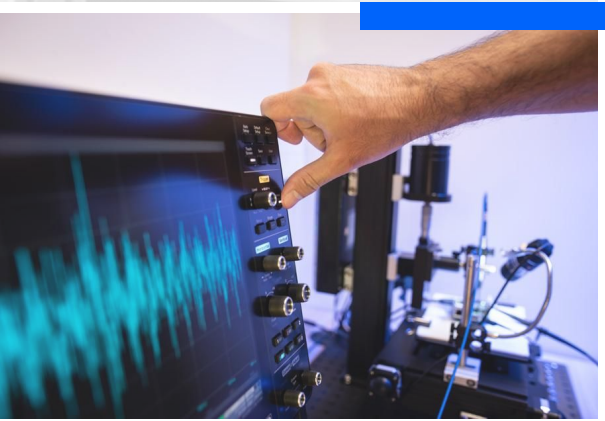
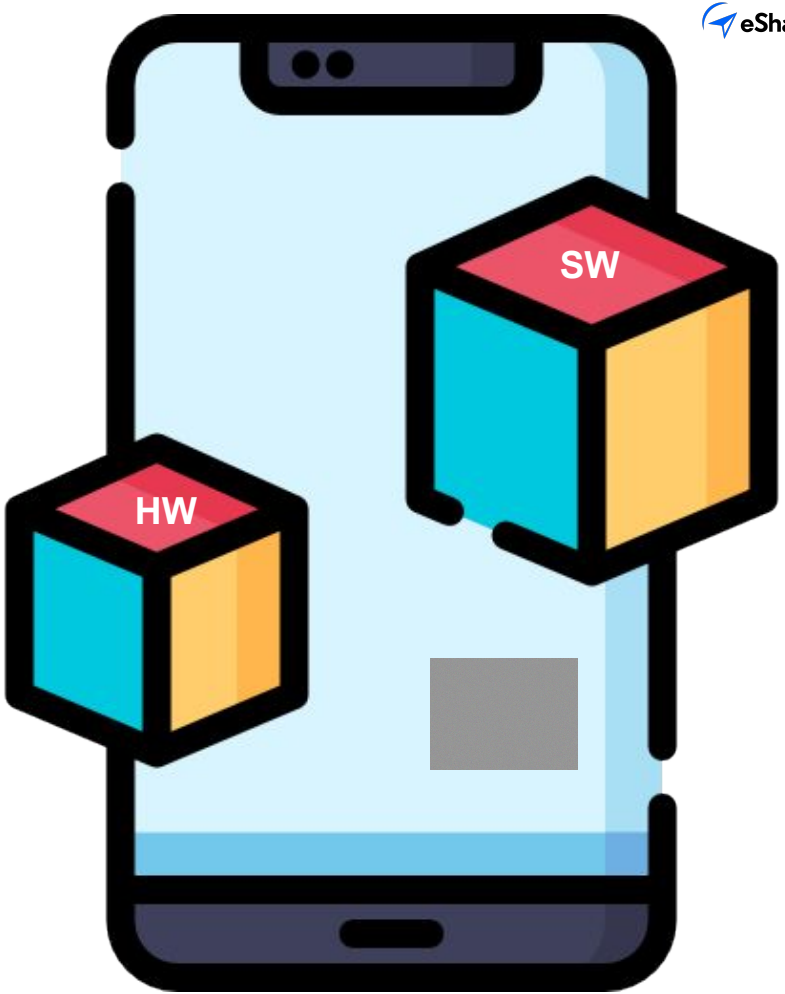
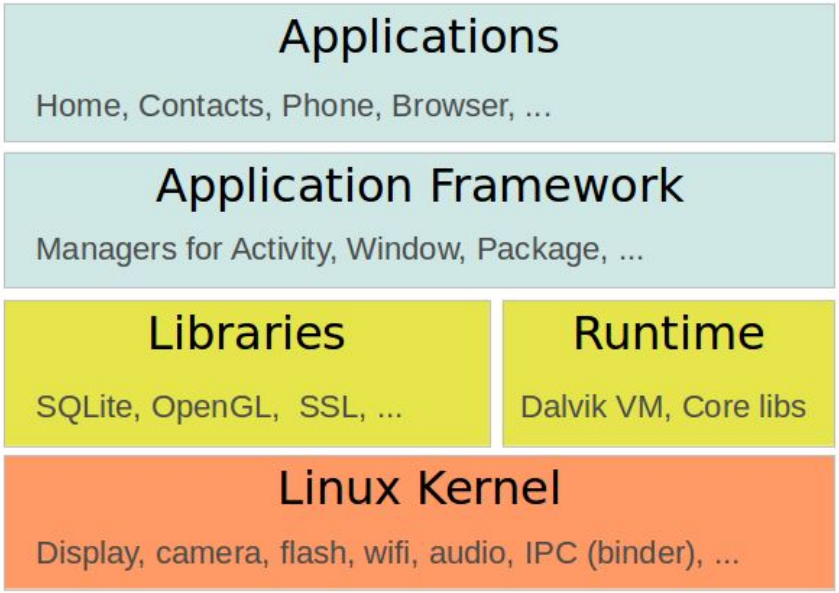


eShard Scaling Security Testing

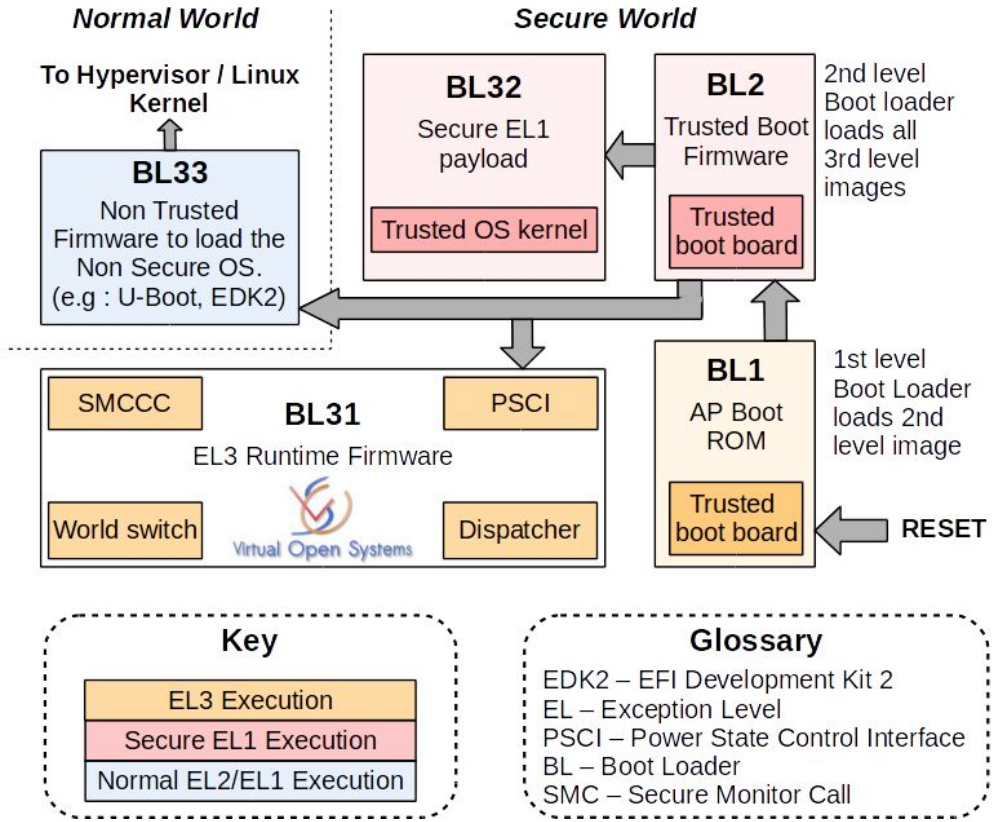
SOC - Spot the Odd Circuit

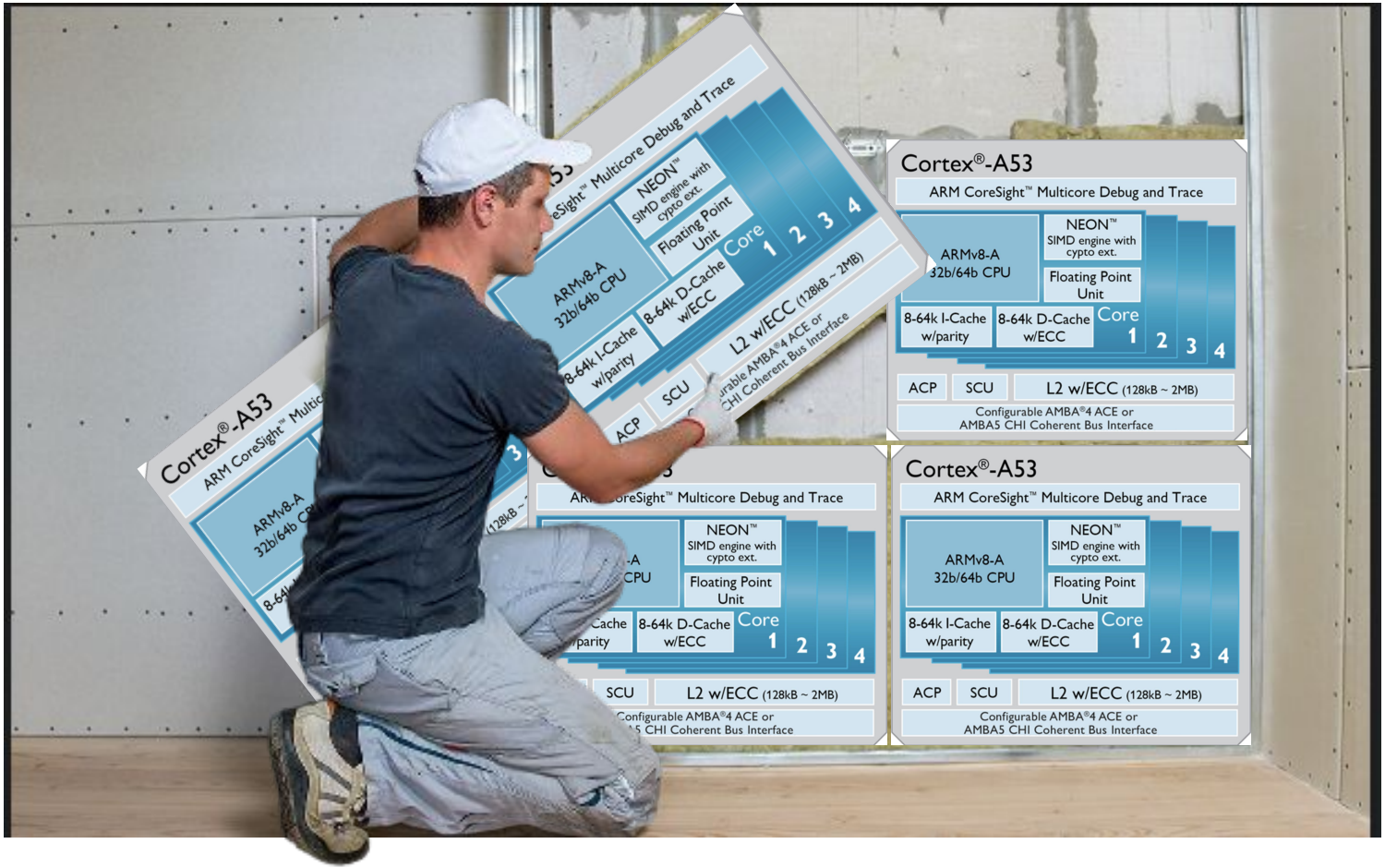

SW
HW

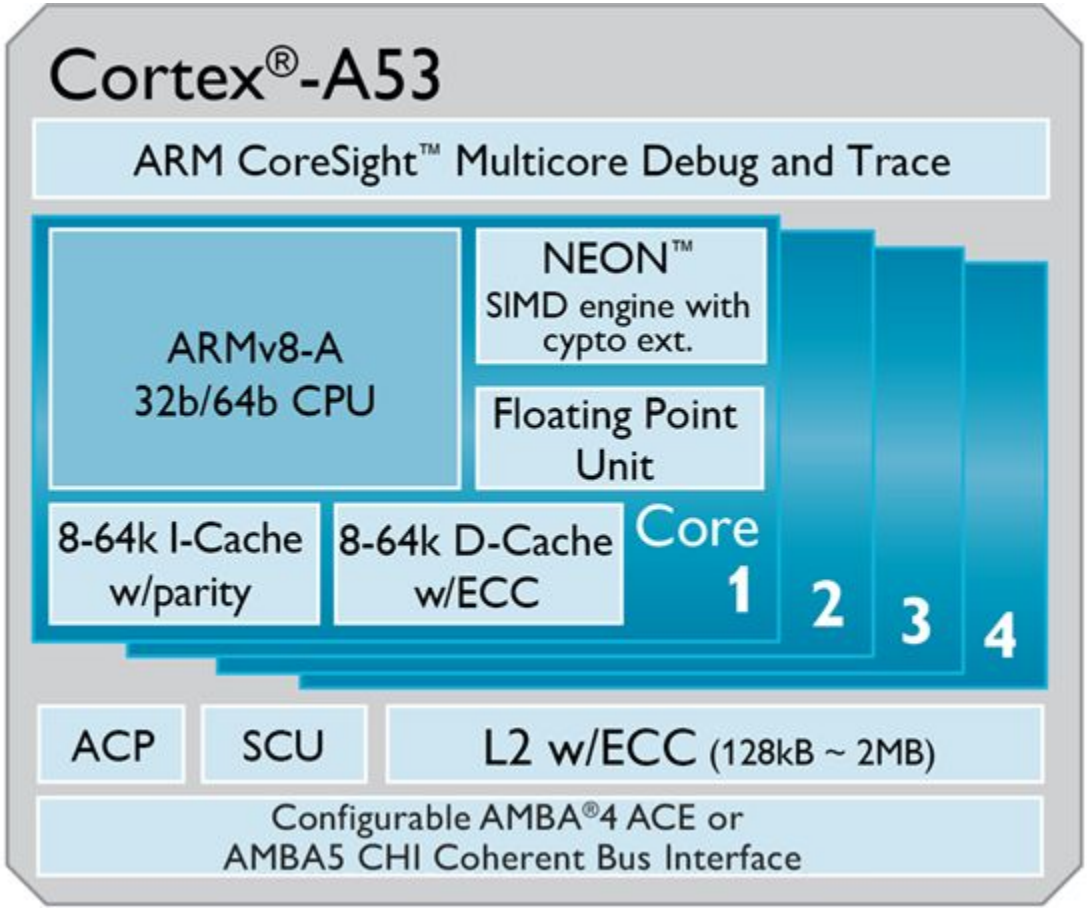


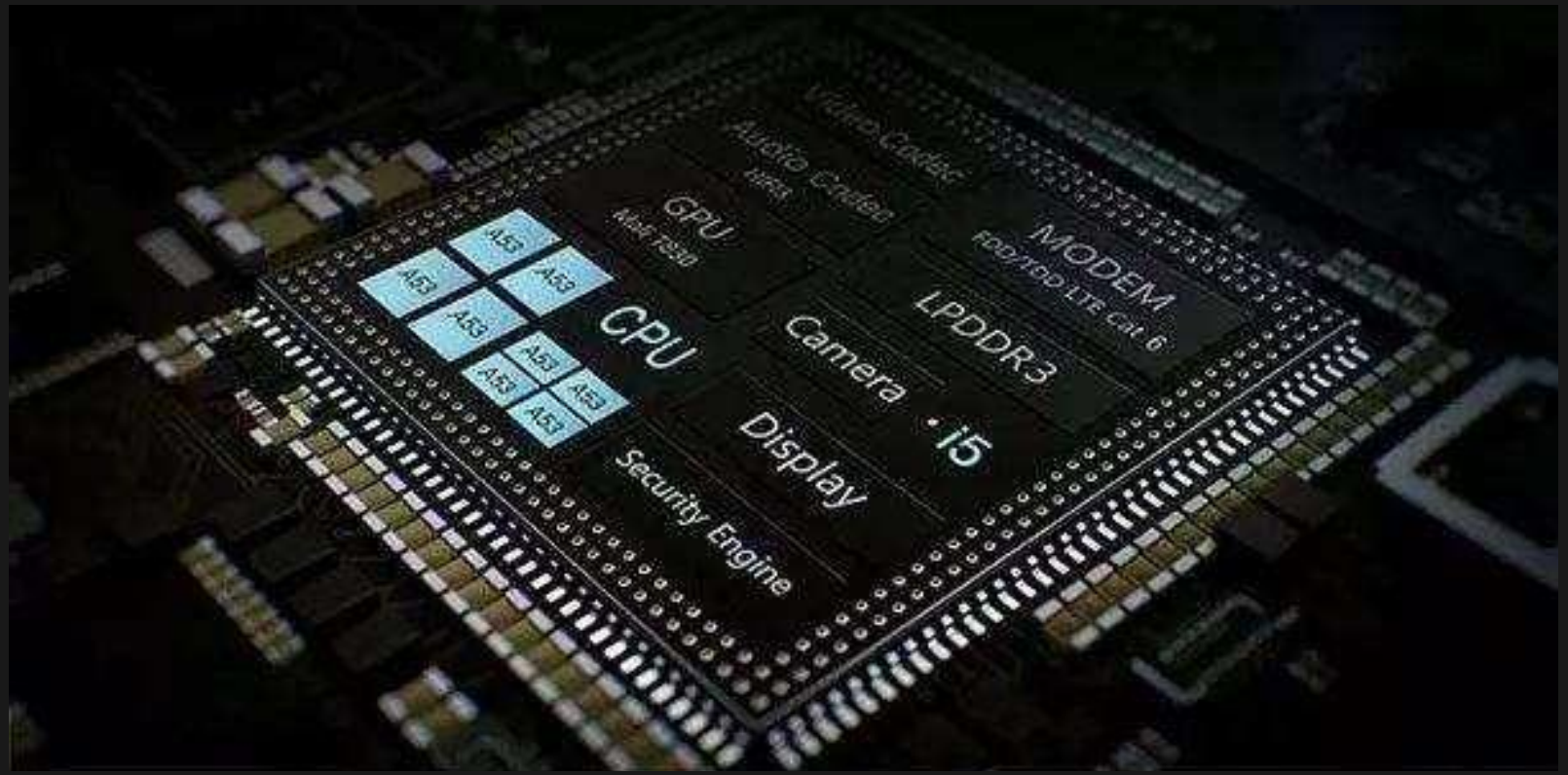
Android Software Stack

Hardware Stack


Normal World
Secure World
To Hypervisor / Linux Kernel
BL33
Non Trusted Firmware to load the Non Secure OS. (e.g : U-Boot, EDK2)
BL32
Secure EL1 payload
Trusted OS kernel
BL2
Trusted Boot Firmware
Trusted boot board
2nd level Boot loader loads all 3rd level images
SMCCC
BL31
PSCI
EL3 Runtime Firmware
World switch
Virtual Open Systems
Dispatcher
BL1
AP Boot ROM
Trusted boot board
1st level Boot Loader loads 2nd level image
RESET
Key
EL3 Execution
Secure EL1 Execution
Normal EL2/EL1 Execution
Glossary
EDK2 – EFI Development Kit 2
EL – Exception Level
PSCI – Power State Control Interface
BL – Boot Loader
SMC – Secure Monitor Call


Cortex®-A53
ARM CoreSight™ Multicore Debug and Trace
ARMv8-A 32b/64b CPU
NEON™ SIMD engine with cypto ext.
Floating Point Unit
8-64k I-Cache w/parity
8-64k D-Cache w/ECC
Core 1 2 3 4
ACP
SCU
L2 w/ECC (128kB ~ 2MB)
Configurable AMBA®4 ACE or AMBA5 CHI Coherent Bus Interface


Cortex®-A53
ARM CoreSight™ Multicore Debug and Trace
ARMv8-A
32b/64b CPU
NEON™
SIMD engine with
cypto ext.
Floating Point
Unit
8-64k I-Cache
w/parity
8-64k D-Cache
w/ECC
Core
1
2
3
4
ACP
SCU
L2 w/ECC (128kB ~ 2MB)
Configurable AMBA®4 ACE or
AMBA5 CHI Coherent Bus Interface


GPU
Mali T830
Audio Codec
HiFi4
Video Codec
MODEM
FDD/TDD LTE Cat 6
LPDDR3
CPU
A53
A53
A53
A53
A53
A53
A53
A53
Camera
i5
Display
Security Engine

Why SCA on a SoC might be Hard

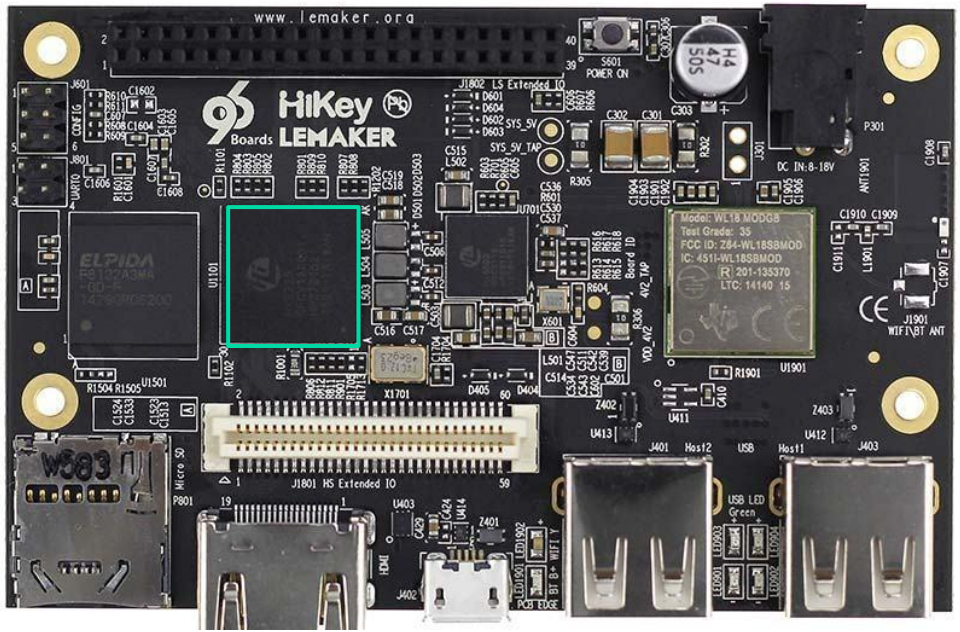
Reality we are facing

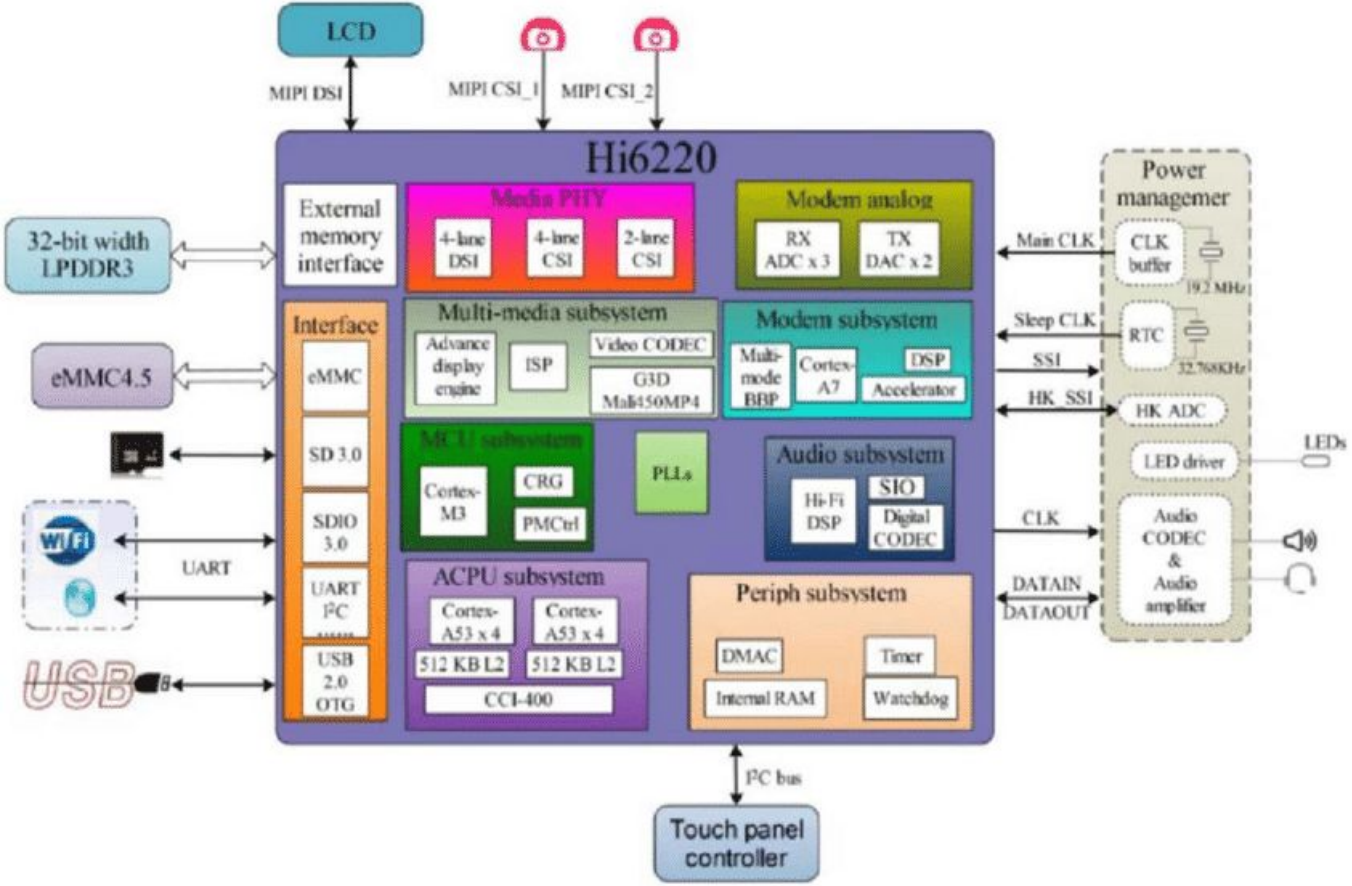
- High Speed more than one Giga Hertz
  - Acquisition chain and setup matters
  - Identify Crypto activity not that simple
- Nanometer Technology
- Two Cache Levels Cache Activities perturbates the acquisition
- Running in a complex OS context
  - Interruption
  - Multiple Crypto options (SW librairies, HW)
  - Several Application Options
- Acces not always easy
- HW AES Enc in 20 cycles …

For all these reasons, studying one of the most deployed core like the ARMv8 is interesting

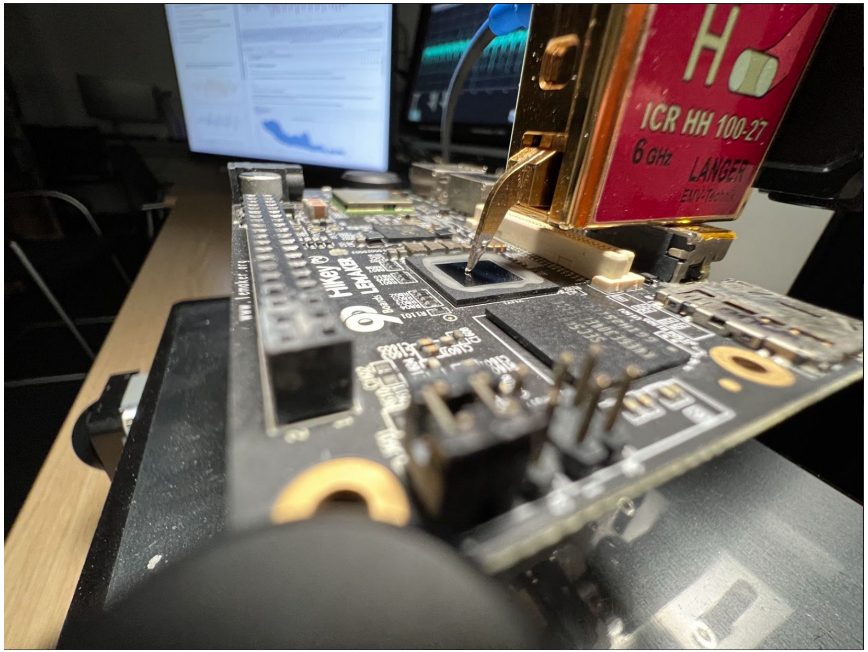
First Target for SCA - Hisilicon Kirin 620 SoC on the Hikey board

8 x ARM © A53 Cores with NEON™ , Freq 1.2Ghz



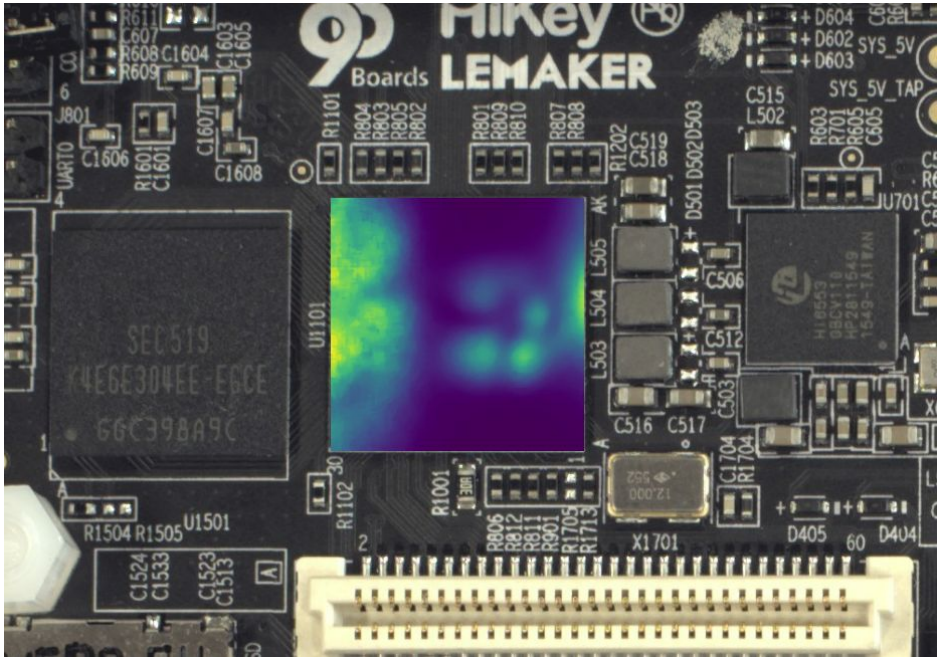

LCD
MIPI DSI
MIPI CSI_1
MIPI CSI_2
Hi6220
External memory interface
Media PHY
4-lane DSI
4-lane CSI
2-lane CSI
Modem analog
RX ADC x 3
TX DAC x 2
32-bit width LPDDR3
Interface
eMMC
SD 3.0
SDIO 3.0
UART I²C
USB 2.0 OTG
Multi-media subsystem
Advance display engine
ISP
Video CODEC
G3D Mali450MP4
Modem subsystem
Multi-mode BBP
Cortex-A7
DSP
Accelerator
eMMC4.5
MCU subsystem
Cortex-M3
CRG
PMCtrl
PLLs
Audio subsystem
Hi-Fi DSP
SIO
Digital CODEC
UART
ACPU subsystem
Cortex-A53 x 4
Cortex-A53 x 4
512 KB L2
512 KB L2
CCI-400
Periph subsystem
DMAC
Timer
Internal RAM
Watchdog
USB
I²C bus
Touch panel controller
Power managemer
Main CLK
CLK buffer
19.2 MHz
Sleep CLK
RTC
32.768KHz
SSI
HK_SSI
HK ADC
LED driver
LEDs
CLK
Audio CODEC & Audio amplifier
DATAIN
DATAOUT

On the SCA Bench ...



EM Activity

While AES encryption, using OpenSSL library



EM Cartography - One active among the eight cores

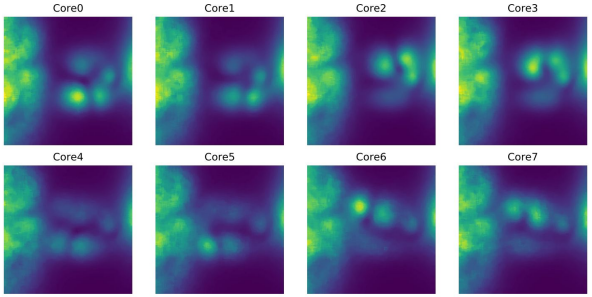


Figure 3.3: Cartography with AES running on each Core. (top) Cluster 0-3 (bottom) Cluster 4-7

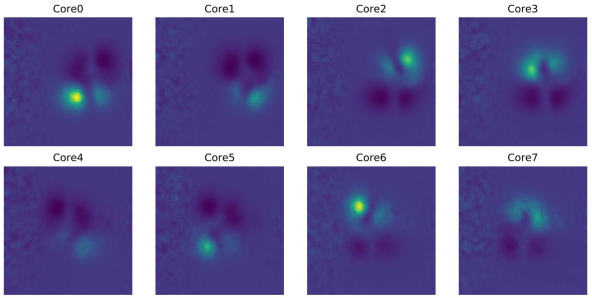


Figure 3.4: Localization of each core by signal amplitude difference

- Core activity is visible on the right, but the PMIC (Power Management Integrated Circuit), on the left side is a source of emission
- This a small signal processing allow to identify the activity of the active core.
- We noticed 2 clusters 0-3 and 4-7 that seems to operate simultaneously
- Core 0, 3, and 6 have a stronger emission

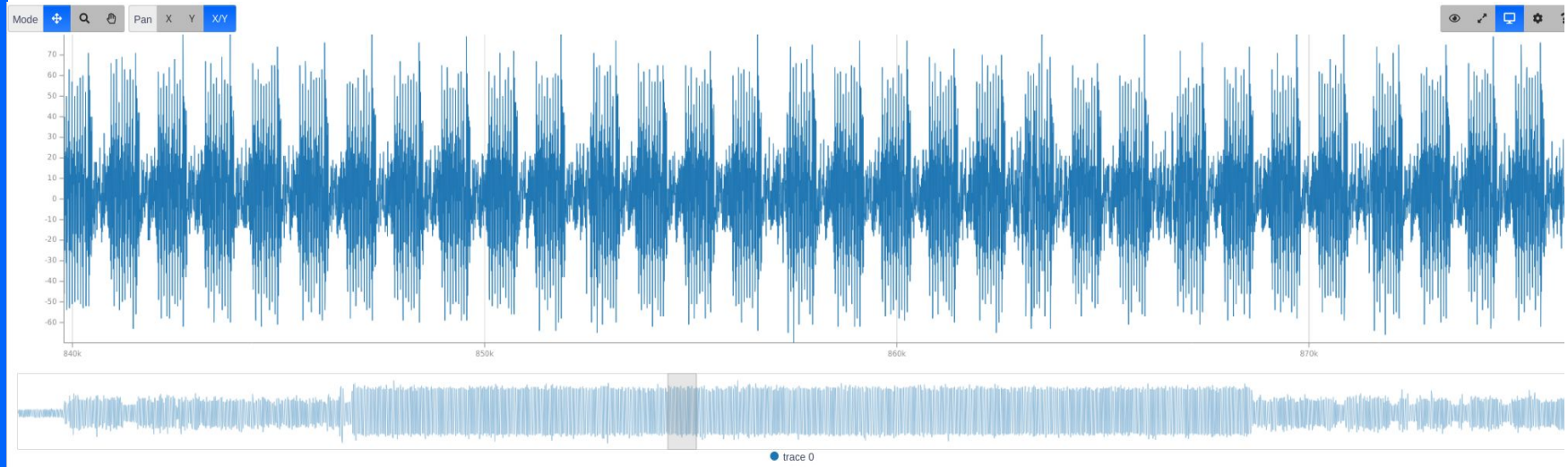
Some results in OpenSSL context

- Mode of operations
- Interruptions
- Cache L1/L2

Mode of Operation Matters

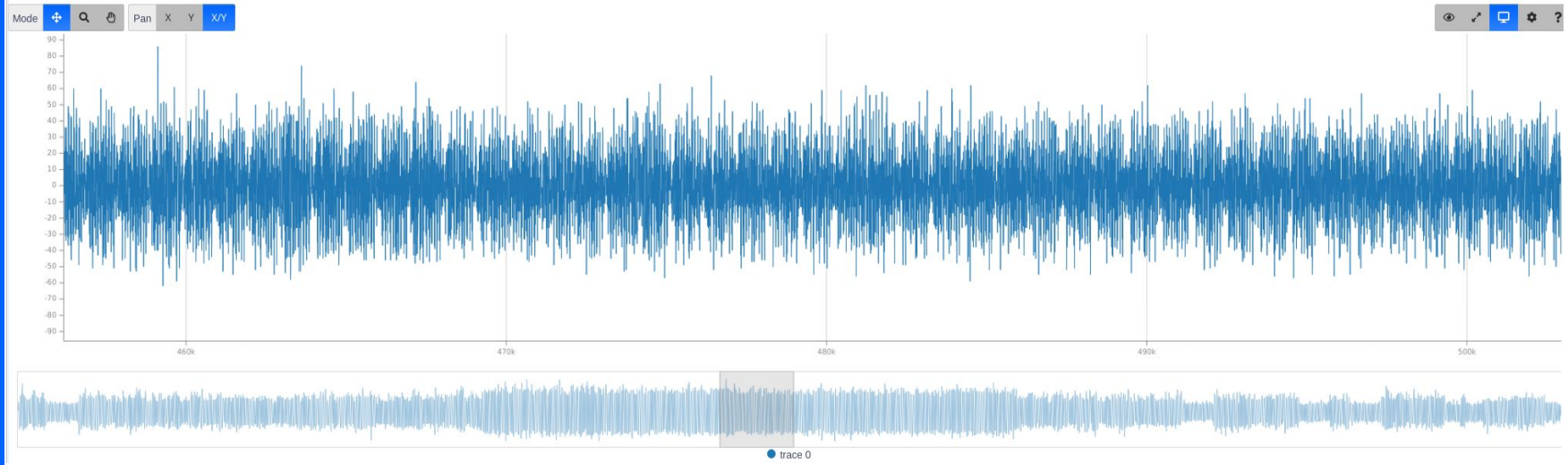
Mode of Operation

ECB



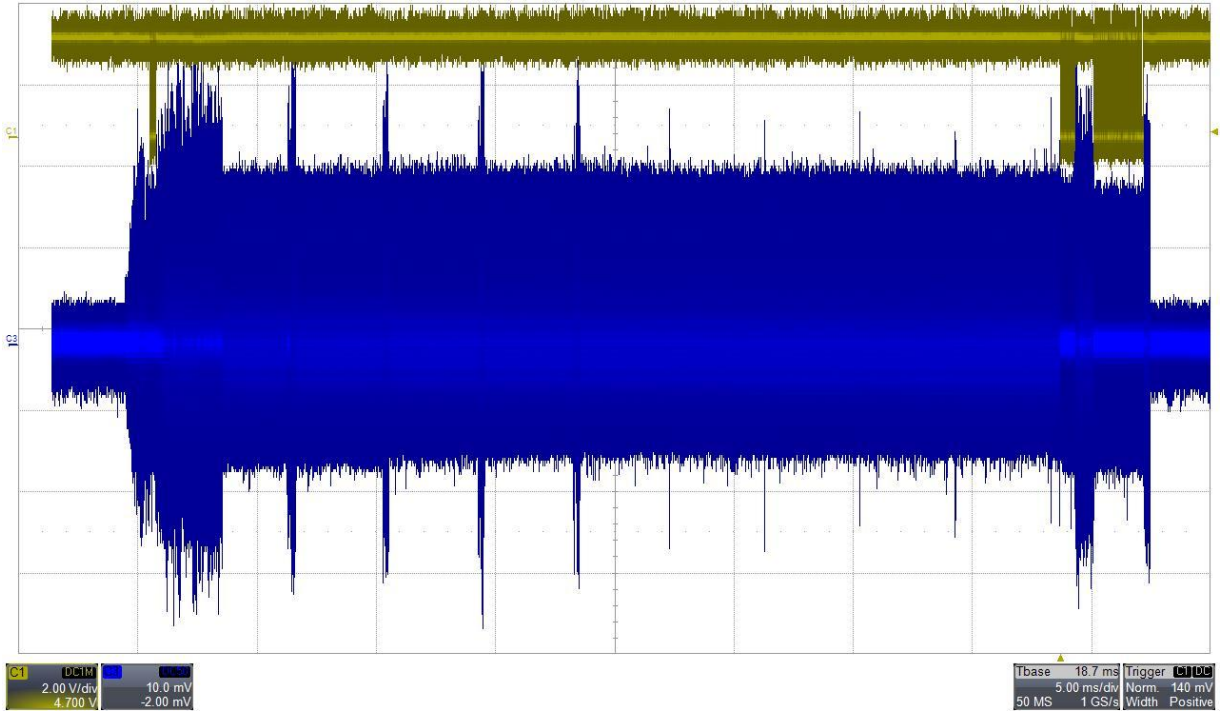
Mode of Operation

CBC



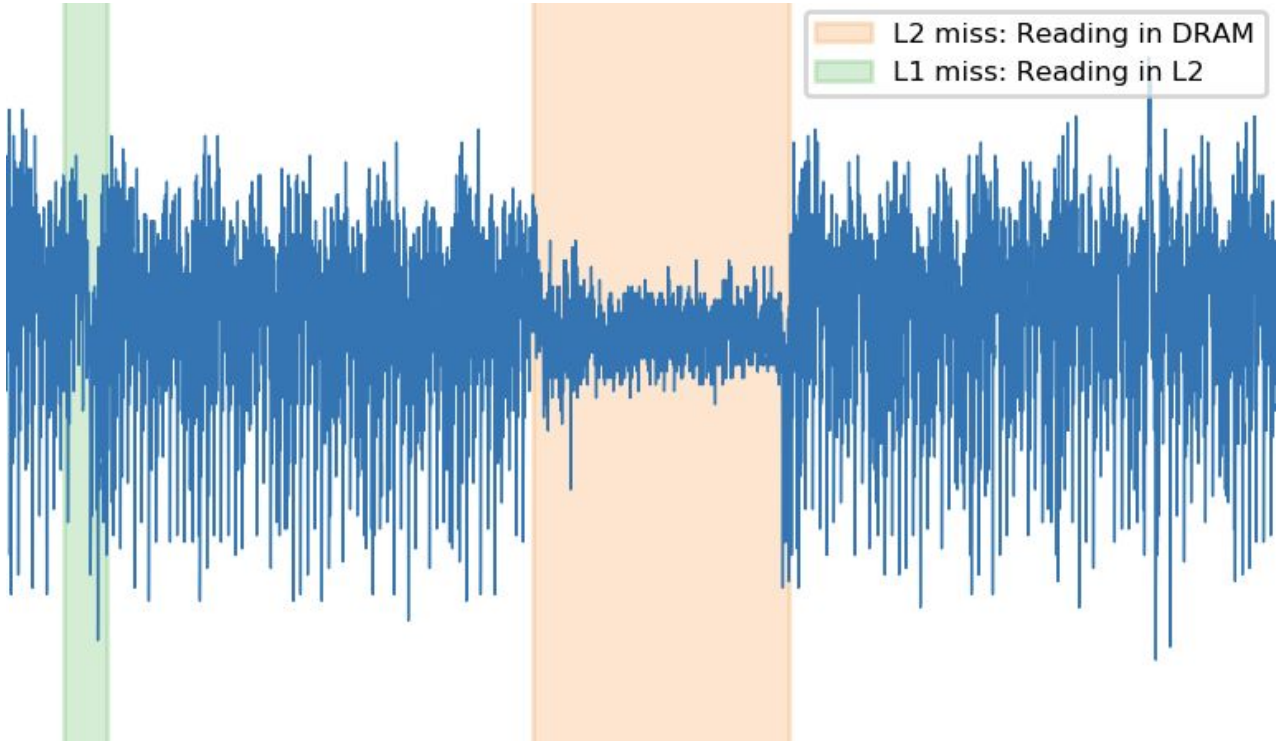
At OS Level…

Interruptions, every 4ms

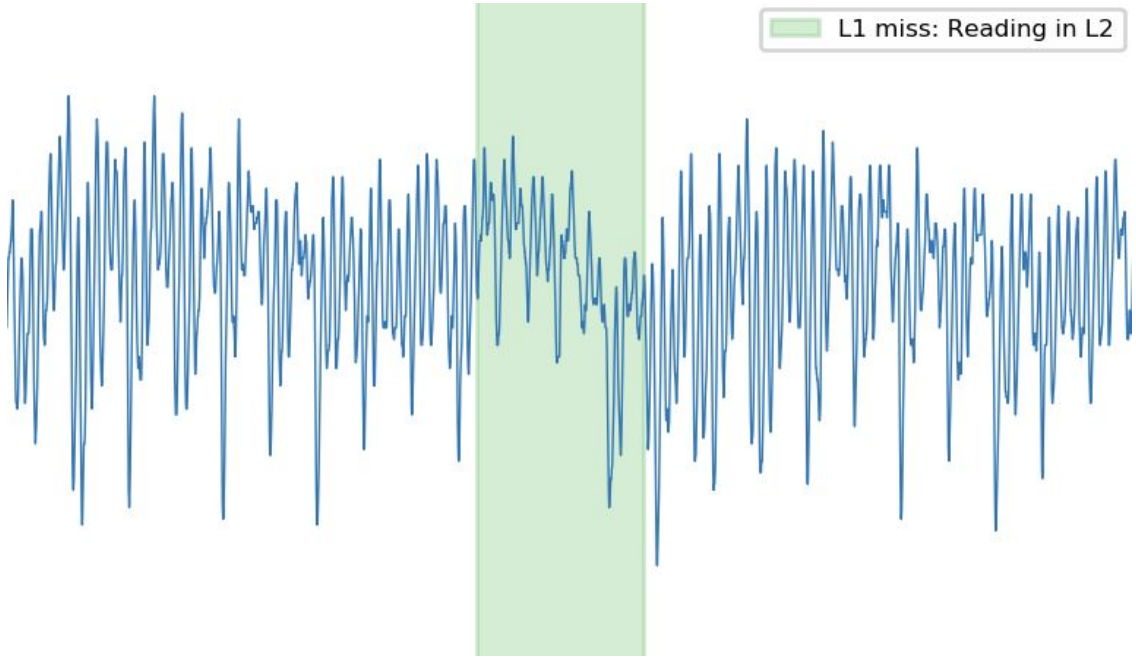


Memory Operations - Cache Effects

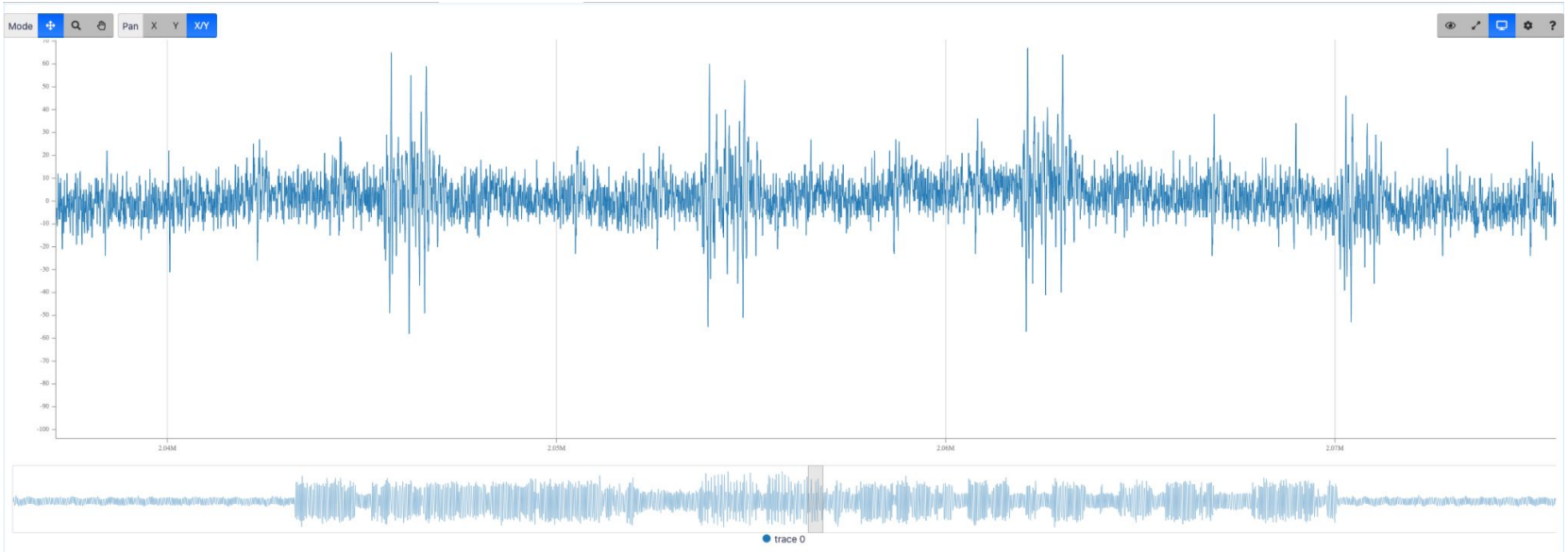
Cache Miss



L1 Miss Read in L2



But, … cache activity (here L1) can be identified



Attack Strategy - Divide and Conquer (always ...)

Let's simplify the work, then scale-up … may the Force be with us!

Classical Approach

Simplify but not too much

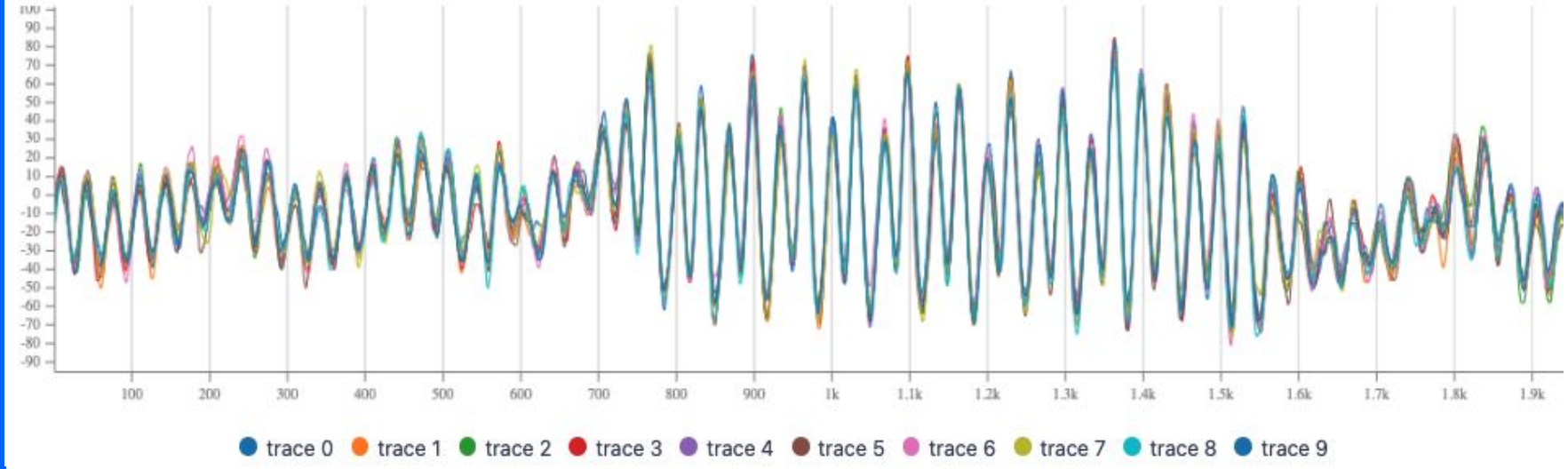
- Known Inputs or Output of targeted Algorithm
- Known Code
- Control the Core(s) where the AES is operated
- Add triggers to ease identification

```
1   func decrypt
2           ldr     x3, =0xf9818010       // Pointer to input data
3           ld1     { v0.16B }, [x3]      // Loading from memory to NEON register
4
5           aesd    v0.16B, v11.16B       // First Round decrypt
6           aesimc  v0.16B, v0.16B        // Inverse MixColumns
7           aesd    v0.16B, v10.16B
8           aesimc  v0.16B, v0.16B
9           aesd    v0.16B, v9.16B
10          aesimc  v0.16B, v0.16B
11          aesd    v0.16B, v8.16B
12          aesimc  v0.16B, v0.16B
13          aesd    v0.16B, v7.16B
14          aesimc  v0.16B, v0.16B
15          aesd    v0.16B, v6.16B
16          aesimc  v0.16B, v0.16B
17          aesd    v0.16B, v5.16B
18          aesimc  v0.16B, v0.16B
19          aesd    v0.16B, v4.16B
20          aesimc  v0.16B, v0.16B
21          aesd    v0.16B, v3.16B
22          aesimc  v0.16B, v0.16B
23          aesd    v0.16B, v2.16B
24          eor     v0.16B, v0.16B, v1.16B // Last AddRoundKey
25
26          st1     { v0.16B }, [x3]       // Storing result in-place
27          ret
28  endfunc decrypt
```

Simple AES bare-metal

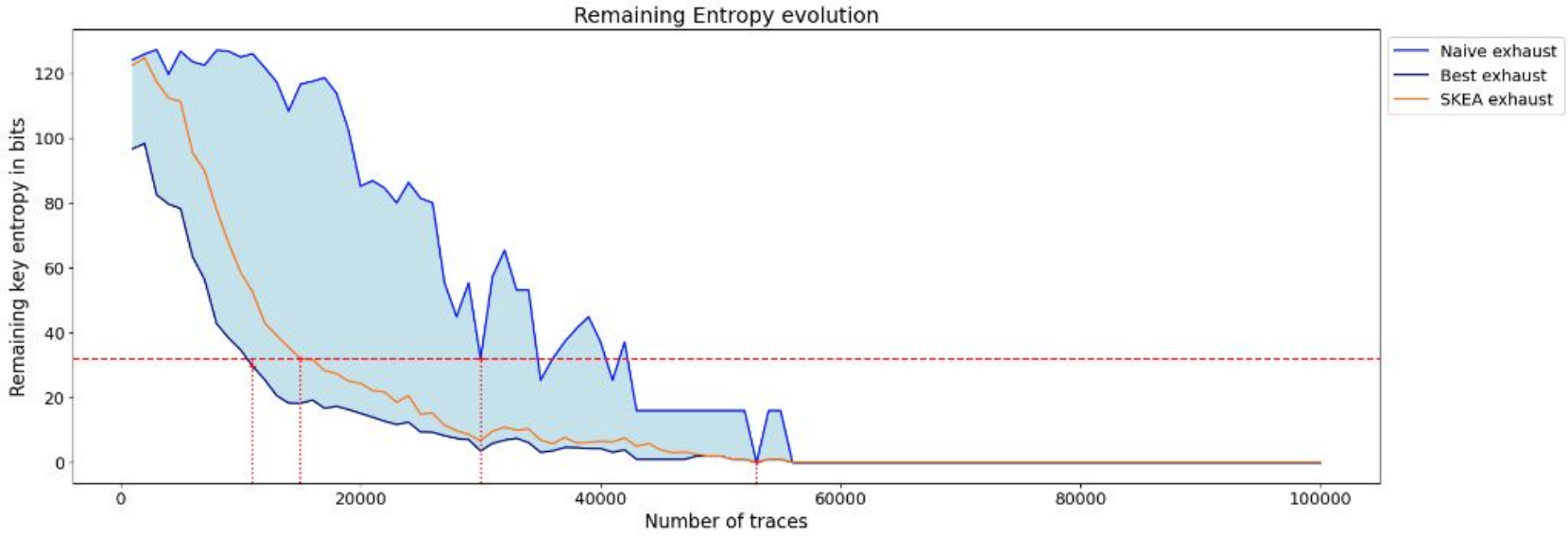
A Single AES 10 traces synchronized

Looks good!



Attack Result

Remaining Key Entropy evolution - Single Core



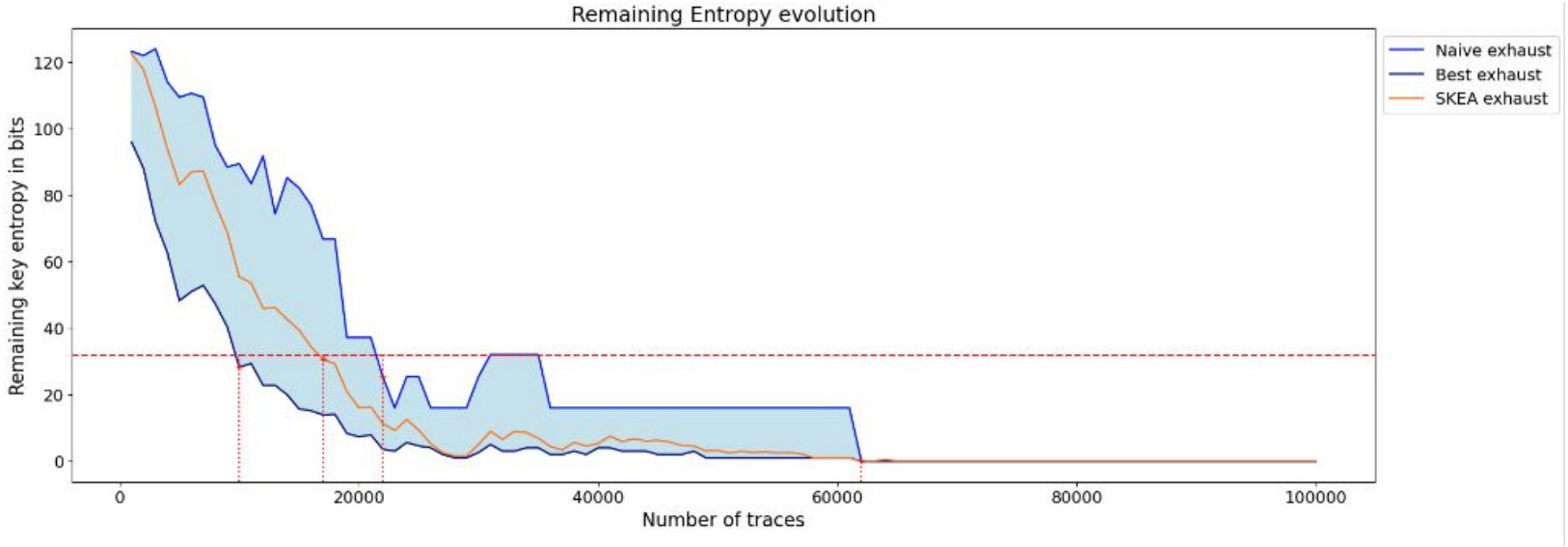
Pipeline Effect ?

```
func decrypt_pipelined
        ldr x3, =0xf9818010         // Pointer to input data
        ld1 { v12.16B }, [x3], #16  // Loading from memory to NEON register
        ld1 { v13.16B }, [x3], #16  // 3 blocks at a time
        ld1 { v14.16B }, [x3], #16

        aesd    v12.16B, v11.16B    // Round 0
        aesimc  v12.16B, v12.16B
        aesd    v13.16B, v11.16B
        aesimc  v13.16B, v13.16B
        aesd    v14.16B, v11.16B
        aesimc  v14.16B, v14.16B

        aesd    v12.16B, v10.16B    // Round 1
        aesimc  v12.16B, v12.16B
        aesd    v13.16B, v10.16B
        aesimc  v13.16B, v13.16B
        aesd    v14.16B, v10.16B
        aesimc  v14.16B, v14.16B

        .../...

        aesd    v12.16B, v3.16B
        aesimc  v12.16B, v12.16B
        aesd    v13.16B, v3.16B
        aesimc  v13.16B, v13.16B
        aesd    v14.16B, v3.16B
        aesimc  v14.16B, v14.16B

        aesd    v12.16B, v2.16B     // Last Round
        aesd    v13.16B, v2.16B
        aesd    v14.16B, v2.16B
        eor     v12.16B, v12.16B, v1.16B
        eor     v13.16B, v13.16B, v1.16B
        eor     v14.16B, v14.16B, v1.16B

        ldr x3, =0xf9818010         // Reload pointer
        st1 { v12.16B }, [x3], #16  // Storing result in-place
        st1 { v13.16B }, [x3], #16
        st1 { v14.16B }, [x3], #16
        ret
endfunc decrypt_pipelined
```

Pipeline 3 AES bare-metal

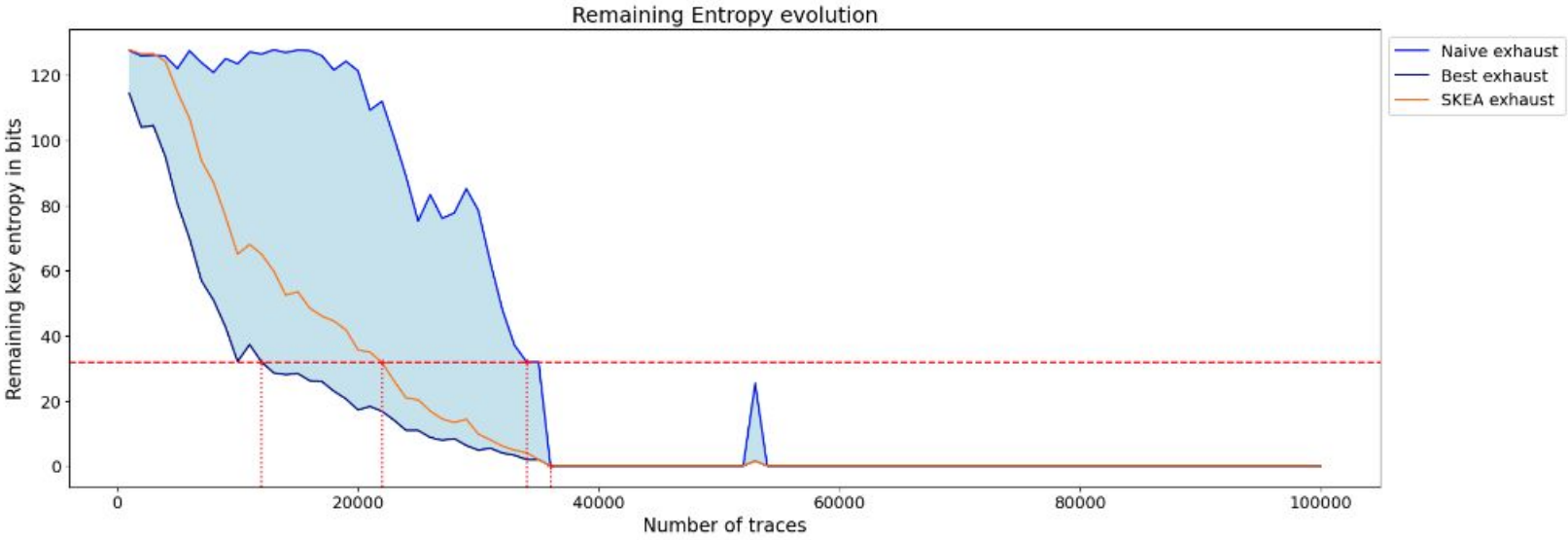
Attack Result

Pipeline by 3 effect target the first core execution



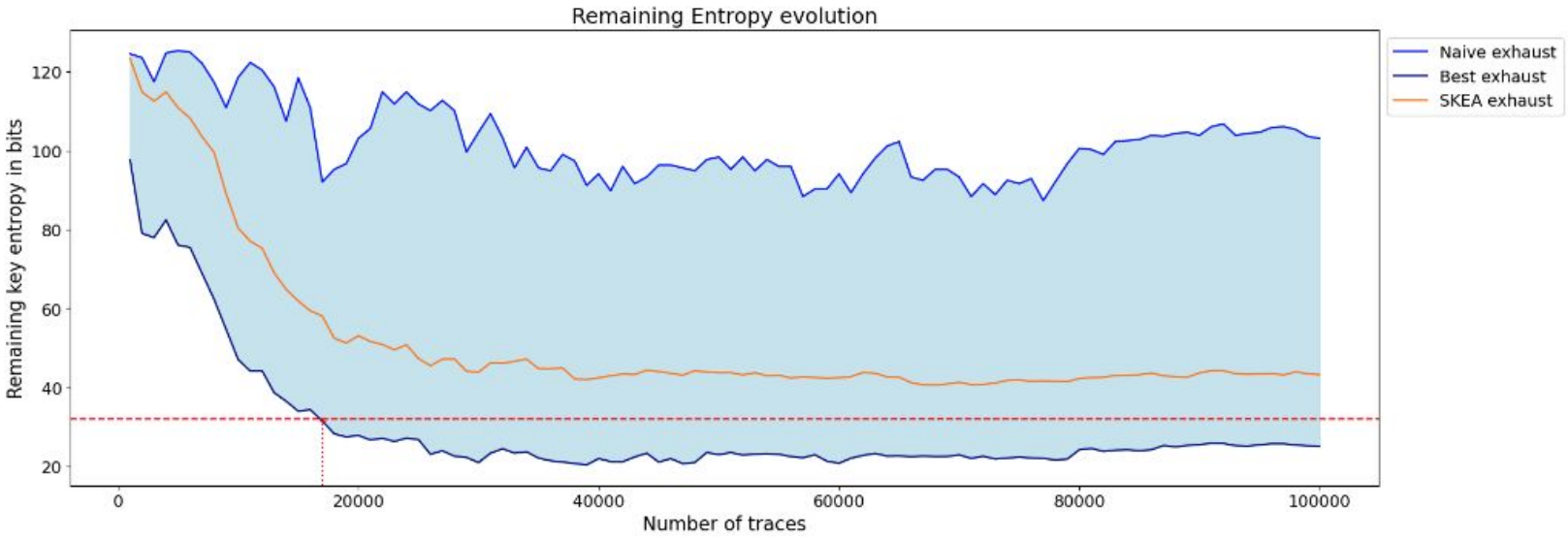
Attack Result

Pipeline by 3 effect, target the second core execution

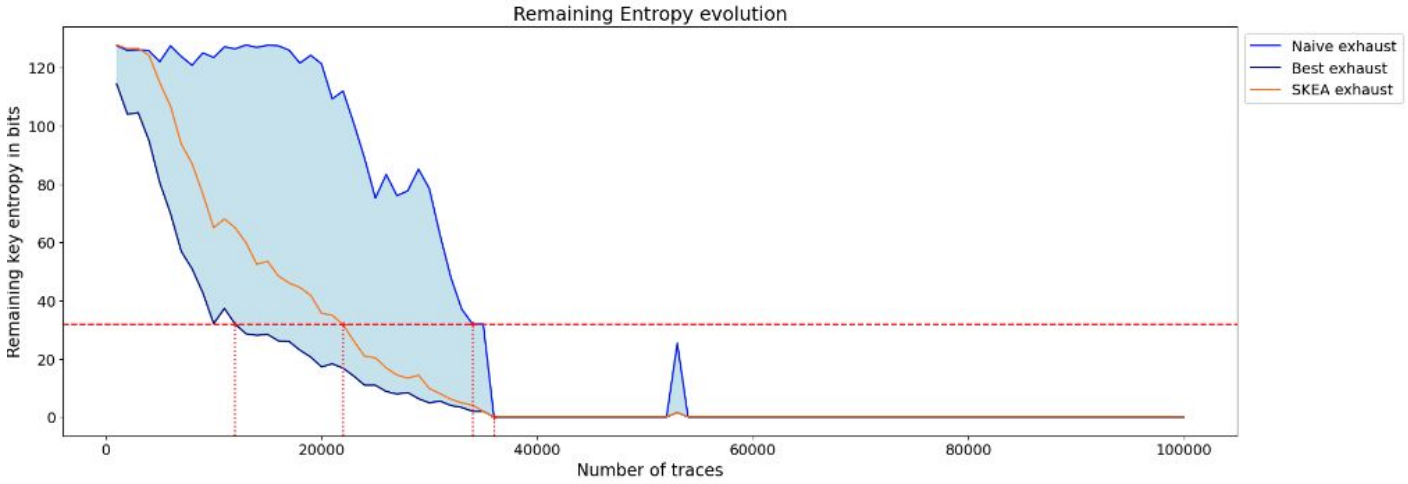


Attack Result

Pipeline by 3 effect, target the third core execution



Attack Result



With the proper Selection Function the third AES can be more easy to break

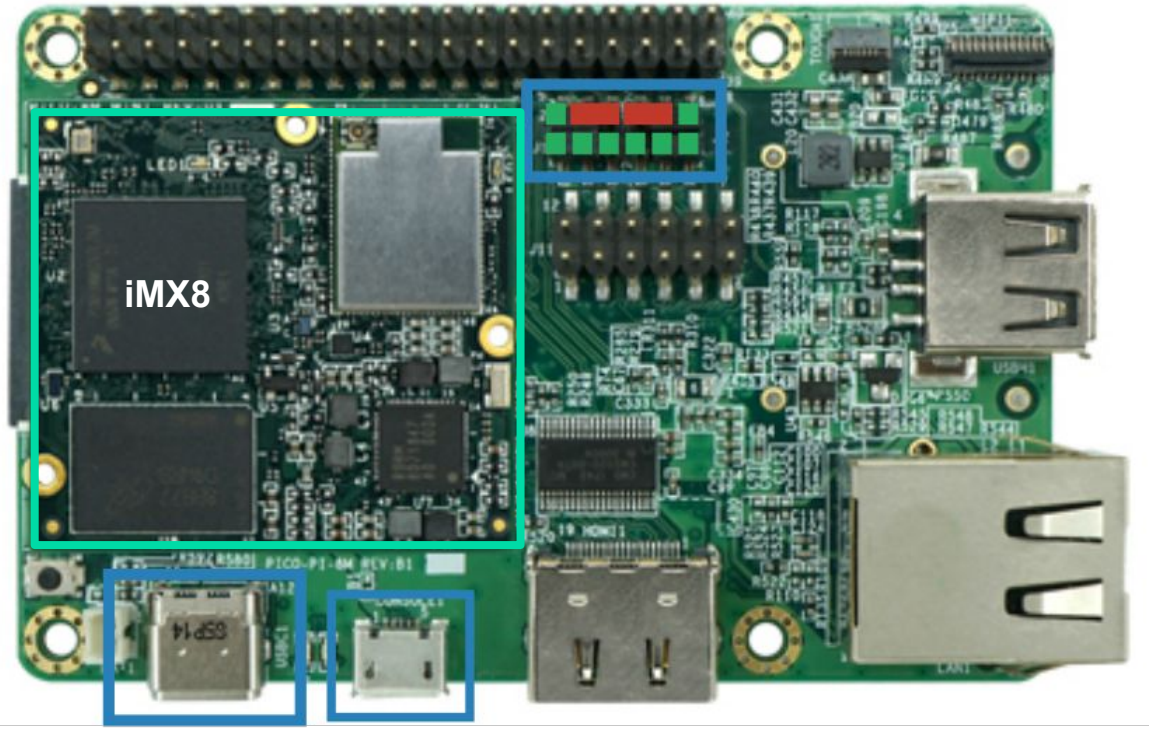
Result

- OS: Capture between interruptions
- Cache: use cache activity as a helper, or remove bad traces
- Pipeline : Leverage architecture and code knowledge to define the proper Selection Function

- Litterature says 4 Million trace to attack an A53 we render with less than 40k or 20k to get 32-bit remaining entropy and get the whole key with SKEA

Faults?

Another SoC to play with : iMX8 mini



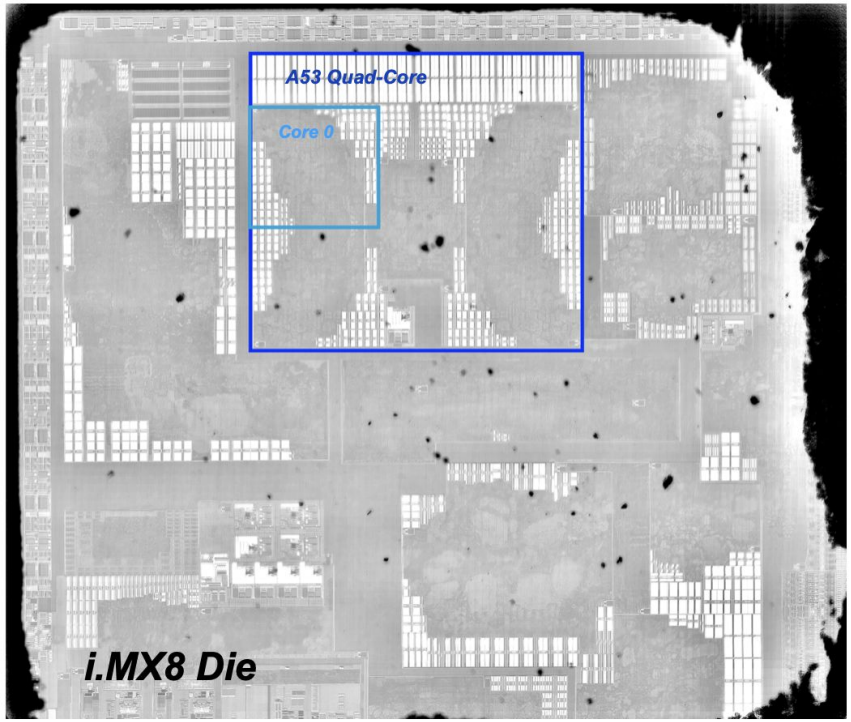
Product description iMX8 mini

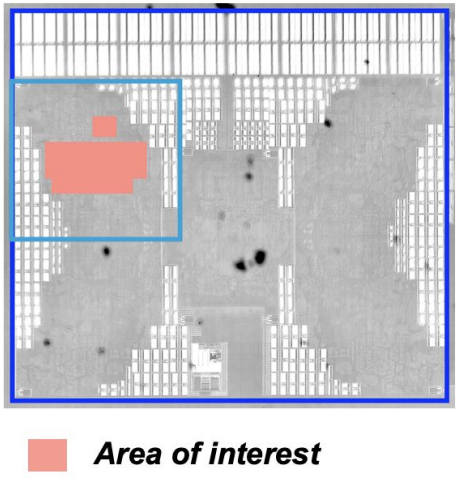
- Motherboard Technexion PICO-PI-8M PICO-IMX8M-
- Technology 14nm LPC FinFET
- SoC: NXP **4x Cortex-A53** core platforms up to 1.8GHz per core
- Caches
  - 32kB L1-I Cache/ 32 kB L1-D Cache
  - 512kB L2 Cache

Methodology

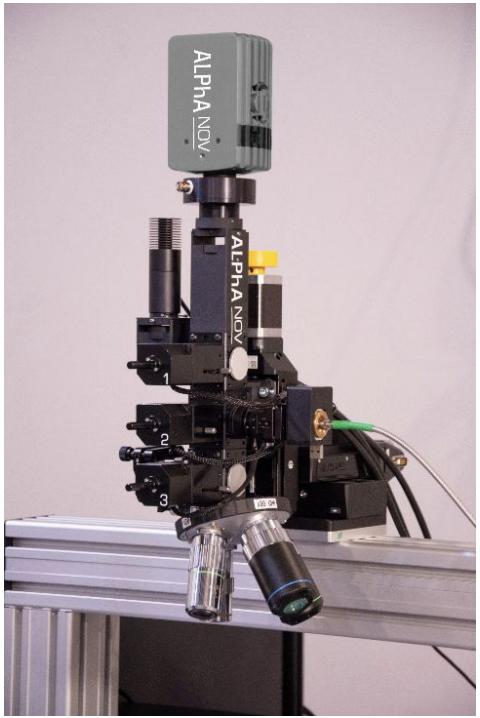
- Software stack analysis from OS to crypto routine call, understand how to reduce architecture effect.
- Dedicated software (bare-metal) for characterization.
- PhotoEmission to find area of activity and refine area of interest.
- Simulation vs IRL comparison to explain or anticipate fault and leakage model.
- Side-Channel - Signal analysis to adjust timing and refine trigger, shooting delay ...

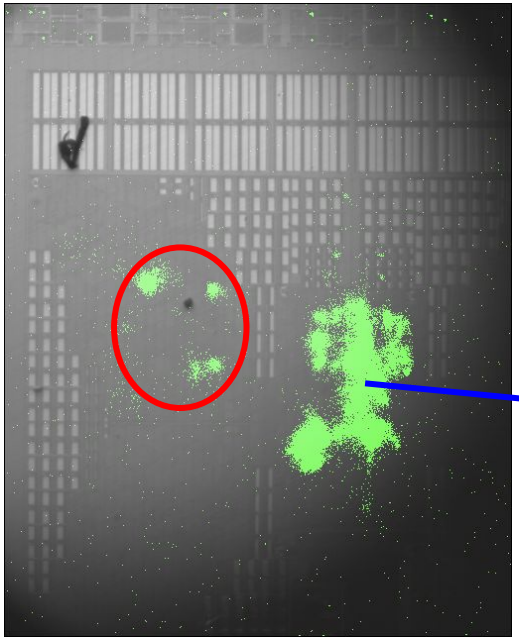
Area Of Interest from EM-SCA





Area of Interest from Photonic Emission

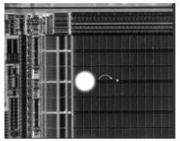


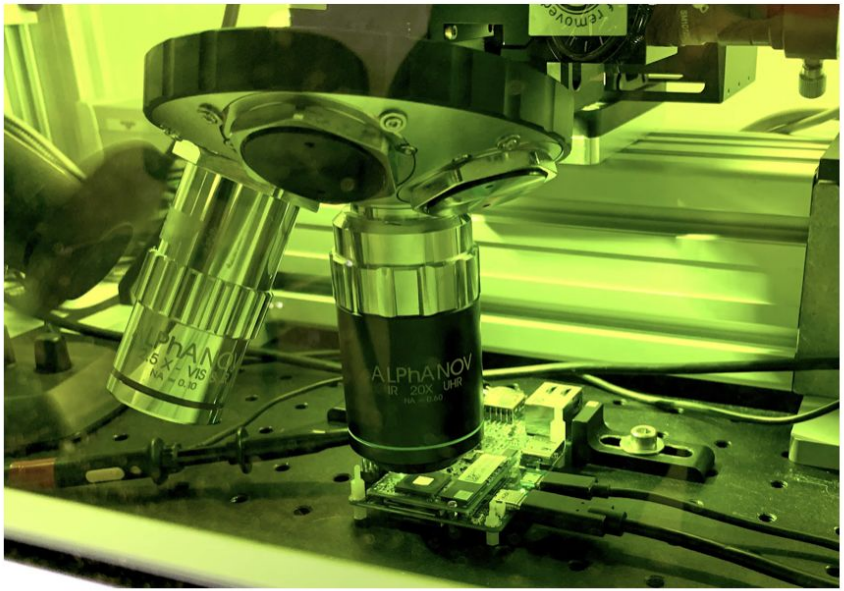


Principle Capture lots of images and stack them up to get highlighted activity by image processing.

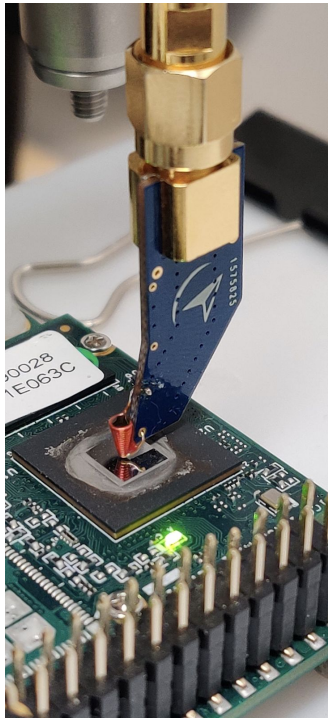
Not interesting Area of cores Interconnect, (buses, I/O,...)

Material for Laser Fault Injection



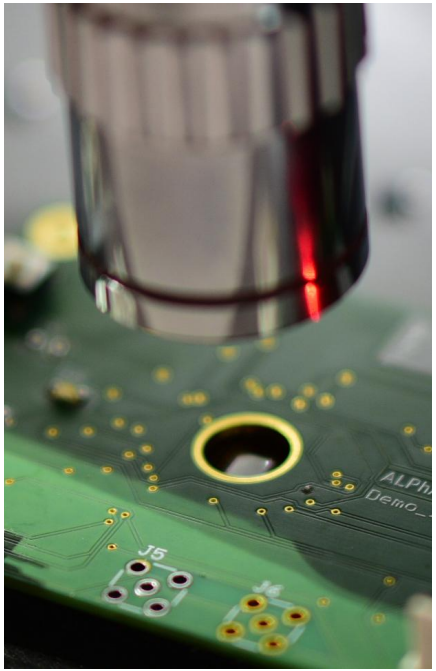


Example of Fault Injection Parameter set



EMFI Parameter		Range		Step (Coarse)	Step (Fine)	Unit
X pos	↔	1000	2400	50 or 100	10 or 25	µm
Y pos	↕	2200	3400	50 or 100	10 or 25	µm
Pulse amplitude	⚡	300	700	50 or 100	10	V
Pulse width	⏳	5	20	5	1	ns
Pulse delay	⏱	300	500	10	1	µs
Repeat	🔁	-	step	1	5	#

LFI Parameter		Range		Step (Coarse)	Step (Fine)	Unit
X pos	↔			10 or 6	3, 2, 1	µm
Y pos	↕			"	"	µm
Optispot	👁	0	1500	100	10	µm
PDM amplitude	⚡	20	80	20	10	%
Pulse width	⏳	100	1000	200	1	ns
Pulse delay	⏱	6875	8875	100	-	ps
Repeat	🔁	0	step	1	5	#



Fault Injection Exploitation

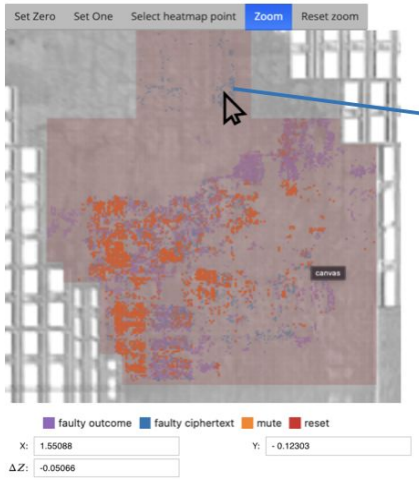


Figure: Fault Visualizer Widget - Click on pts shows pie chart.

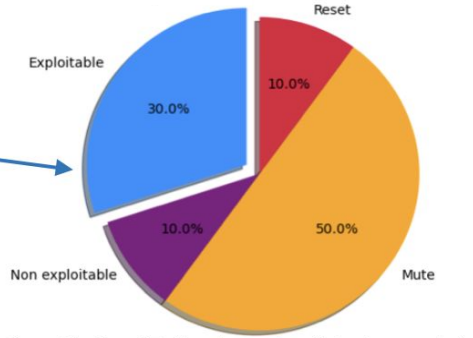


Figure: Pie Char of Faulty outcomes (no effect not represented)

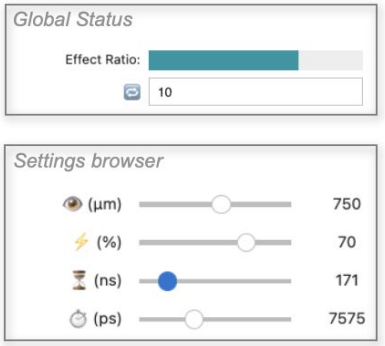


Figure: Browse results according to parameters

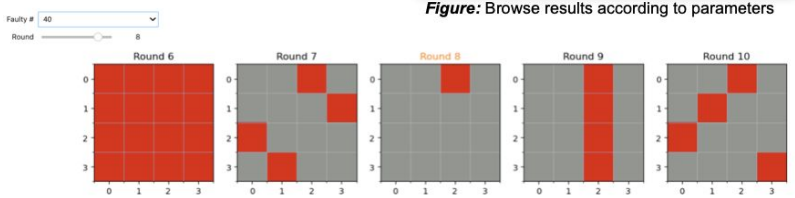


Figure: Fault effect, back tracking knowing the key


Leffe

Thank you.
Any questions?

contact@eshard.com

www.eshard.com

/company/eshard

@eshard

France HQ

Bâtiment GIENAH
11 avenue de Canteranne
33600 Pessac, France

France R&D

7 rue Gaston de Flotte
13012 Marseille, France

Germany

eShard GmbH
Beethovenallee 21
53173 Bonn