger, need to sequence with SAD for most reliable triggering
- Finding the best offset requires characterization, staring at side-channel traces and likely a few corrupted STM32F4s



VFI (F4)

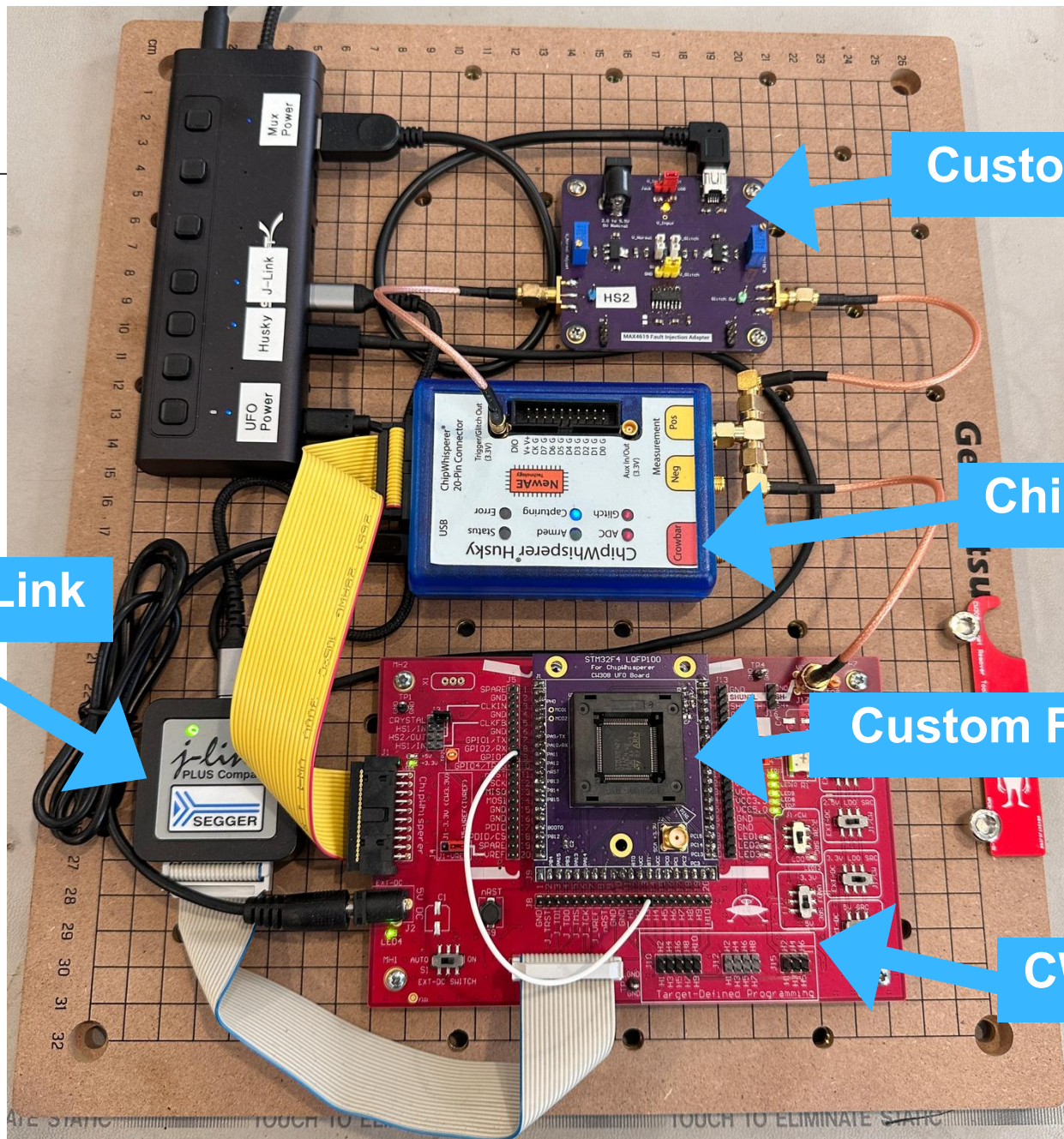
SEGGER J-Link

Custom MAX4619 PCB

ChipWhisperer Husky

Custom F4 LQFP100 Adapter

CW308 UFO Board



```

from bokeh.layouts import column
from bokeh.palettes import Dark2_5 as palette
import numpy as np
import pylink
import struct
import params
import re

```

Connect to the ChipWhisperer Husky

`ScopeWrapper` is convenience wrapper over the `Scope` class that adds some convenience functions.

```
[*]: scope = ScopeWrapper()
```

Connect to J-Link

```

[ ]: # https://pylink.readthedocs.io/en/latest/
jlink = pylink.JLink()

if jlink.opened() == False:
    jlink.open()

jlink.set_tif(pylink.enums.JLinkInterfaces.JTAG) # Set Target Interface as JTAG

print(jlink.product_name)

[ ]: # Retrieve IDCODE via low-level JTAG SHIFTDR, does not connect to target
def pylink_idcode():
    jlink.open()
    jlink.set_tif(pylink.enums.JLinkInterfaces.JTAG) # Set Target Interface as JTAG
    jlink.jtag_raw_rw([0], [0x1F], 5) # JTAG Reset (TMS = 1 for 5 bits)
    jlink.jtag_raw_rw([0], [0x02], 4) # JTAG ID read (IDCODE should be in DR upon TAP reset)
    result = jlink.jtag_raw_rw([0]*8, [0]*8) # Shift to see what's on scan chain
    hex_id = hex(struct.unpack("<I", bytes(result[0:4]))[0])
    return(hex_id)

```

IDCODE == 0xFFFFFFFF --> Target chip is in RDP2

IDCODE == 0x4ba00477 --> Target chip is in RDP1 or RDP0

```
[ ]: pylink_idcode()
```

3. Setup SAD trigger for downgrade glitch

1. Take a few side-channel measurements during a power cycle, we use the nRST signal as a trigger
2. Select a pattern for SAD triggering

STM32F4 TAKEAWAYS

- Downgrading the F4 (and F2) with VFI and FPB
 - Requires stable triggering and a precise offset
 - Requires identifying “safer” glitching settings
 - We used our custom MAX4619 board
 - Offset is very consistent between different F4 devices
 - We visually determine the offset for each F4 based on side-channel measurements

WISH LIST COMPLETE!

- Complete Flash extraction from an RDP2-secured MCU 
- Near-zero risk of unintentional corruption or data loss 
- Reliable and repeatable 
- Limit the number and type of faults 
- Use accessible/available hardware 
- No semi-invasive or invasive attacks 

EVERYTHING WAS DISCLOSED MARCH 2026

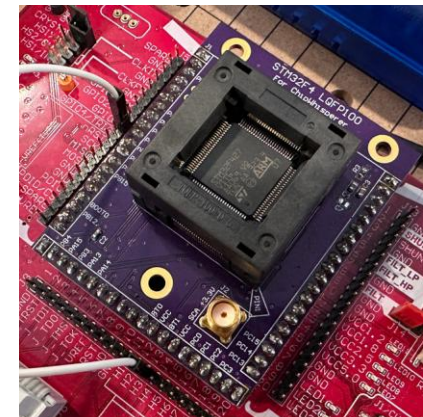
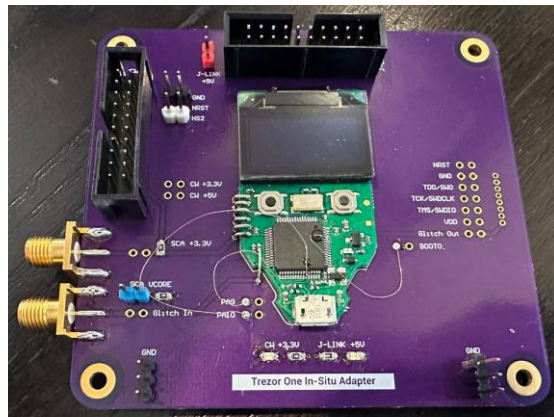
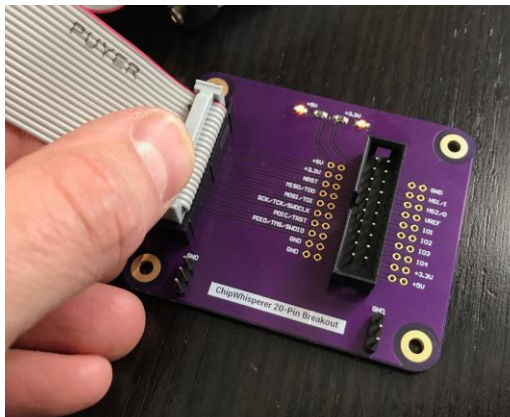
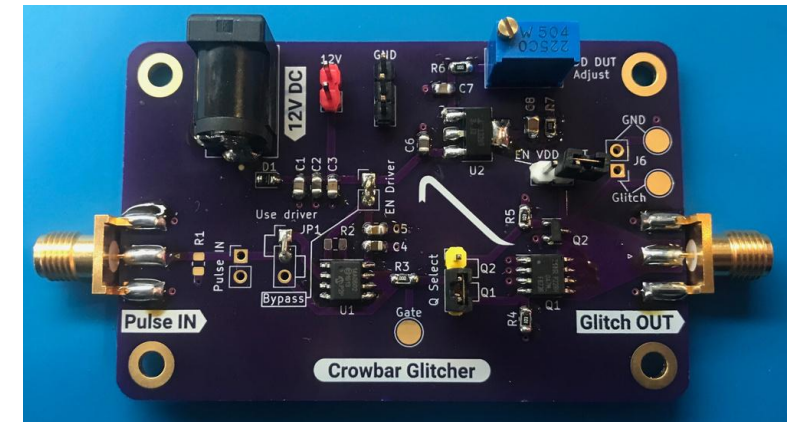
- ST MICROELECTRONICS (NO RESPONSE)
- TREZOR/SATOSHILABS (RESPONSE)

HARDWARE DESIGNS

- MAX4619 Analog Switch/Mux
- Crowbar Glitcher
- ChipWhisperer Breakout
- Trezor One In-Situ
- Trezor Model T In-Situ
- STM32F4 LQFP100 UFO Socket
- STM32F4 LQFP100 UFO Non-Socket

<https://github.com/joegrand/stm32-fault-injection>

REcon 2026: <https://youtu.be/4LahQG09NzI>



Follow us

- ❖ Website: <https://www.esat.kuleuven.be/cosic/>
- ❖ X @CosicBe
- ❖ Bluesky @cosic.bsky.social
- ❖ Mastodon @CosicBe
- ❖ LinkedIn: <https://www.linkedin.com/school/cosic-ku-leuven-/>

