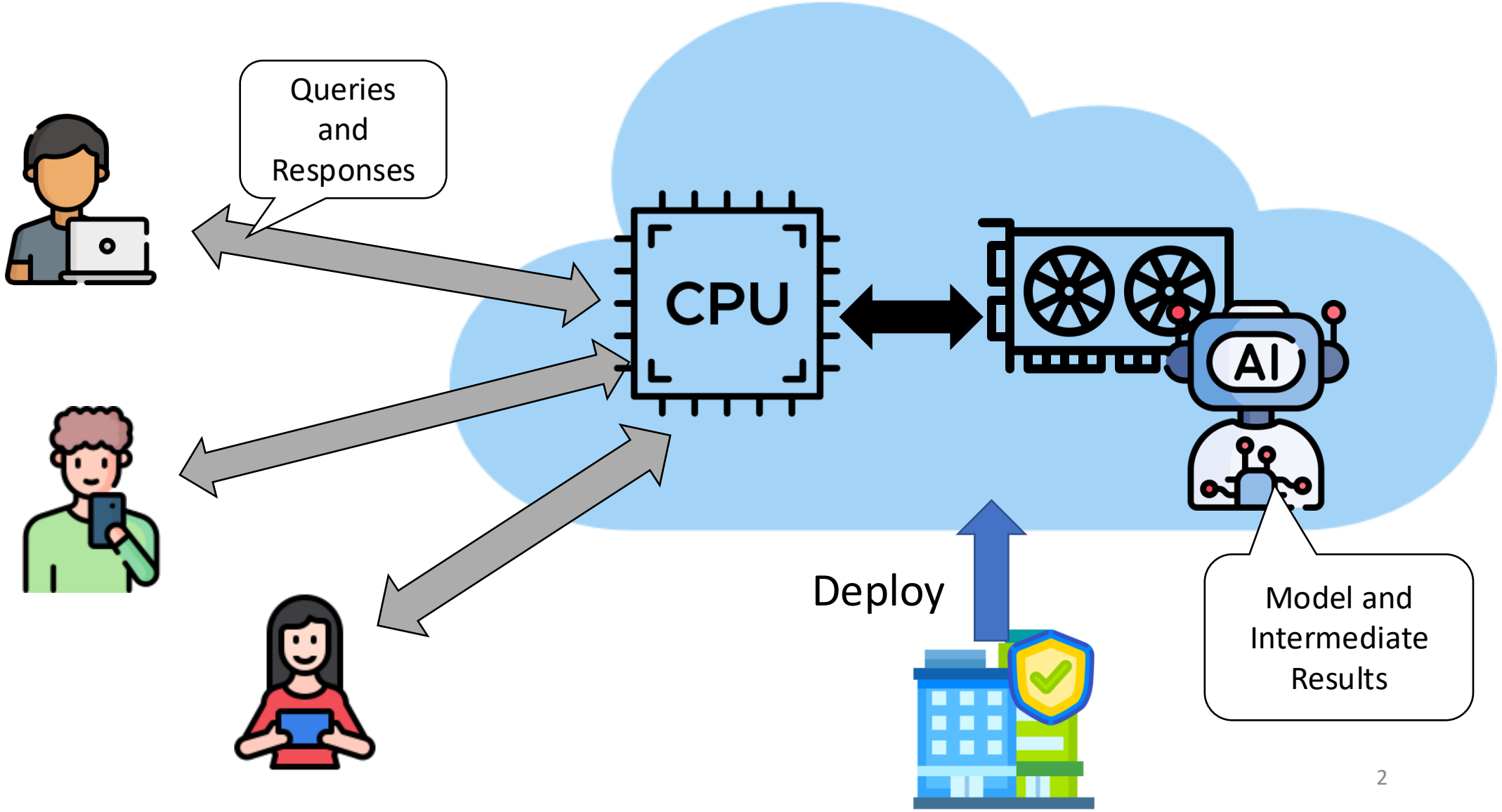
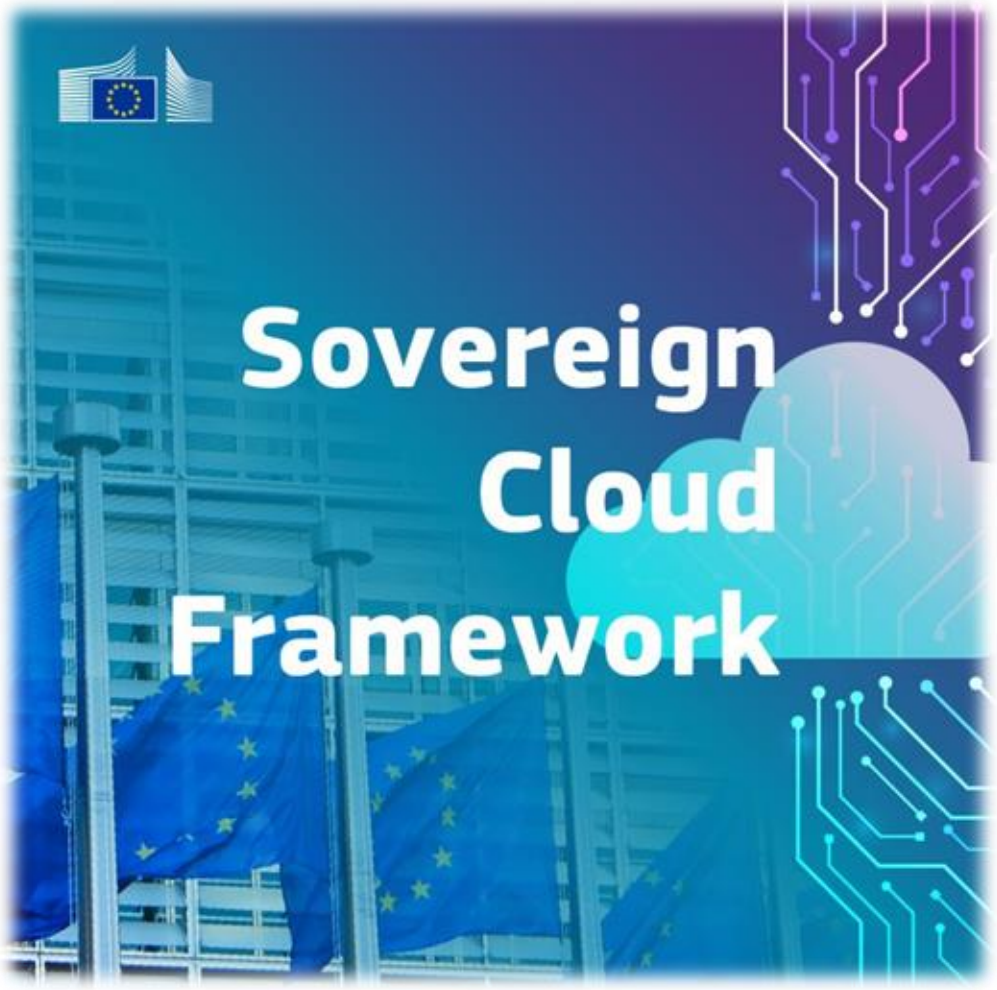
# From Smartphones to Cloud: Trustworthy Foundations for Digital Sovereignty

Shweta Shinde
ETH Zurich

# Example: LLM Chatbot/Agent






Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Der Bundesrat

Bern, 26. November 2025

**Digitale Souveränität der Schweiz**

Bericht des Bundesrates
in Erfüllung des Postulats 22.4411
Z'graggen, 14.12.2022

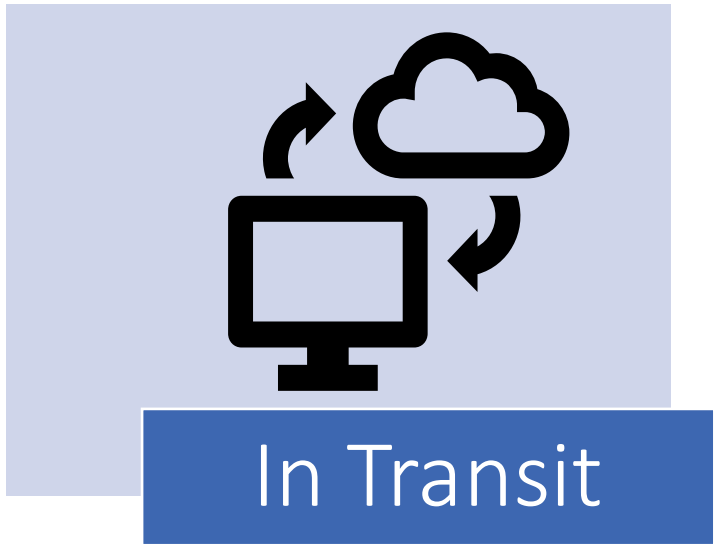
***Digital sovereignty*** means neither self-sufficiency nor isolation, but rather the ability of public authorities to manage and *control the use* of external digital services *independently*.

# Who truly controls our data & computation?

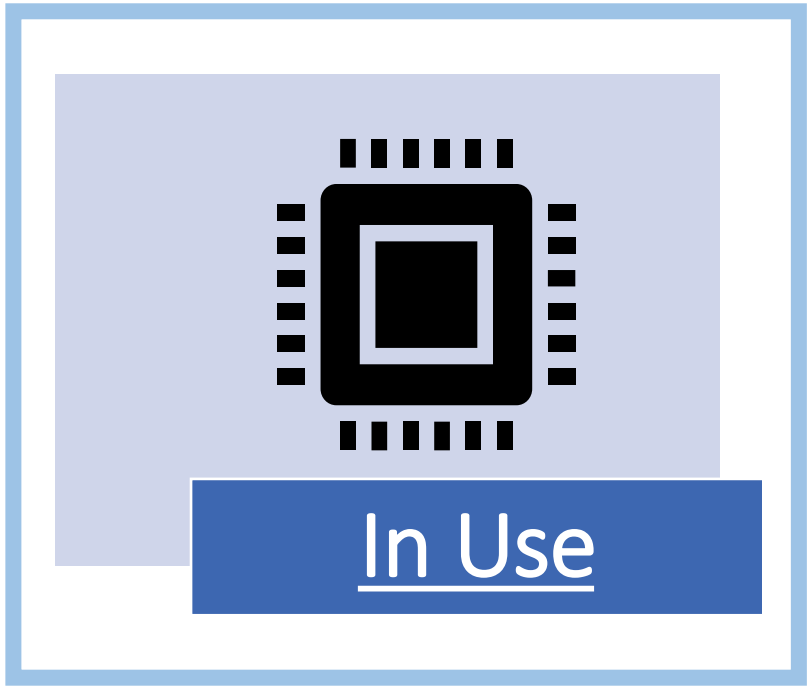
# Attack surfaces for confidential data



At rest

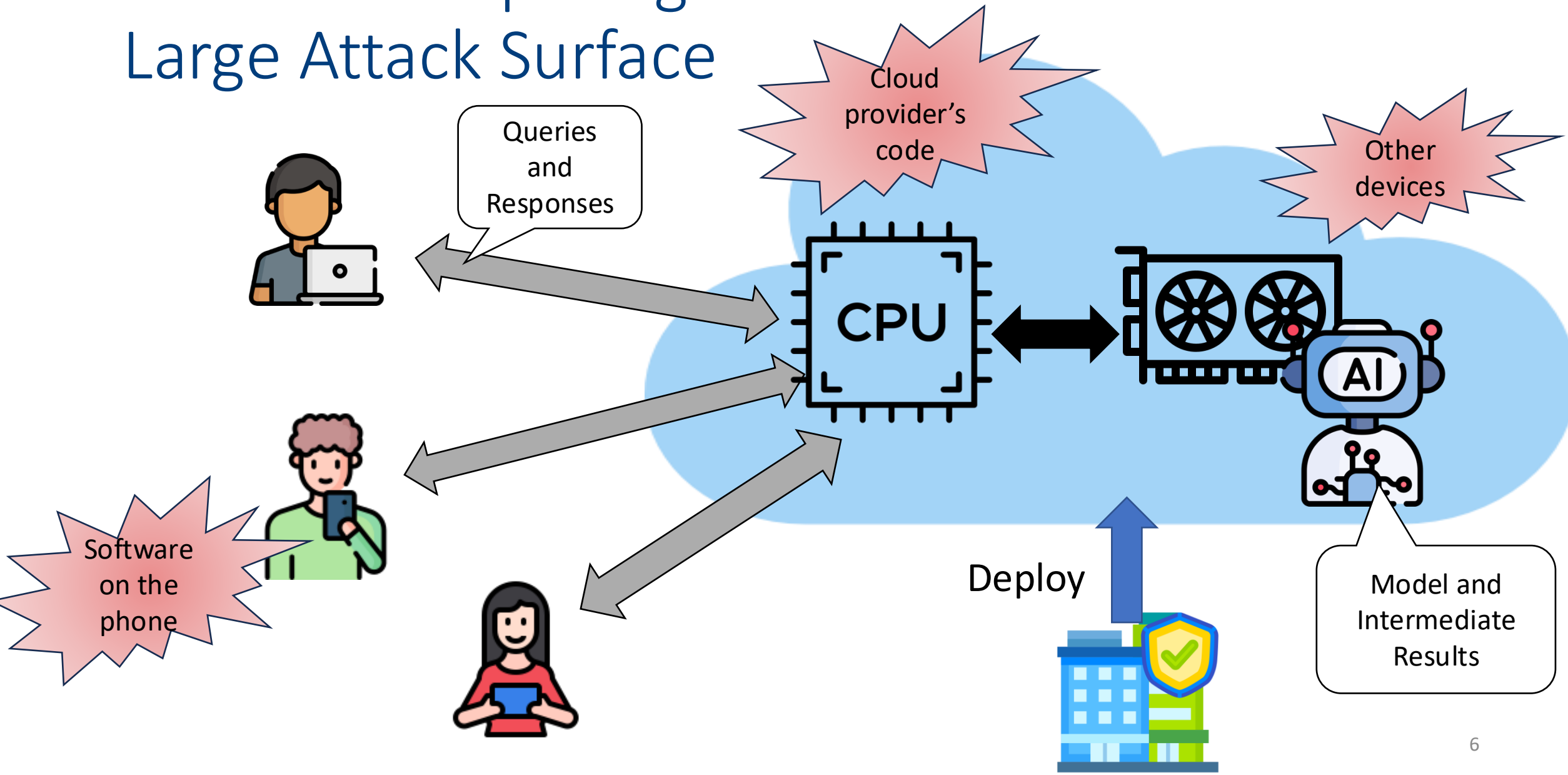


In Transit



In Use

# Current Computing Stack: Large Attack Surface



# Confidential Computing: Versatile Principle Applied to Real Systems




Sensitve Apps




Cloud Providers, Operating Systems




Servers, Mobiles, Sensor, GPUs, FPGAs, NICs

# *An incomplete history of the evolution from* Trusted to Confidential Computing




ARM TRUSTZONE










AMD

NVIDIA

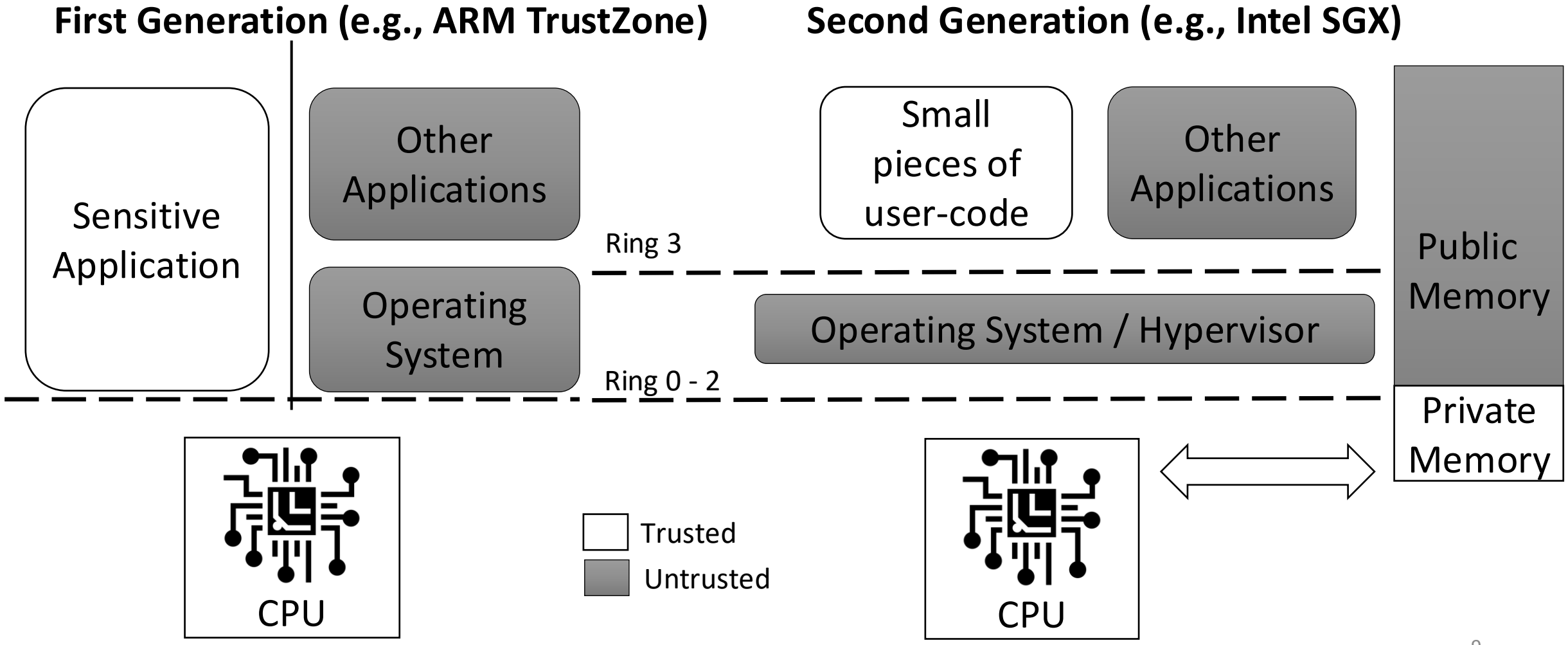
arm

intel



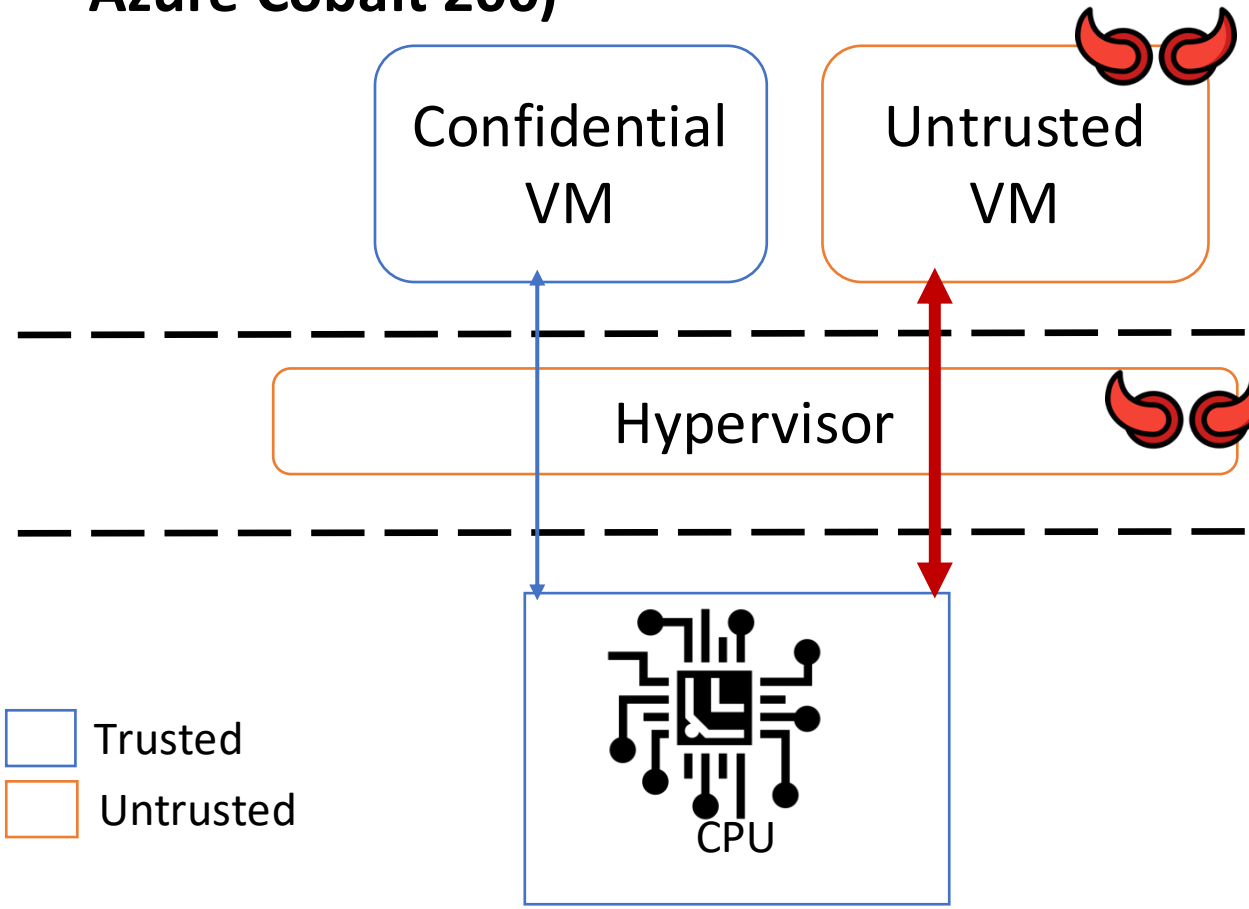
| 2004 | 2005 | 2006 | 2014 | 2015 | 2017 | 2022 | 2023 | |
|---|---|---|---|---|---|---|---|---|
| TPM 1.2 | ARM TrustZone | Intel TXT | TPM 2.0 | Intel SGX | AMD SEV | NVIDIA CC | Intel TDX | ARM CCA |

# Past Trusted Execution Environments

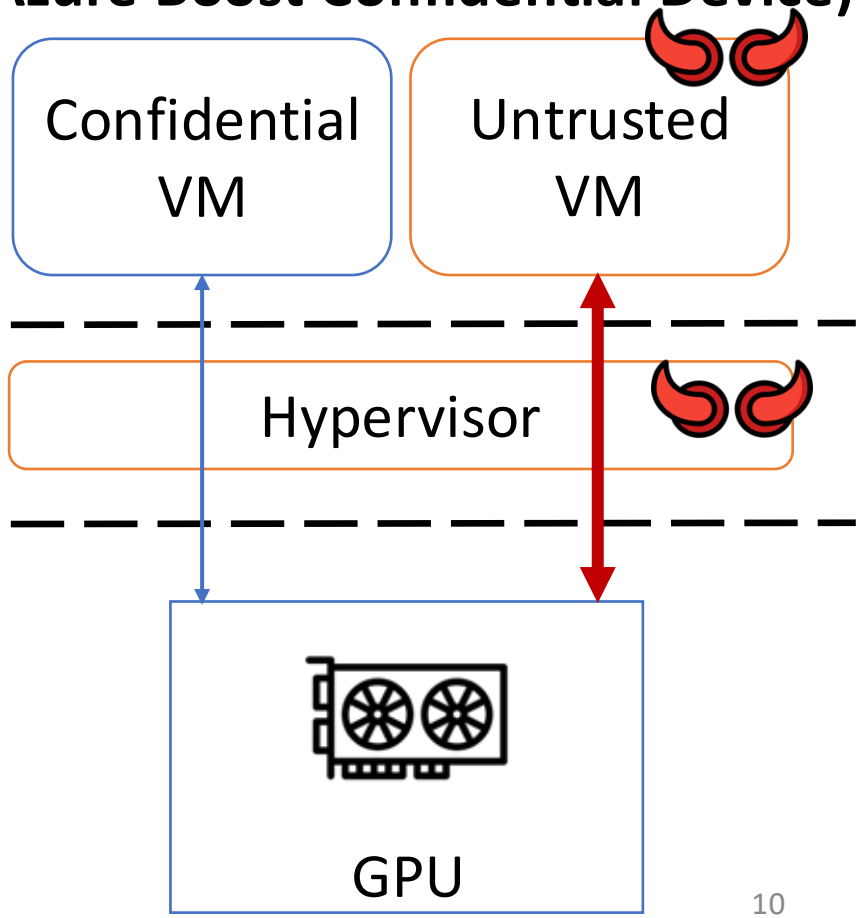


# Present Trusted Execution Environments

**New Generation CPUs**
**(e.g., AMD SEV-SNP, Intel TDX, Arm CCA, Azure Cobalt 200)**




**New Generation Accelerators**
**(e.g., NVIDIA H100 CC, Azure Boost Confidential Device)**




Trusted
Untrusted

# Confidential Computing has gone mainstream

## Confidential VMs - a security breakthrough for medical device software

April 1, 2022

**Shawn Dragann**
CEO, Idea Evolver

**Jerry Valentine**
Senior Director, Business Development New Technologies at AstraZeneca

## Bullish Embraces Confidential Computing

Bullish deploys Google Cloud instances powered by AMD to deliver the confidence needed for transactional integrity for its blockchain-based platform.

## Google launches synched passkeys as tech giants move away from passwords

Sep 20, 2024, 6:01 pm EDT | Abhishek Jadhav

Google's passkey automatic synchronization operates under AMD's Secure Encrypted Virtualization — Secure Nested Paging (SEV-SNP) to provide security for virtual machines. This feature utilizes the Oak project framework.

Signal



**Technology preview: Private contact discovery for Signal**

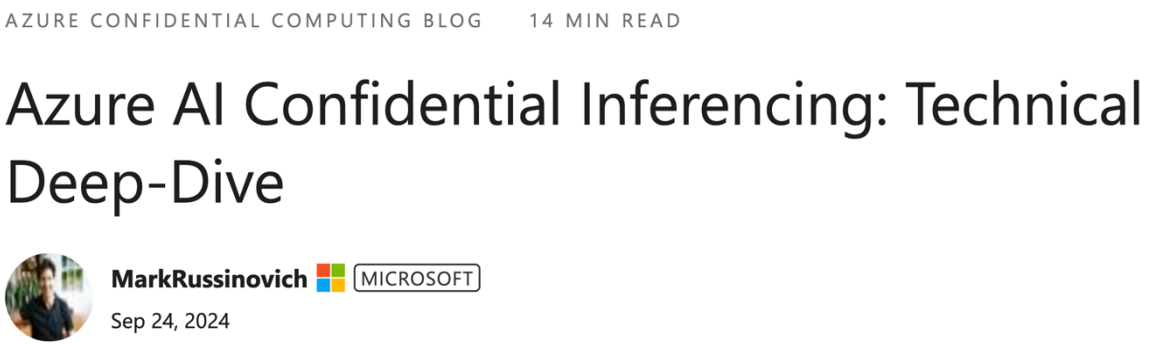
## MonetaGo Detects Duplicate Financing Fraud

MonetaGo uses Google Cloud Confidential Kubernetes Engine powered by AMD EPYC™ CPUs to detect Duplicate Financing fraud and keep data protected.

## Maximum Security at the Processor Level: Intel® SGX Protects Electronic Patient Record

# Confidential Computing: For AI Inference






# Confidential Computing: Phone to Cloud

POSTED ON APRIL 29, 2025 TO SECURITY & PRIVACY

## Building Private Processing for AI tools on WhatsApp




oppo

Newsroom All Press Release Stories Intellectual Property

## How OPPO and Google Are Redefining Mobile AI with Seamless Integration and Enhanced Security

Stories · Mar 04, 2025

Security Research Overview Blog Bounty Research Device Submit a Report

Blog

June 10, 2024

## Private Cloud Compute: A new frontier for AI privacy in the cloud

Written by Apple Security Engineering and Architecture (SEAR), User Privacy, Core Operating Systems (Core OS), Services Engineering (ASE), and Machine Learning and AI (AIML)

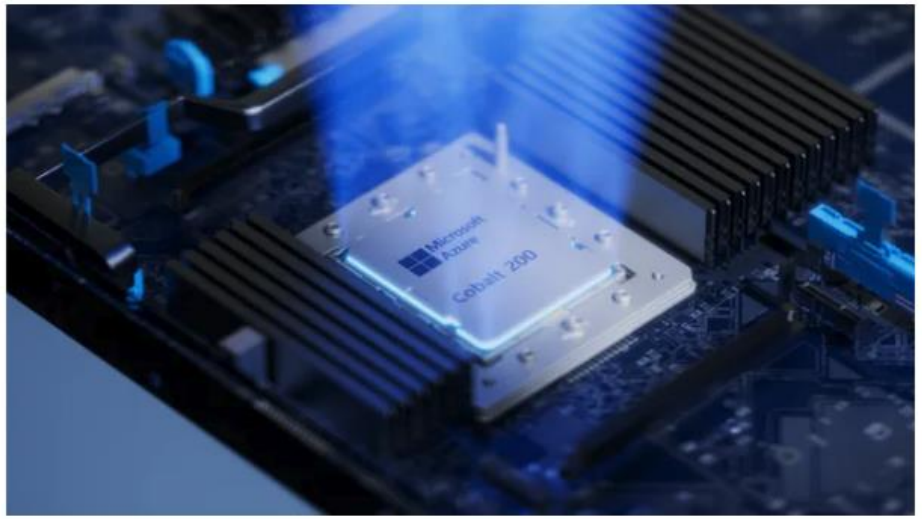
# Confidential Computing: Upcoming Support

Qualcomm Dragonfly C1000

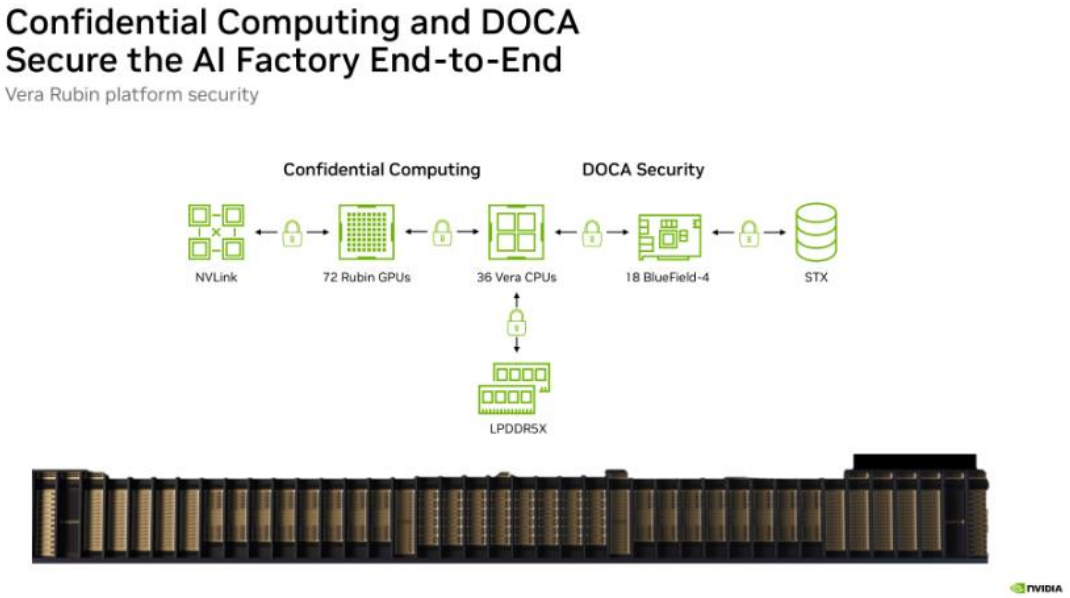
Features / Specs

- Confidential computing, server-class RAS, telemetry, and debug features

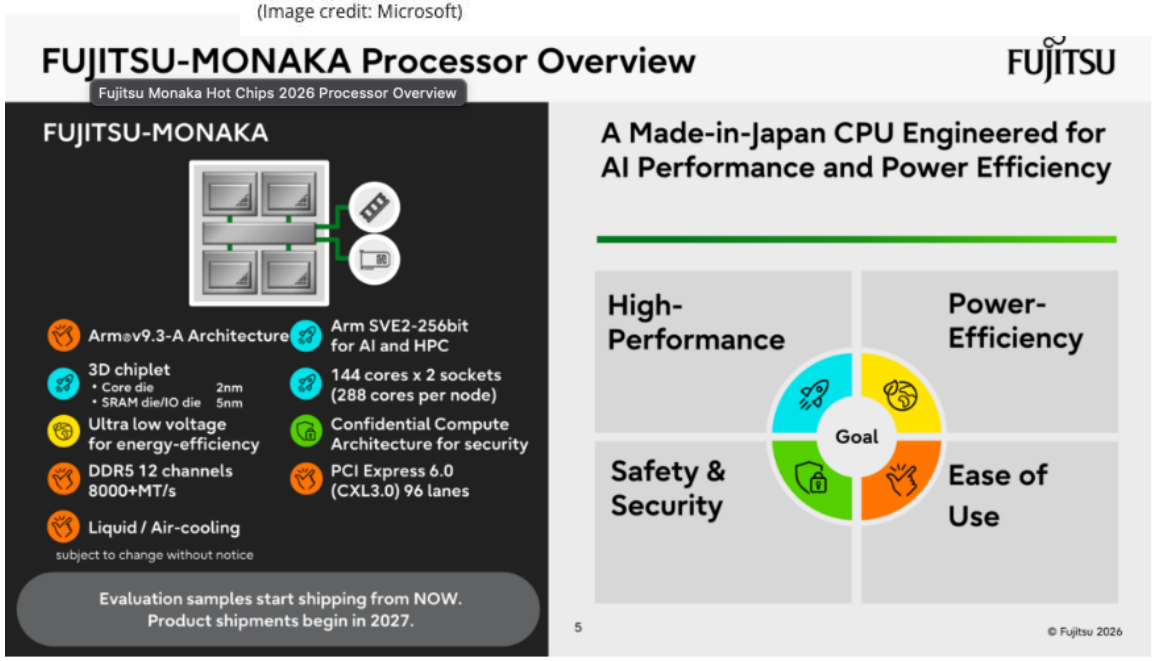
Microsoft Cobalt 200 uses Arm Confidential Compute Architecture for memory isolation




(Image credit: Microsoft)

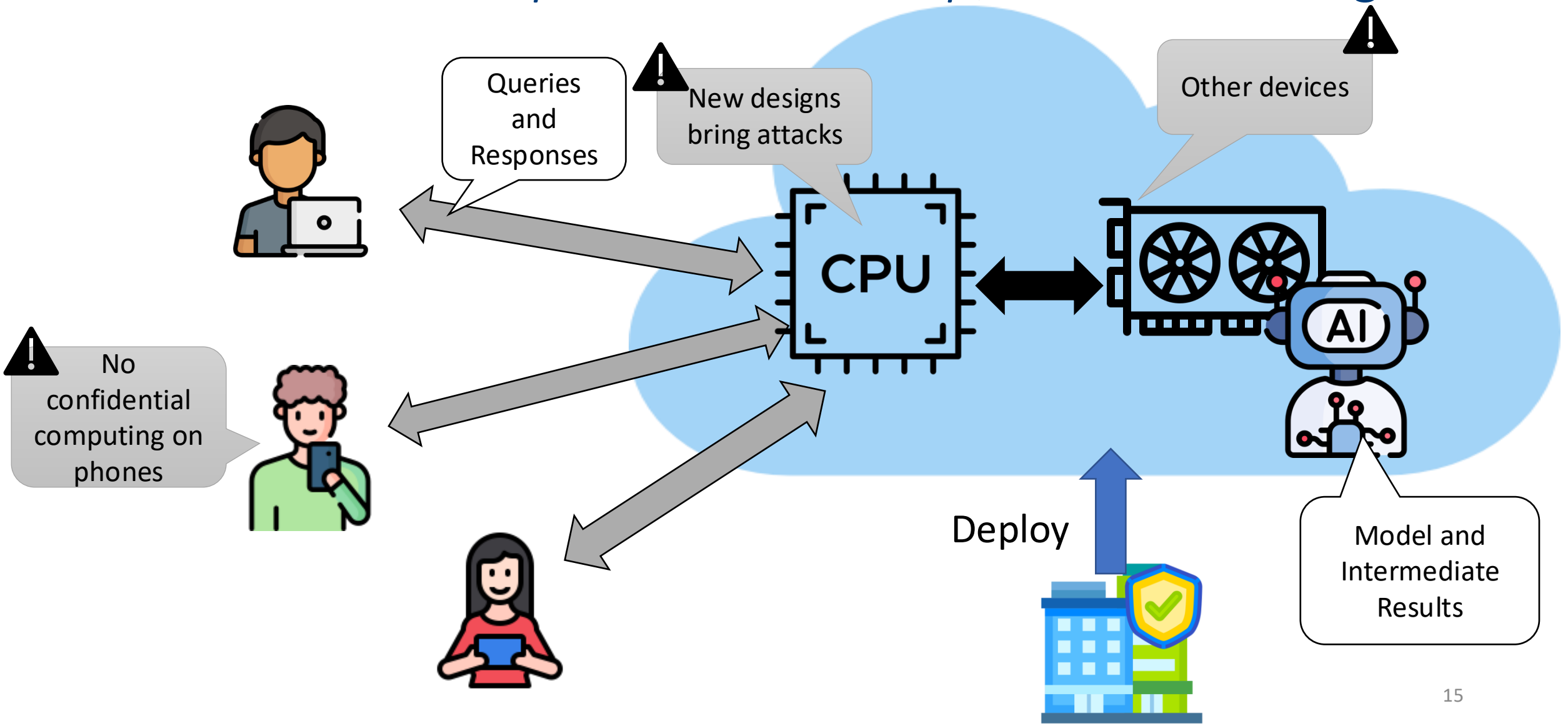



*NVIDIA Vera Hot Chips 2026 Confidential Computing*




*Fujitsu Monaka Hot Chips 2026 Processor Overview*

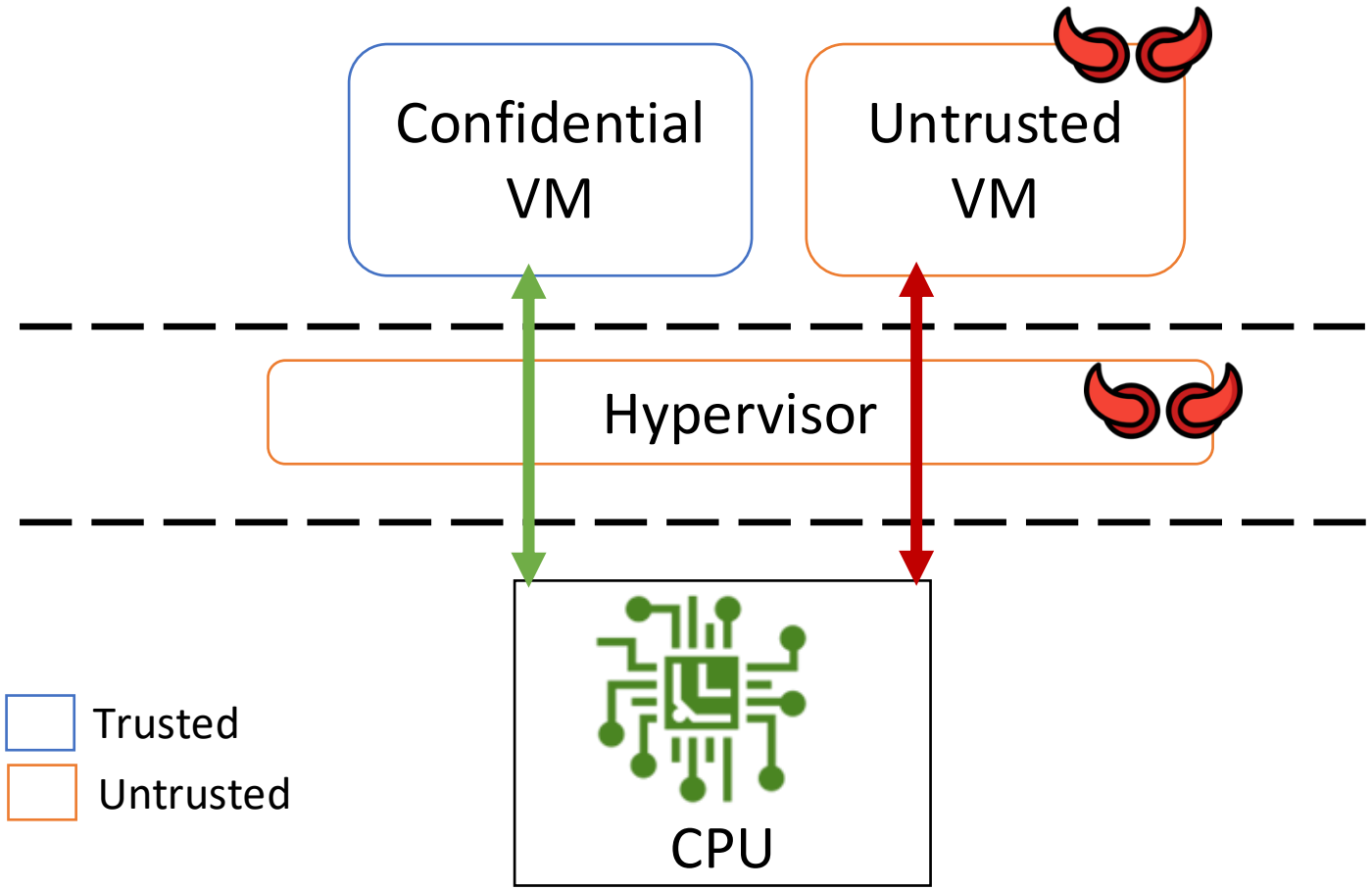
# Current State: *promises to keep, and miles to go*



# Interrupts as an example for attacks & defenses

# Confidential Virtual Machines

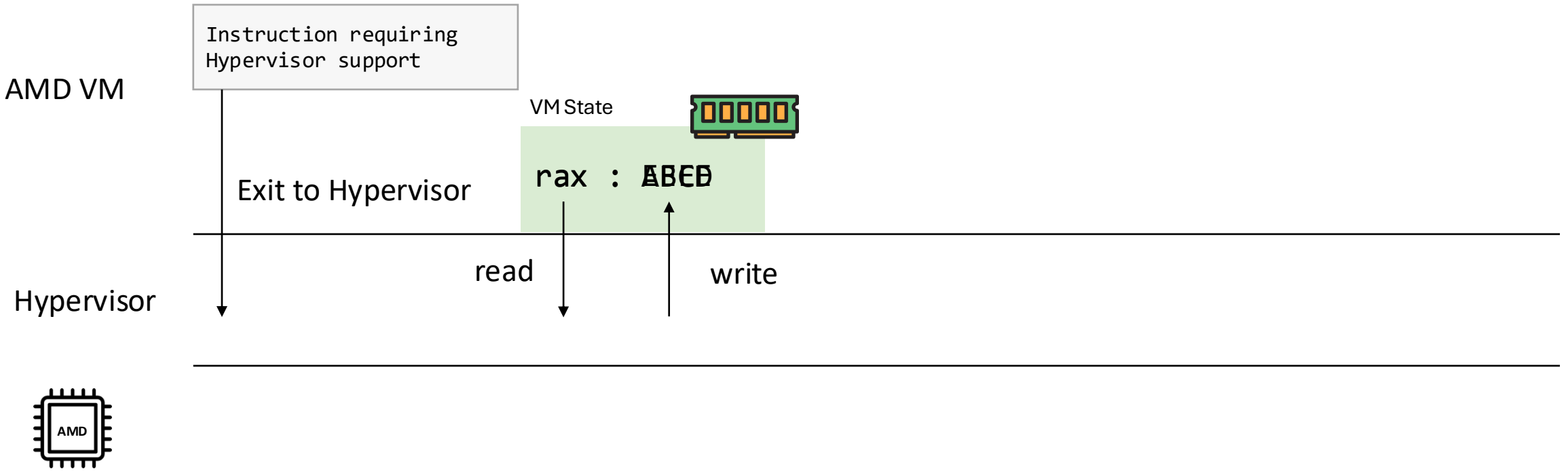
**New Generation CPUs**
**(e.g., AMD SEV-SNP, Intel TDX, Arm CCA)**



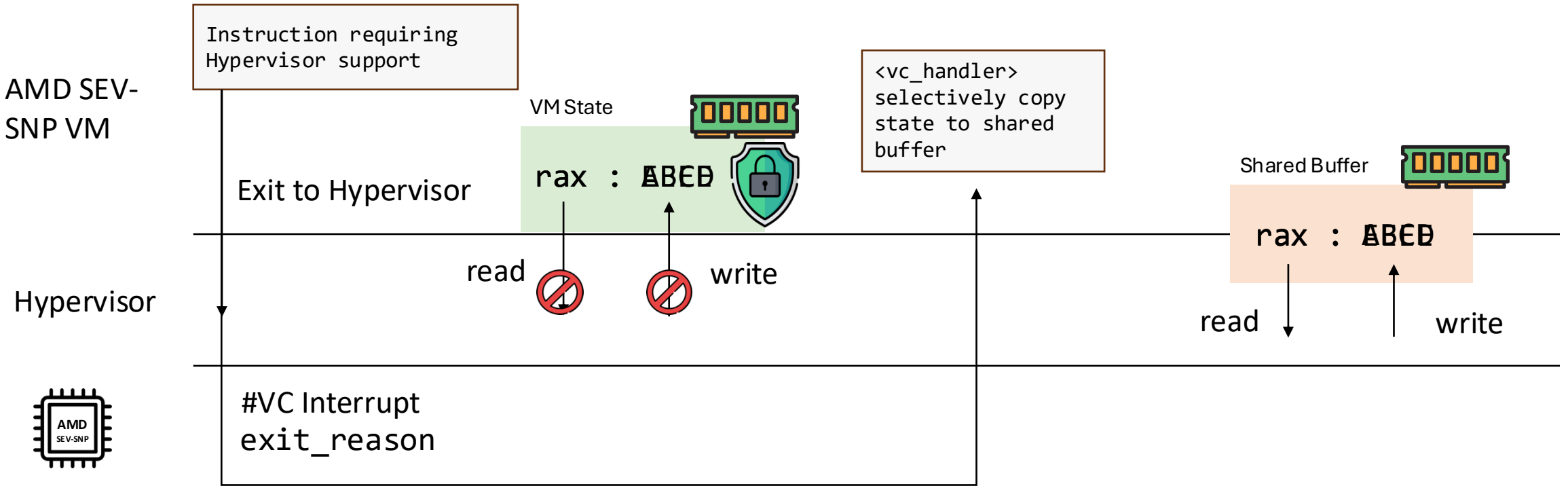



Cloud provider's hypervisor orchestrates, hardware enforces isolation

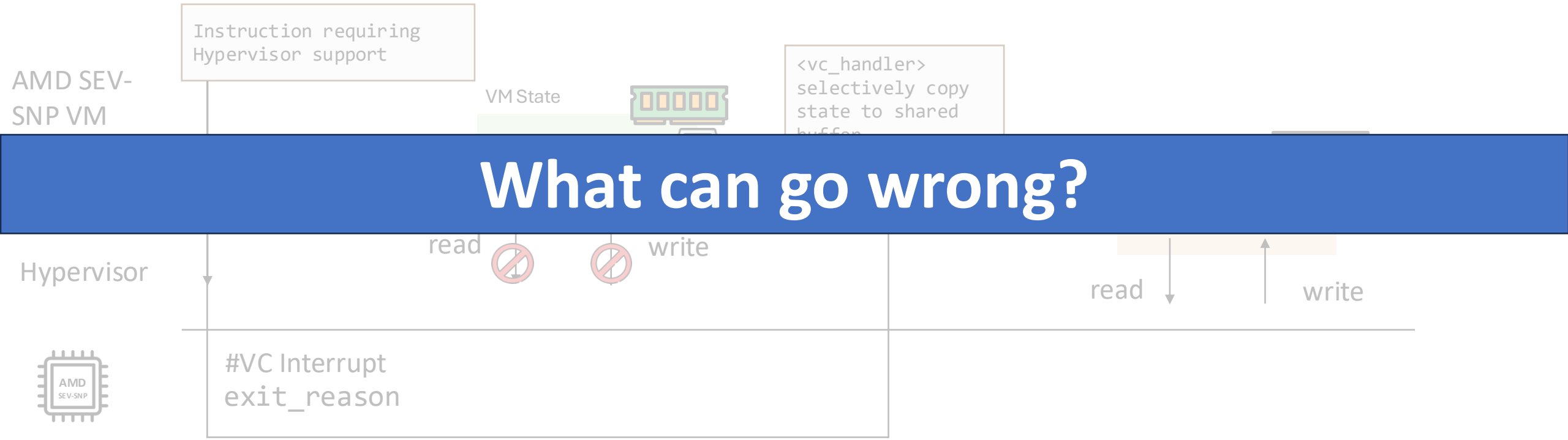
# AMD Virtualization: Guest – **Trusted** Hypervisor Communication



# AMD SEV-SNP: Guest – **Untrusted** Hypervisor Communication



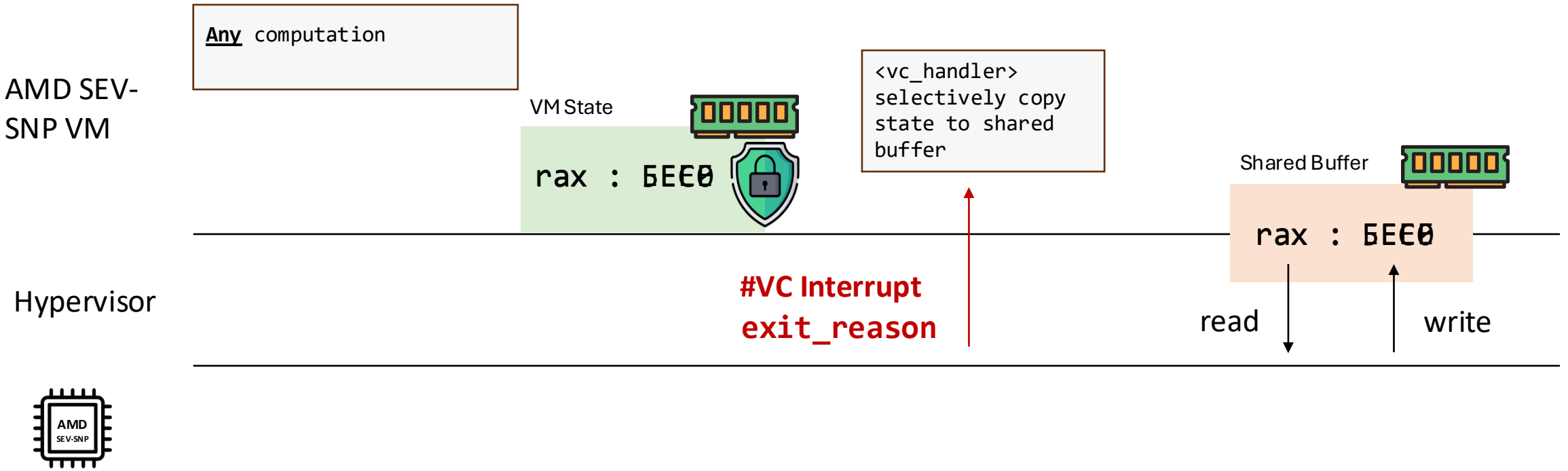
# AMD SEV-SNP: Guest – Untrusted Hypervisor Communication




**What can go wrong?**

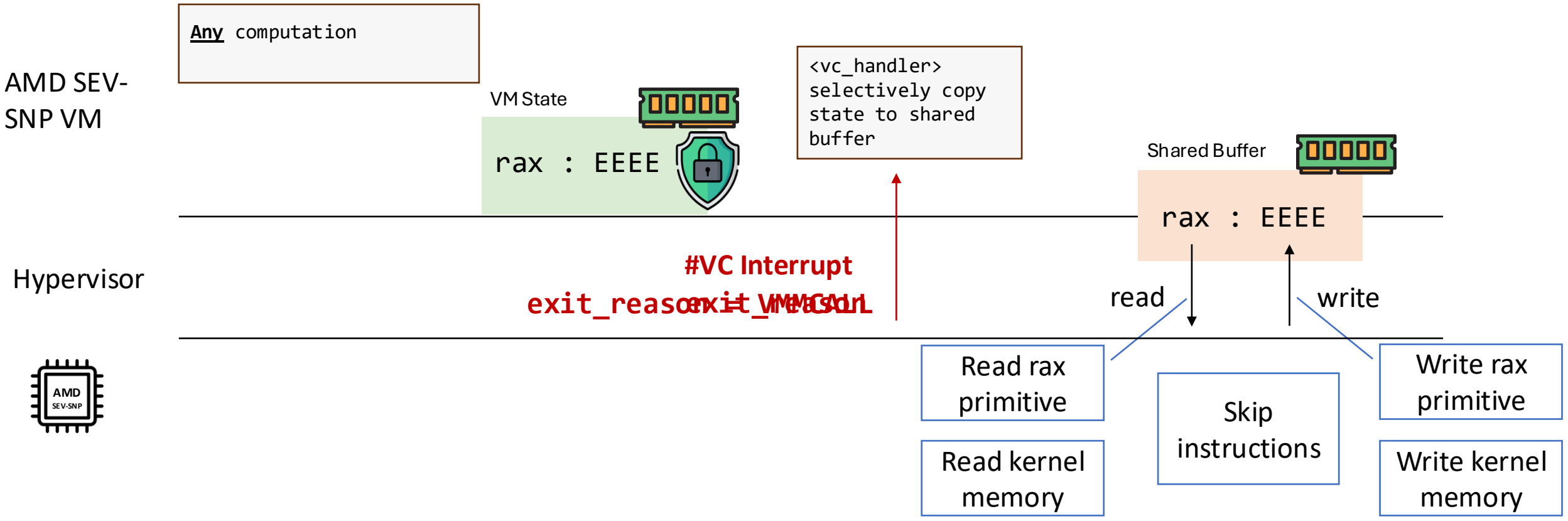
# WeSee Attack

Hypervisor can fake a #VC interrupt ***at any time***, the VM will start executing the handler ☹
Hypervisor can control the exit_reason ☹ ☹



# WeSee Attack

| Exit Reason | Description | Reg. → Hyp. | Hyp. → Reg. | Sample Inst. |
|---|---|---|---|---|
| VMMCALL | Call to Hypervisor | rax | rax | |
| MMIO WRITE | Memory Mapped I/O write | #reg | - | mov rbx,[rax] |
| MMIO READ | Memory Mapped I/O read | - | #reg | mov [rax],rbx |

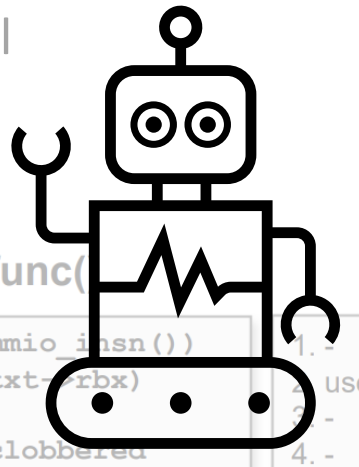


# WeSee Attack: Write kernel memory

1. **control** a **pointer**
2. **dereference** the **pointer** and write to kernel

1. #VC MMIO read
2. write rax
3. write rax
4. skip instructions

Total **#VC interrupts** = 34



**handler description**

1. decode instruction at context rip
2. get pointer to rbx of the application context
3. save rax in r14
4. clobber rax
5. restore rax from r14
6. copy value of application rbx to ghcp
7. get PTE for MMIO page [rdi] (cause of the VC)
8. check if page walk was successful
9. terminate VM in case of failure
10. check that rdi is marked as MMIO page
11. copy physical address of page to ghcb
12. exit the VM and let the hypervisor do the MMIO

(a)

**mmio_write_func(**

```
1.  if (check_decode_mmio_insn())
2.    rax = get_ptr(ctxt->rbx)
3.    r14 = rax
4.    ... //rax gets clobbered
5.    rax = r14
6.    memcpy(rax, ghcb->buf)
7.    rax = get_pte(ctxt->rdi)
8.    if (!rax)
9.      terminate_VM()
10.   if (rax->PAGE_ENC != 1)
11.     ghcb->gpa = rax->gpa
12.     perform_VMEXIT(ghcb)
```

(b)

**attack steps**

1. -
2. use $W_{rax}$ to set rax to $VA_{secret}$
3. -
4. -
5. -
6. -
7. use $W_{rax}$ S to set rax to $VA_{valid}$
8. skipped
9. use S to skip
10. check succeeds due to step 7
11. -
12. -

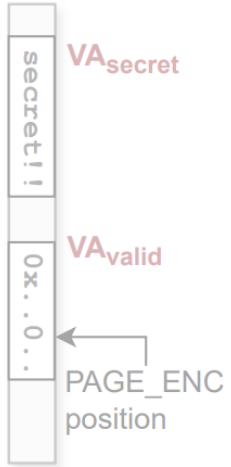
(c)

**attack$_{dataflow}$**

1. -
2. rax <- $VA_{secret}$
3. r14 <- $VA_{secret}$
4. -
5. rax <- $VA_{secret}$
6. buf <- *$VA_{secret}$
7. rax <- $VA_{valid}$
8. -
9. -
10. -
11. -
12. -

(d)

**RAM**




(e)

Figure 5. Read memory primitive. (a) Handler description for `#VC` caused by instruction `mov [rdi], rbx`. (b) Pseudo-code of MMIO write handling. (c) Attack steps. (d) Attack dataflow. (e) Memory used for attack.

# WeSee Case Studies



Leak KernelTLS keys

**288 #VCs** Session key stored in kernel memory



Disable `iptables` Firewall

**238 #VCs** `iptable` rules to filter network traffic in the kernel



Root Shell

**2891 #VCs** Code pages in the kernel

https://ahoi-attacks.github.io

# Disrupting TEEs with Malicious Notifications

Ahoi Attacks is a family of attacks on Hardware-based Trusted Execution Environments (TEEs) to break AMD SEV-SNP, Intel TDX and Intel SGX.



## Heckler
**(USENIX Sec'24)**

Breaking Confidential VMs with Malicious Interrupts



## WeSee
**(IEEE S&P'24)**

Using Malicious #VC Interrupts to Break AMD SEV-SNP



## Sigy
**(ACM AsiaCCS'25)**

Breaking Intel SGX Enclaves with Malicious Exceptions & Signals

# Summary



Ahoi attacks: New class of attacks
Malicious Interrupt Injection to trigger handlers

- CVE 2024-25742
- CVE 2024-25743
- CVE 2024-25744
- CVE 2024-25371
- CVE 2024-29970
- CVE 2024-29971

## Mitigations

- (Temporary) patches to guest Linux

[https://ahoi-attacks.github.io](https://ahoi-attacks.github.io)



CLOUD SECURITY

**Confidential VMs Hacked via New Ahoi Attacks**

New Ahoi attacks Heckler and WeSee target AMD SEV-SNP and Intel TDX with malicious interrupts to hack confidential VMs.

By Eduard Kovacs | April 8, 2024 (9:16 AM ET)

**Ahoi Attacks Impacting VM Security: Threats and Industry Responses**

Brittany Day · 2 - 3 min read · Apr 25, 2024

COMPUTER AND INFORMATION TECHNOLOGY · RESEARCH

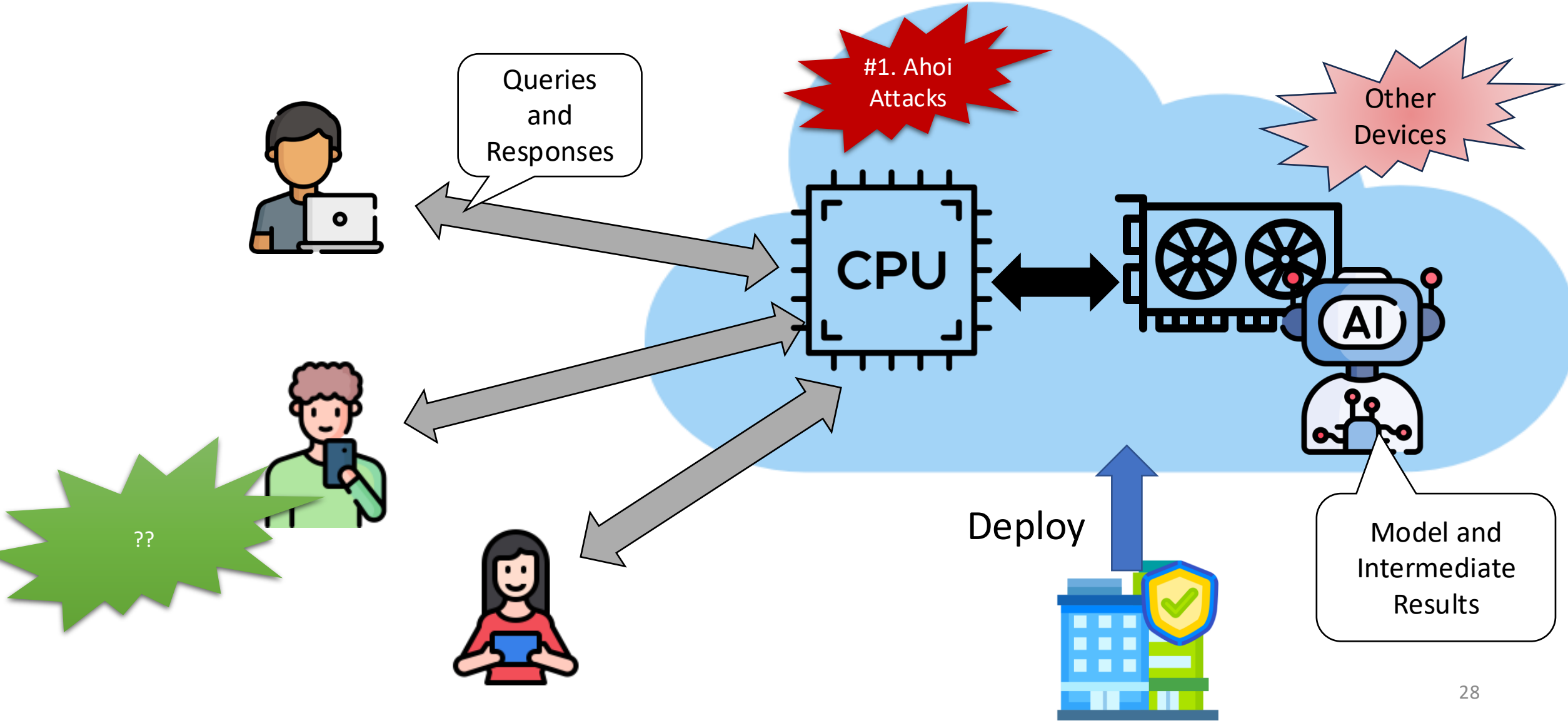
Knocking cloud security off its game

Public cloud services employ special security technologies. Computer scientists at ETH Zurich have now discovered a gap in the latest security mechanisms used by AMD and Intel chips. This affects major cloud providers.

04.04.2024 by Daniel Meierhans
(Updated: 07.04.2024)

Read · Share

# Have we made progress?



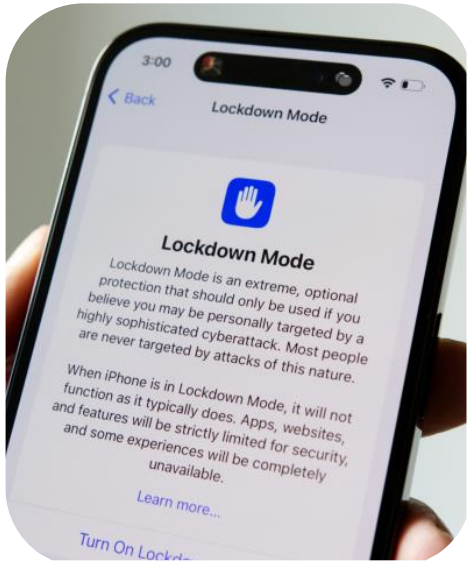
# Sovereign Smartphone

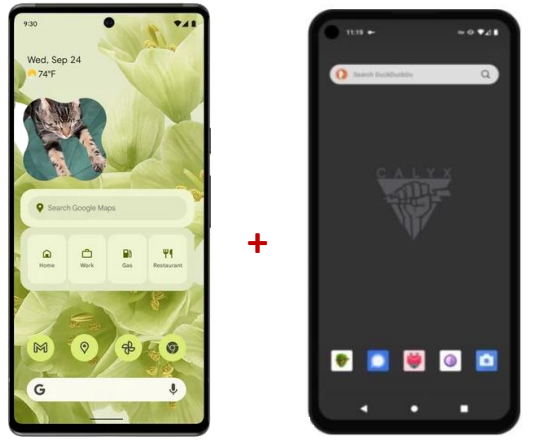
# Smartphones are critical, yet difficult to protect



**> 45 %**

of organizations compromised from employees' mobile device[1,2]




Personal Phone Work Phone

When functionality is limited, employees tend to undermine security efforts

**Smartphone are prime attack targets**

**Existing solutions trade-off functionality**

**Too strict policies lead to less security**

1. Verizon Mobile Security Index, 2024
2. Checkpoint Mobile Security Report, 2021

# We don’t control our phones



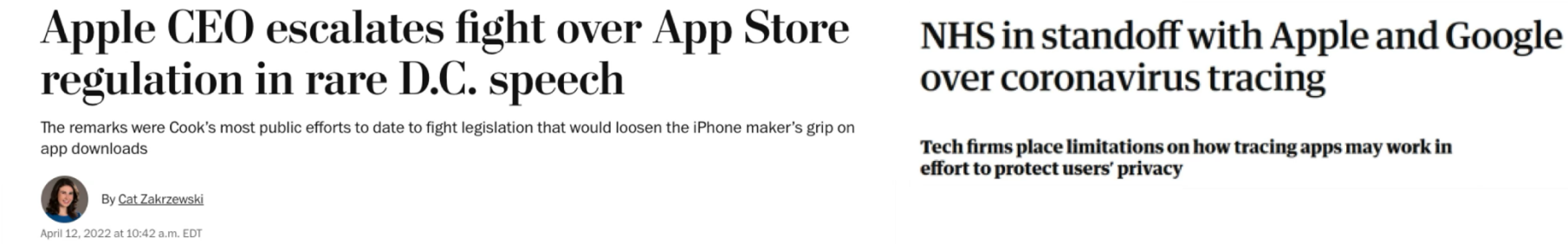
**Apple's Tim Cook Says the Fight to Protect Privacy Is a Crucial One**

Apple's CEO also pushes back against charges that its app store controls are anti-competitive.

Bree Fowler
April 12, 2022 7:29 a.m. PT
3 min read

https://www.cnet.com/news/privacy/tim-cook-says-the-fight-to-protect-privacy-is-a-crucial-one/

**Google tracks your movements, like it or not**



**Apple CEO escalates fight over App Store regulation in rare D.C. speech**

The remarks were Cook’s most public efforts to date to fight legislation that would loosen the iPhone maker’s grip on app downloads

By Cat Zakrzewski
April 12, 2022 at 10:42 a.m. EDT

https://www.washingtonpost.com/technology/2022/04/12/apple-ceo-tim-cook-app-store/

**NHS in standoff with Apple and Google over coronavirus tracing**

**Tech firms place limitations on how tracing apps may work in effort to protect users’ privacy**

# Ideally: Sovereignty in the existing ecosystem

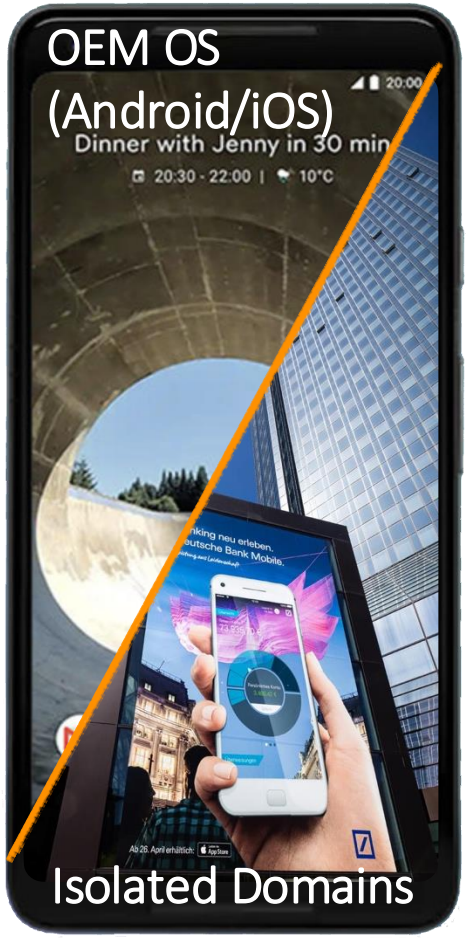


Credit: rawpixel.com

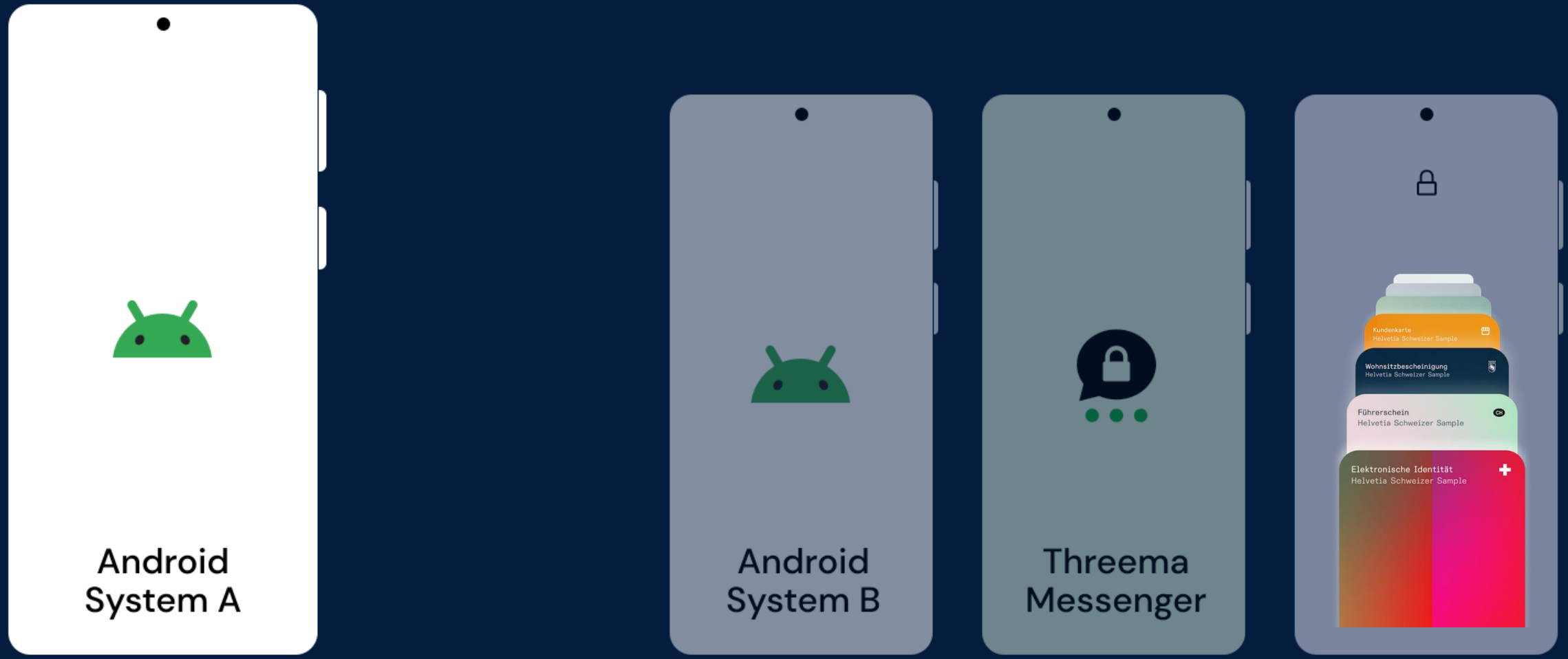
A new smartphone architecture bringing sovereignty from the OS without sacrificing functionality

# Sovereign Smartphone

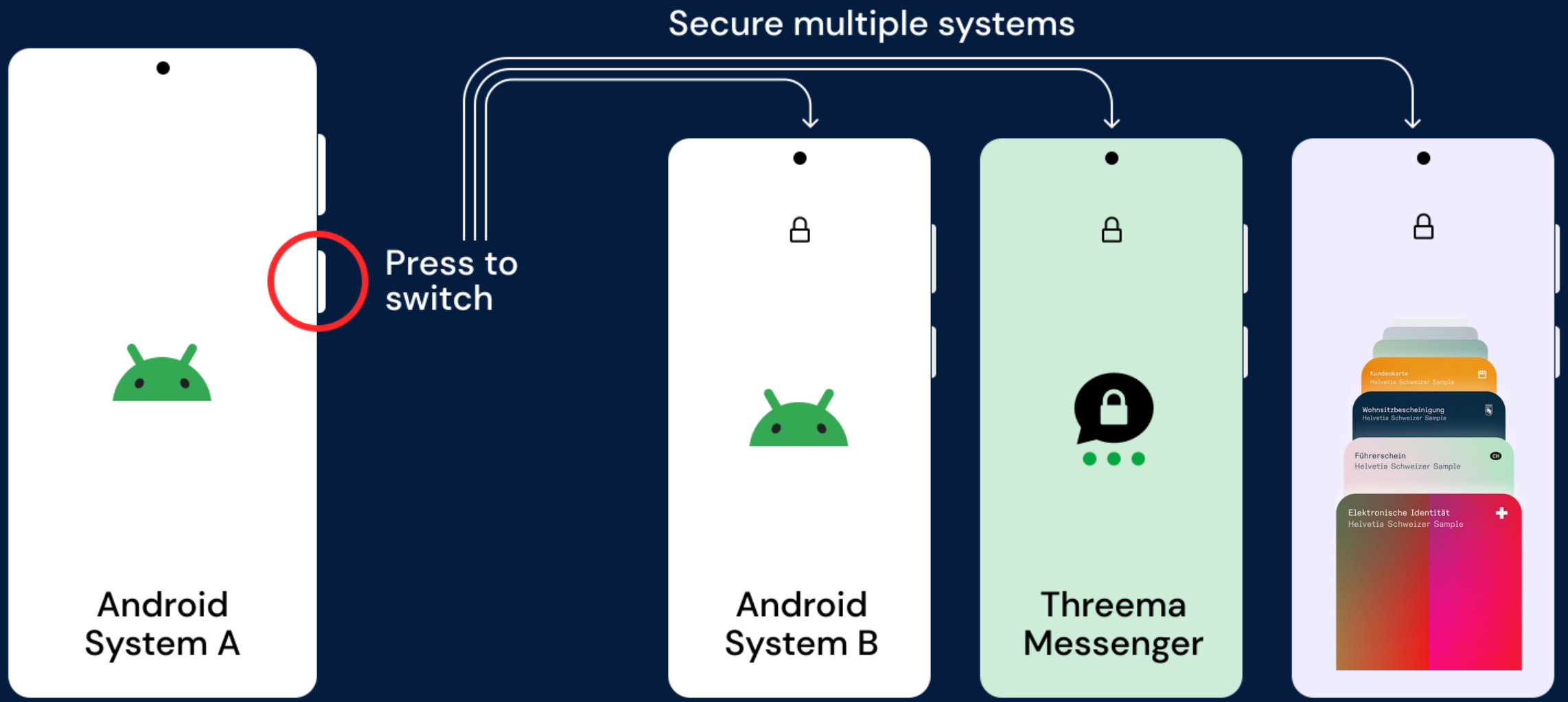
- Users can customize the full software
- Full control and transparency over ***all*** the software that has access to their data
- Users still have a “second phone” *on the same device* bare metal or with their favorite OS



# One device – multiple phones

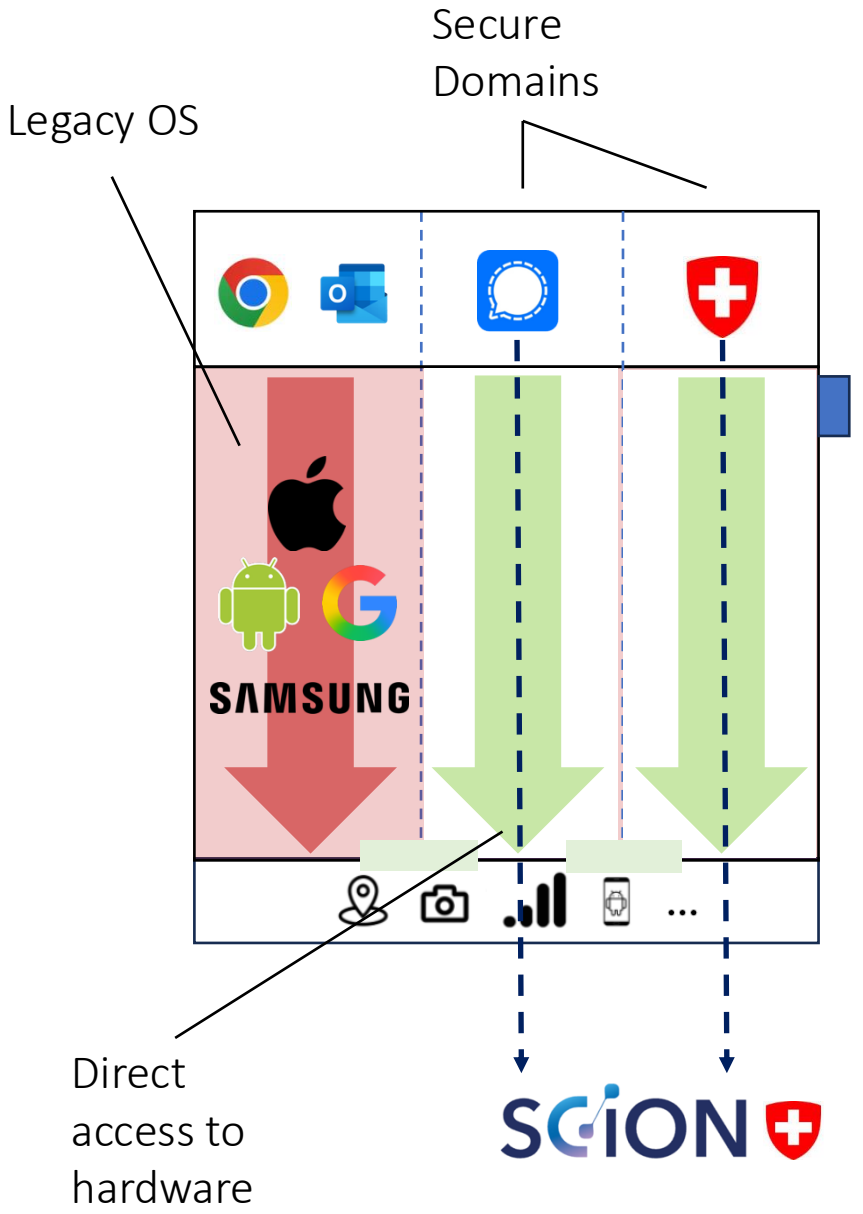


# One device – multiple phones

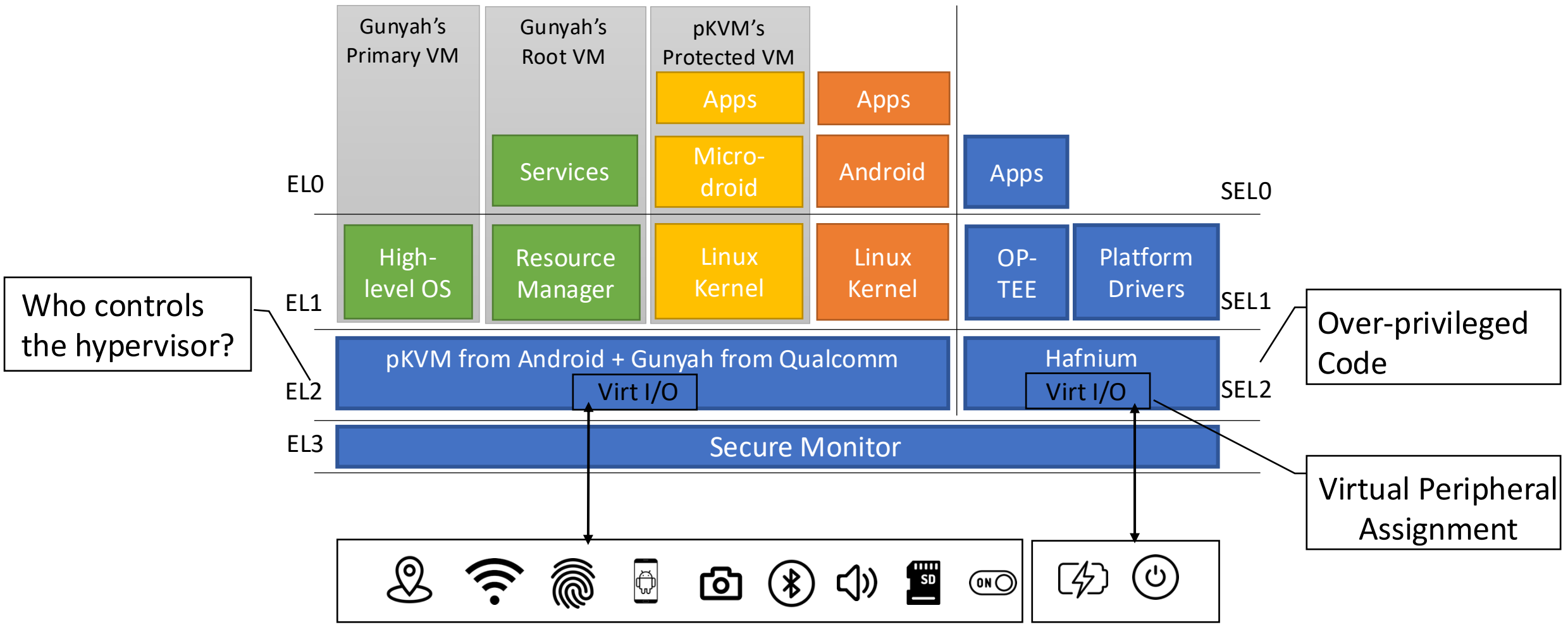


# Sovereign Smartphone

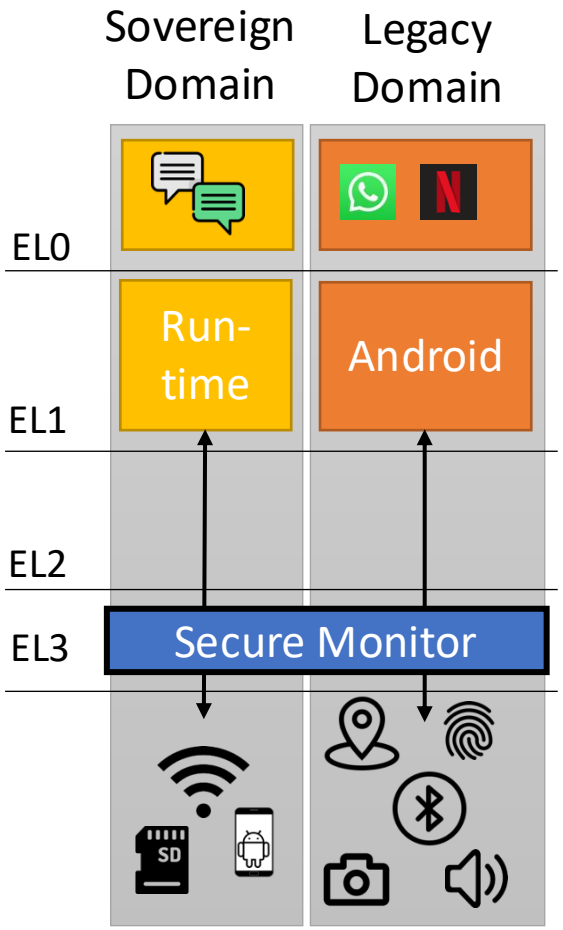
- High usability + rich app ecosystems
- Critical apps are isolated and have direct access to screen, network
- Critical app data stays safe even if the Legacy OS is malicious/untrusted



# Current Ecosystem is Unsuitable for Our Goals



# Our Proposal




- Simple and pragmatic for sovereignty
- Doesn’t disrupt existing OSes and apps

- Hypervisor is not in the TCB
- Only firmware modifications

- Peripherals directly assigned to domains
- Isolation enforced by existing hardware

# Example Use-case: Secure Chat App

- Deploy chat app in sovereign domain, Android in the legacy domain



Sovereign domain needs

- Execution time
- Isolated memory
- Access to touchscreen
- Access to network
- Access to storage

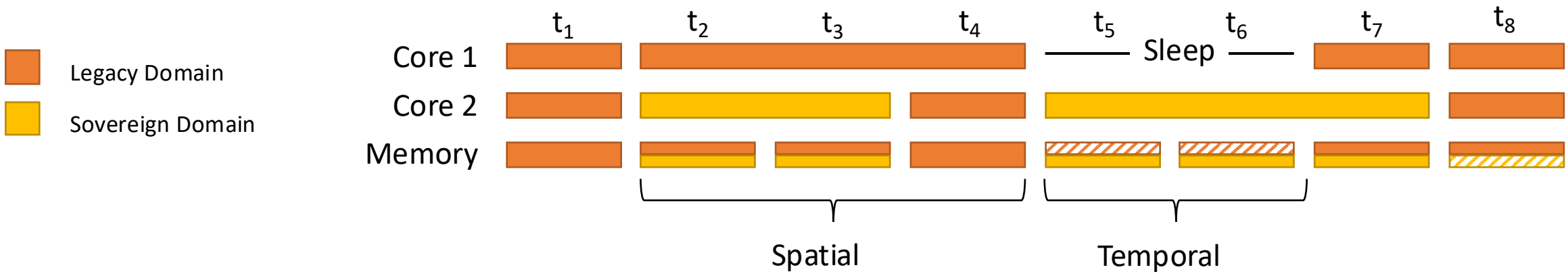
Legacy domain continues to

- Perform scheduling
- Manage memory
- Access other peripherals



# Challenge 1: Execution and Memory Isolation

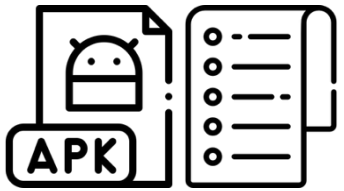
- Secure monitor configures isolation, hardware mechanisms enforce it
  Domains are isolated by executing
  - On different cores




- Memory accesses are isolated by
  - Allowing accesses based on core ID

# Challenge 2: Peripheral Assignment

App + Manifest




Sovereign domain needs:



Access to **touchscreen**

**Exclusive**: Always auto-assigned to the Sovereign domain when it executes, taken away from Legacy domain, handed-over at the end

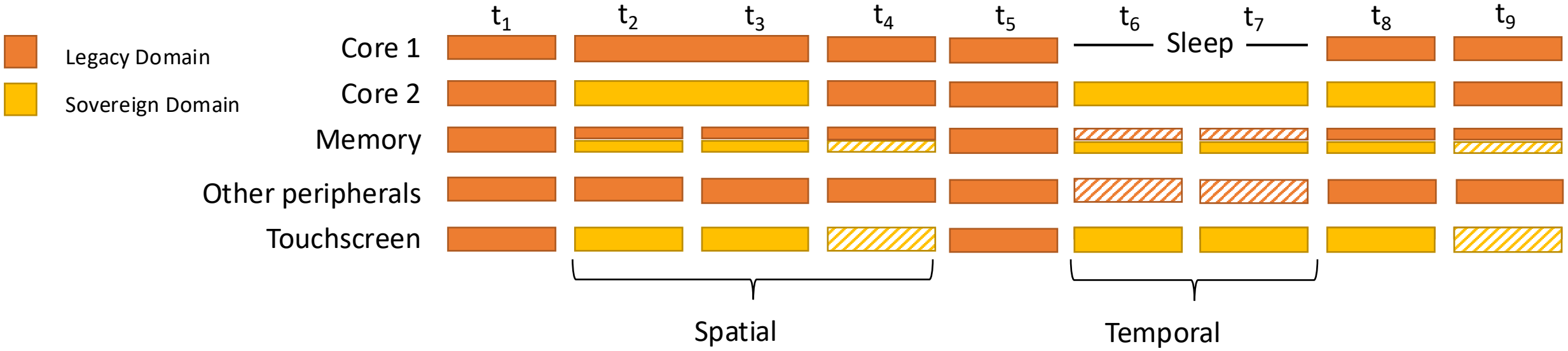
Access to **storage and network**

**Proxy:** Always owned by Legacy domain
Allows Sovereign domain to access via secure channel

# Challenge 3: Peripheral Isolation

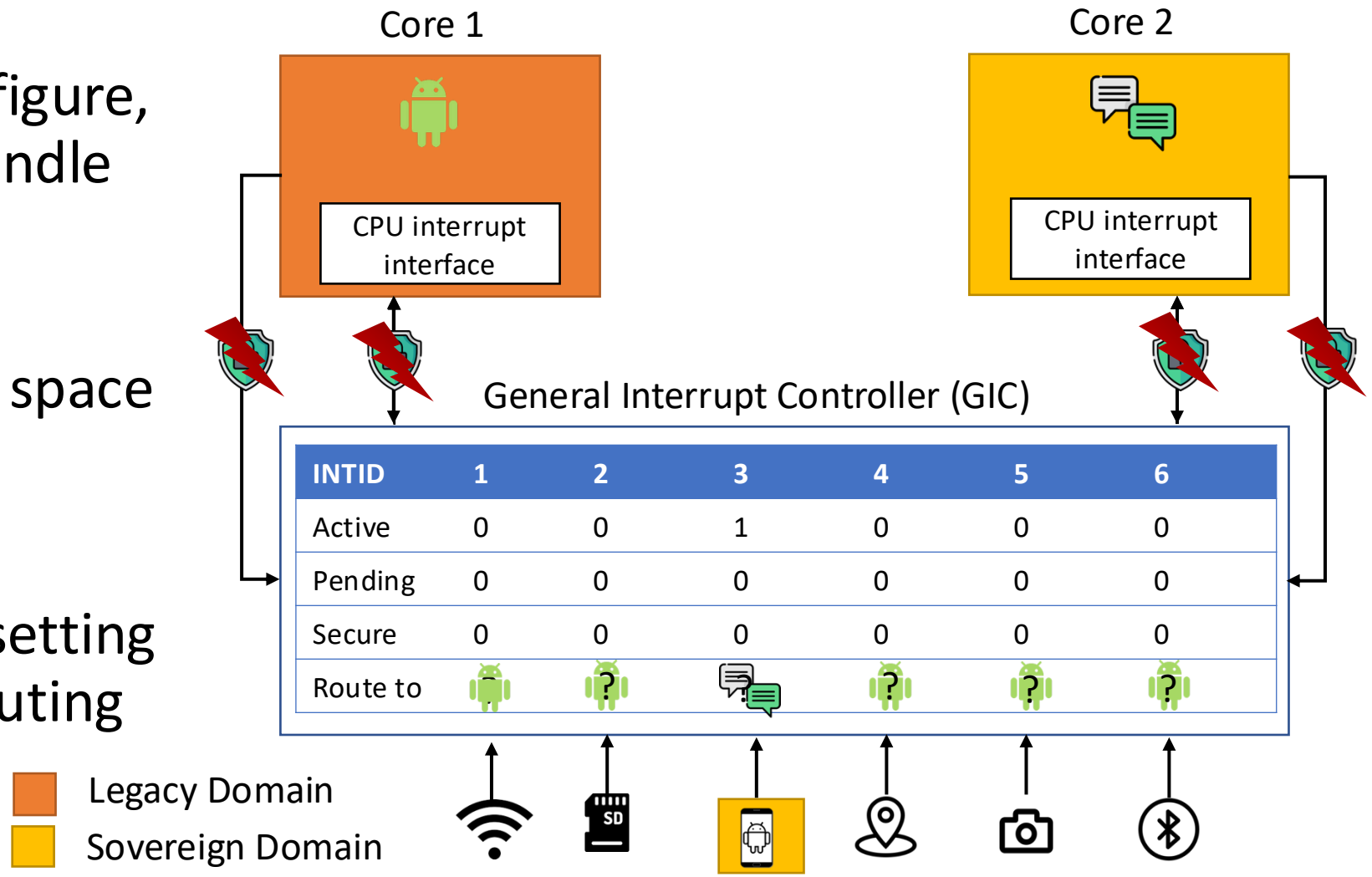
**Peripheral Access**: Read from and write to the memory-mapped peripheral address regions

Apply isolation at address-space level



# Challenge 4: Interrupt Management

- Domain needs to configure, route, receive, and handle peripheral interrupts
- Cannot apply address space based isolation here
- Isolate configuration setting and enable correct routing



# Supported Peripheral Access Modes

- Supports different modes of peripheral sharing:
  Exclusive, Handover, Multiplexing, Read Only, Proxy

- Secure Chat app example:

| CPU + Memory | Screen | Ethernet, Storage |
|---|---|---|
| Spatial | Handover | Proxy |

# The making of Sovereign Smartphone



# Winter 2020 - Spring 2021

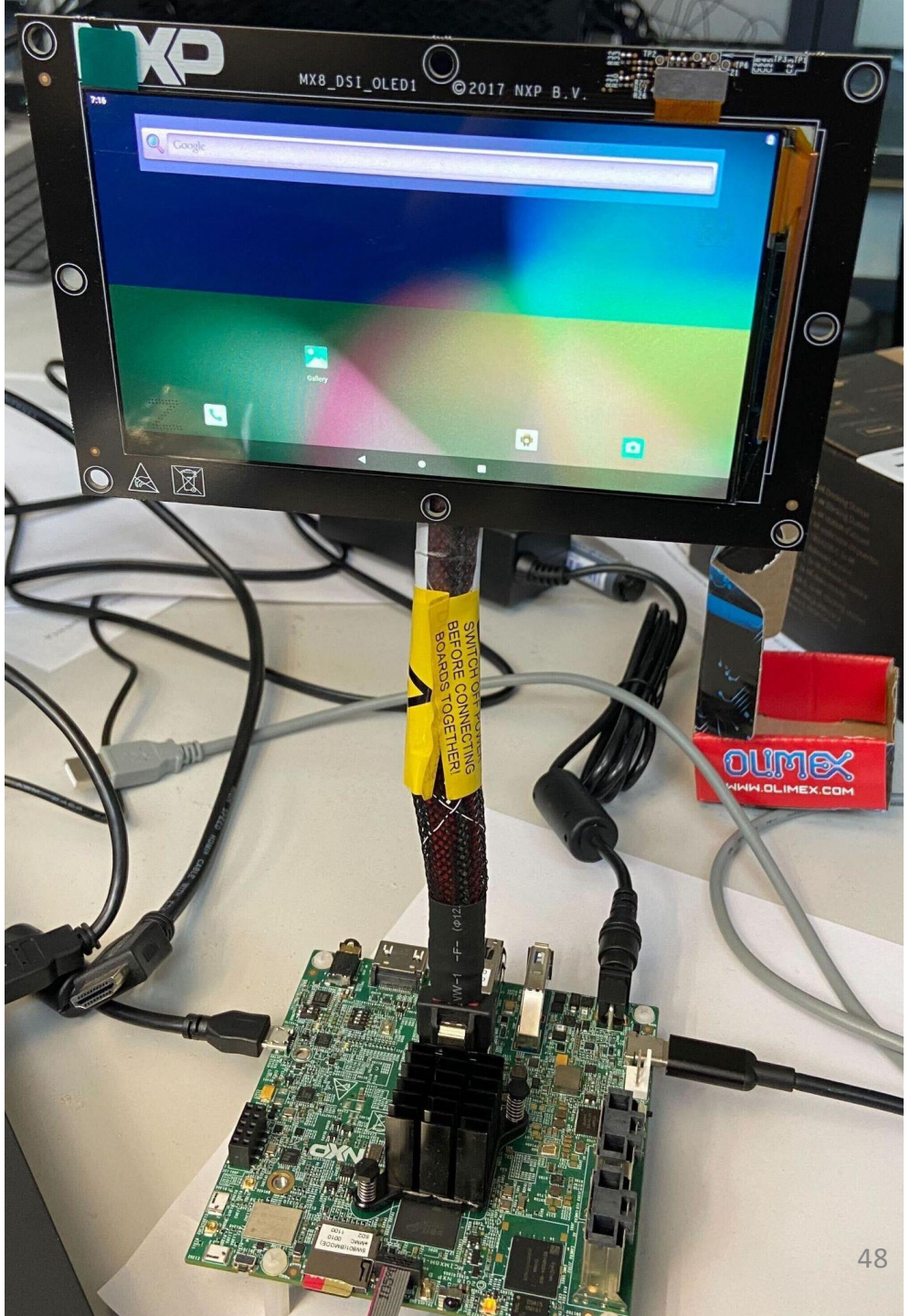
**Sovereign Smartphone:**
**To Enjoy Freedom We Have to Control Our Phones**

Friederike Groschupp Moritz Schneider Ivan Puddu Shweta Shinde Srdjan Capkun
*ETH Zurich*

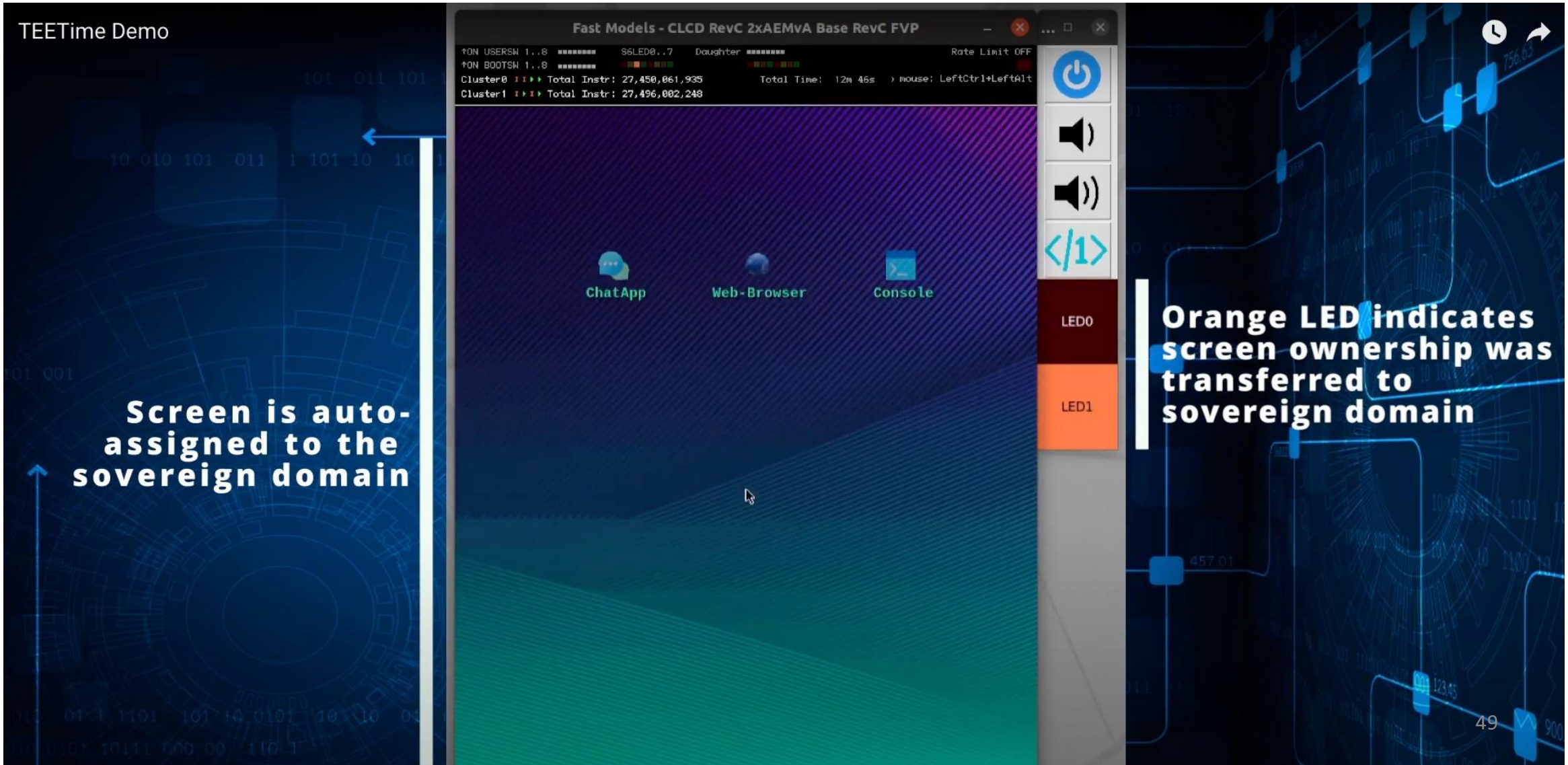
https://arxiv.org/abs/2102.02743

# Summer 2022

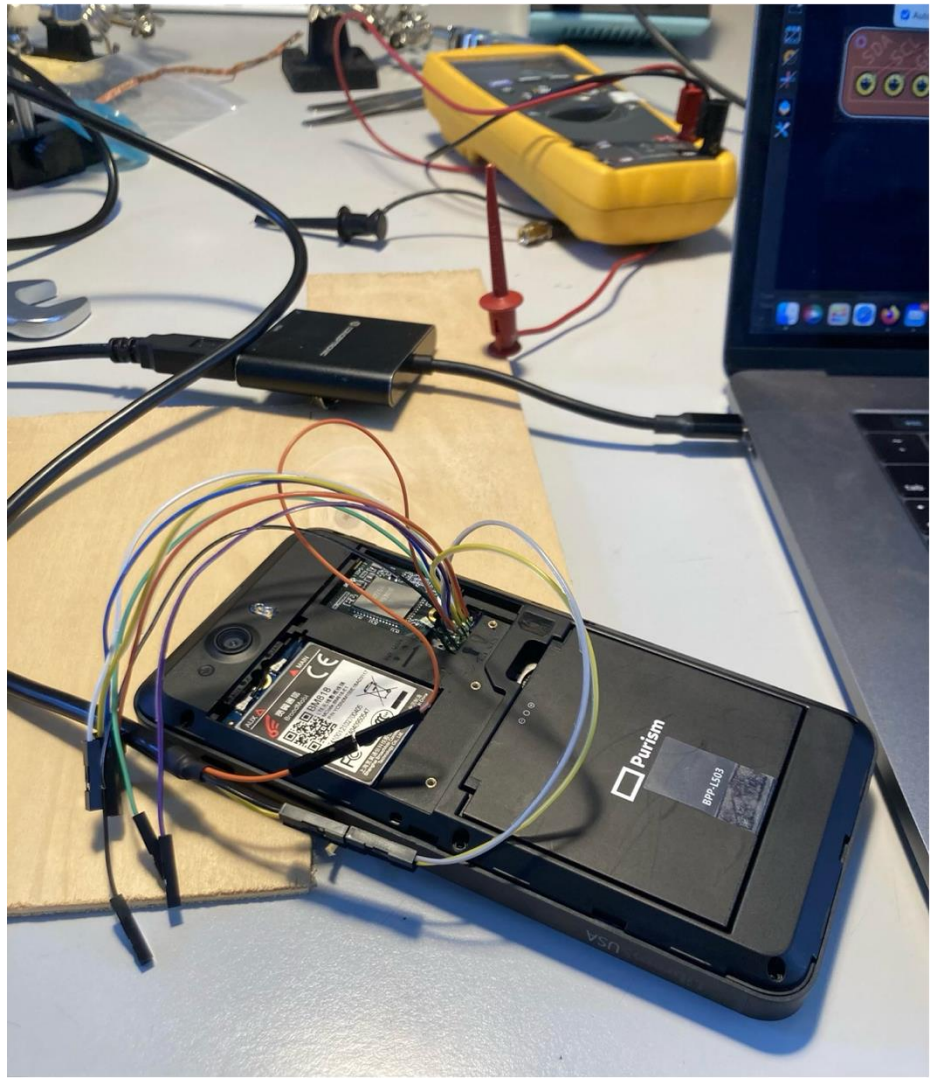
```
HELLo
HELLo
HELLo
HELLo
HELLo
HELLo
HELLo
HELLo
HELLo
HNOTICE:  S -FIQ-> L
NOTICE:  Switching to NS timer
INFO:    ID_S: 1d
INFO:    Interrupt count: 19
INFO:    DUMPING CURRENT TZASC CONFIG:
INFO:    enter
INFO:    security_inversion_en 0
INFO:
INFO:    region 1
INFO:    region_base: 0x0000000020000000
INFO:    region sp: c
INFO:    region size: 1b
INFO:    exit
INFO:    ----------------------------
```

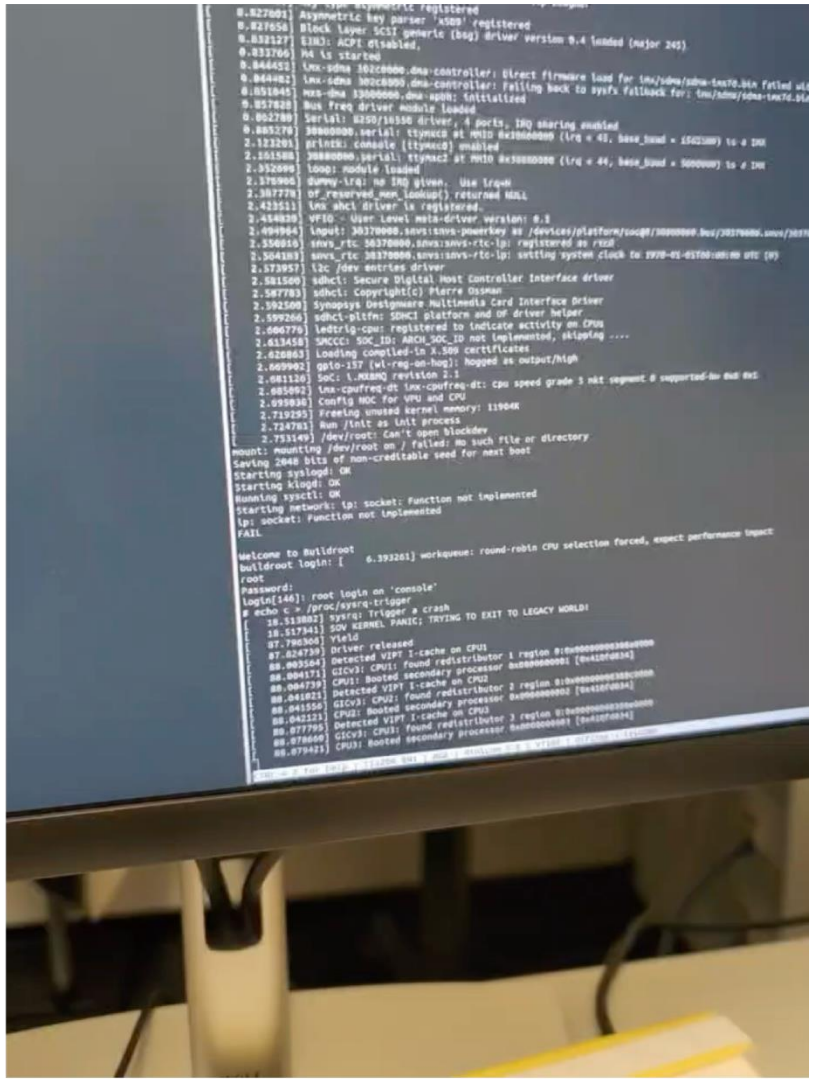


# Summer 2023

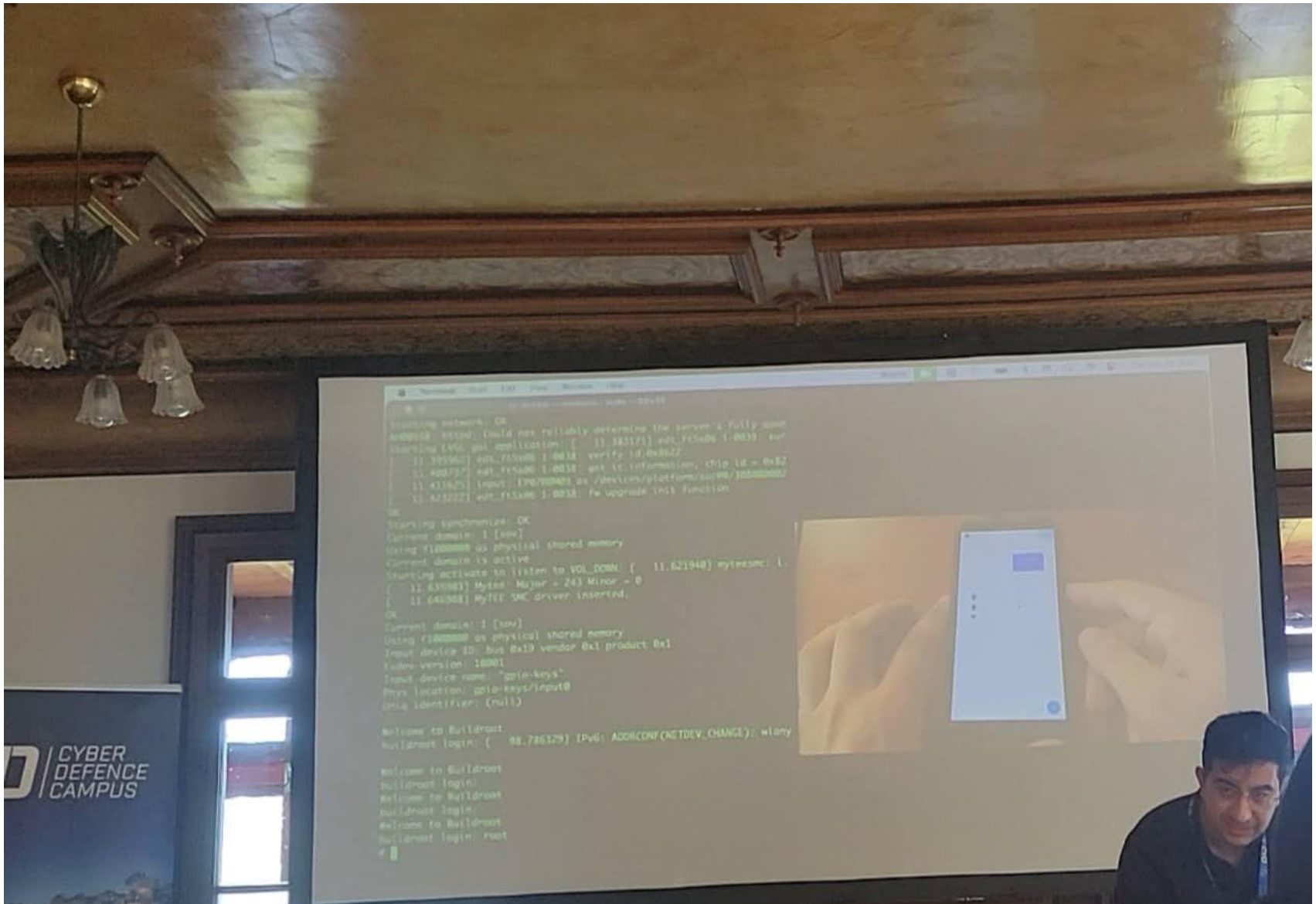

TEETime Demo
Screen is auto-assigned to the sovereign domain
Fast Models - CLCD RevC 2xAEMvA Base RevC FVP
Cluster0 Total Instr: 27,450,061,935
Cluster1 Total Instr: 27,496,002,248
Total Time: 12m 46s
mouse: LeftCtrl+LeftAlt
ChatApp
Web-Browser
Console
LED0
LED1
Orange LED indicates screen ownership was transferred to sovereign domain

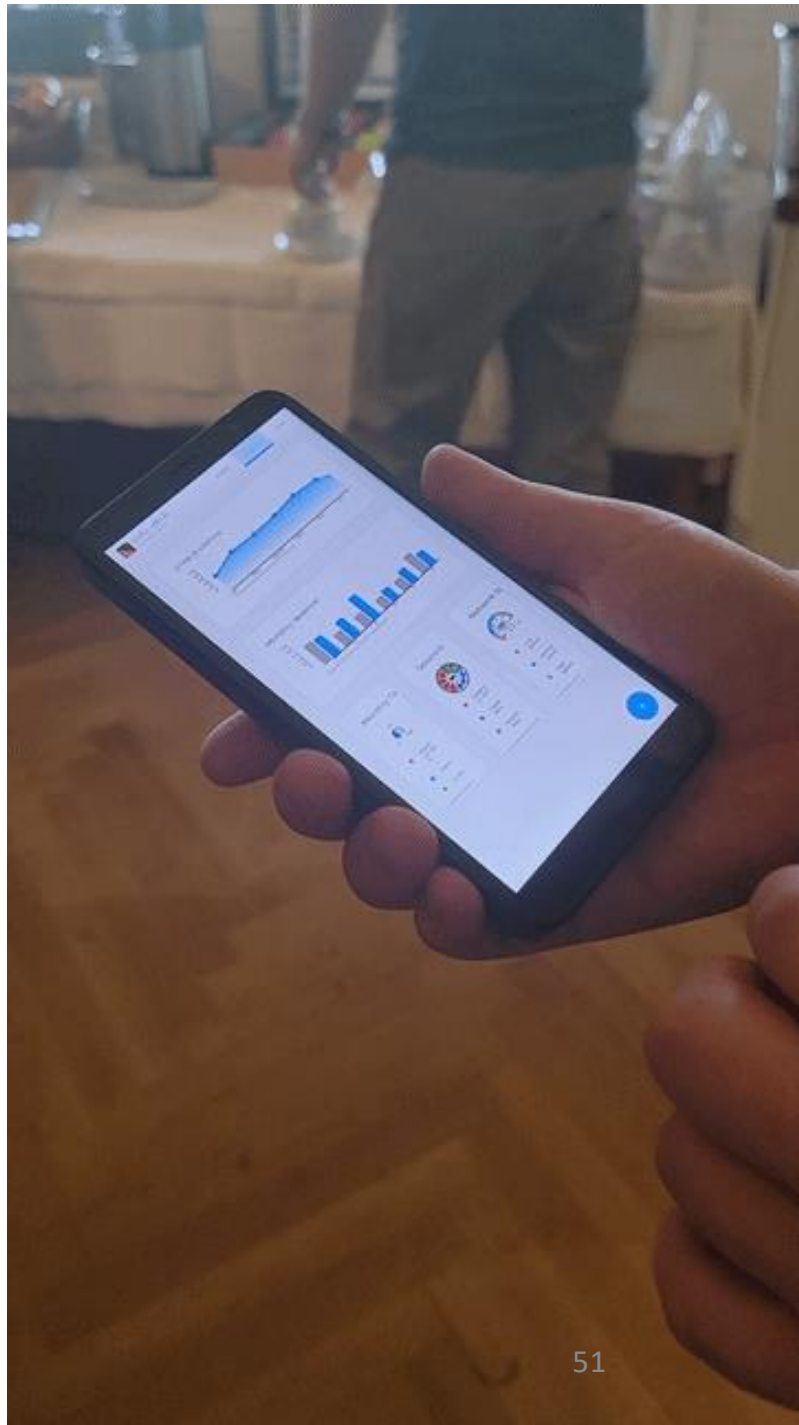
# Spring 2024


Purism
BPP-1403
USA


Welcome to Buildroot
buildroot login:
root
Password:
login[146]: root login on 'console'
# echo c > /proc/sysrq-trigger
mount: mounting /dev/root on / failed: No such file or directory
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting network: ip: socket: Function not implemented
ip: socket: Function not implemented
FAIL

# Summer 2024



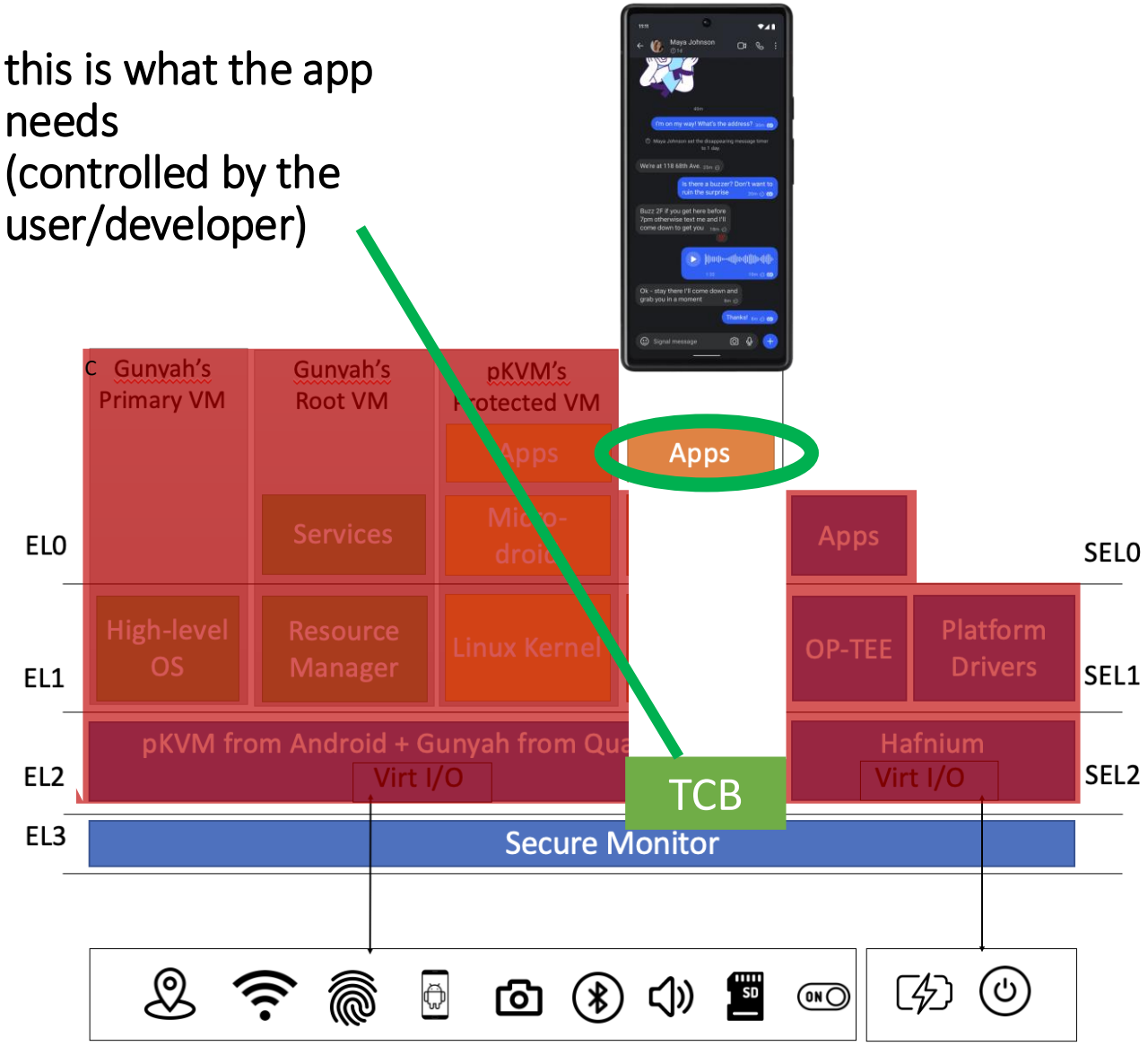


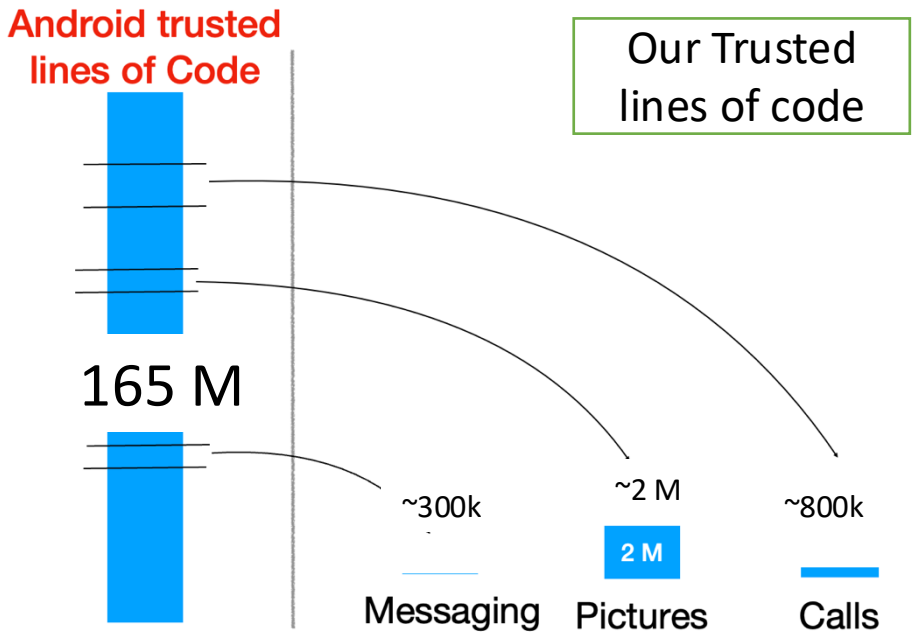
Phone:
Punkt. MC03
Dual OS by Soverli

Configuration:
Domain 1: Android
Domain 2: Linux with WebApps

# Sovereignty for Smartphones

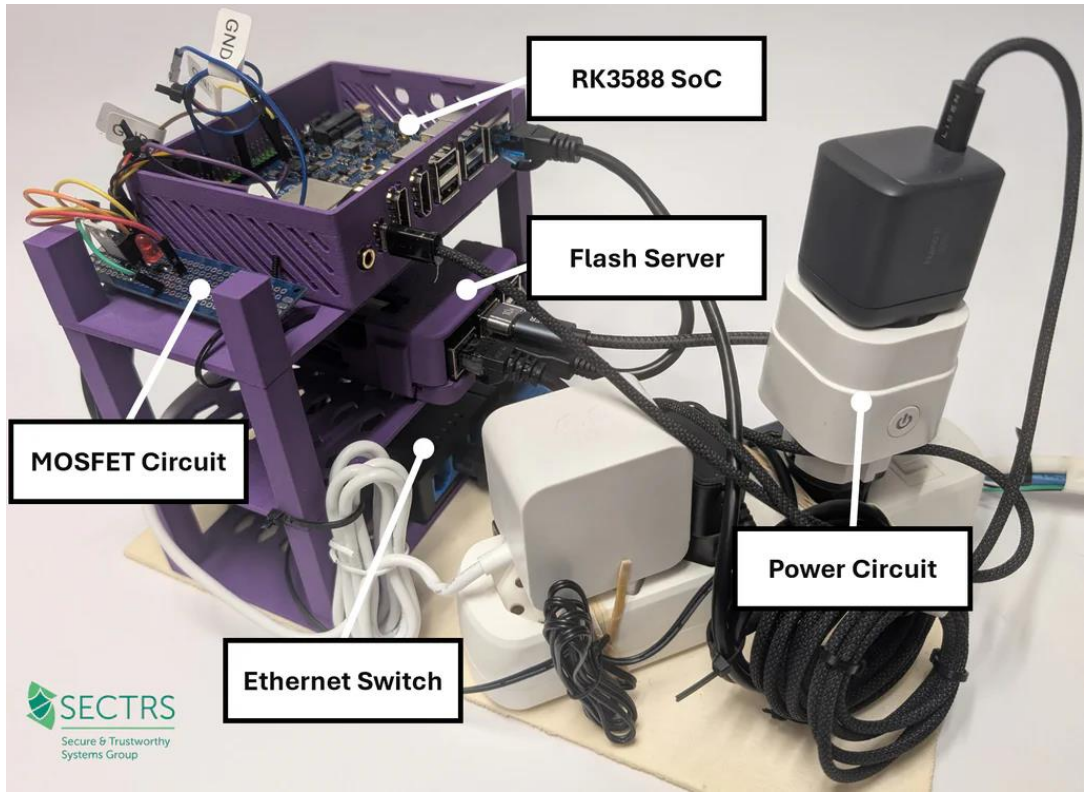
this is what the app needs
(controlled by the user/developer)



# What about next generation of hardware?

# It doesn’t exist yet publicly, so…



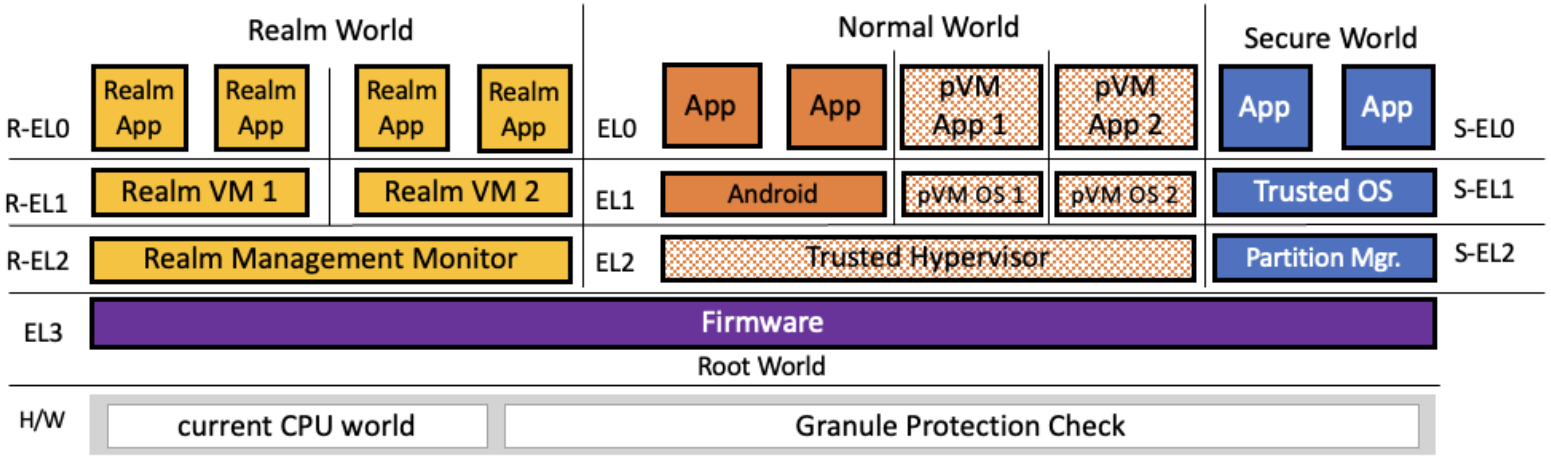


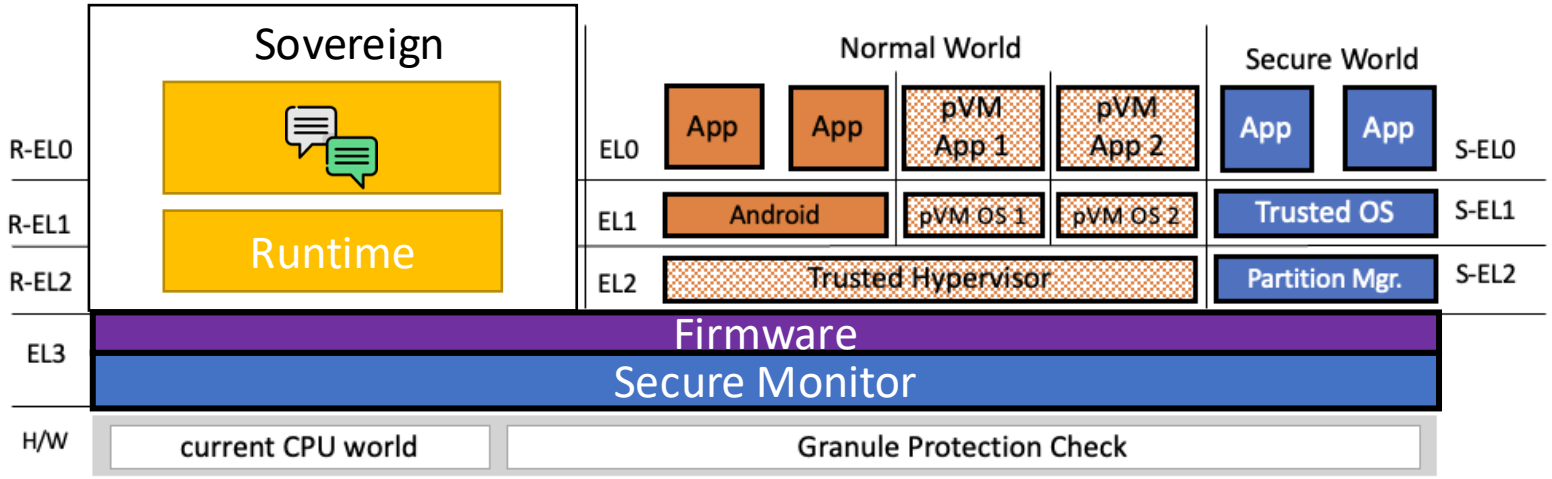


https://opencca.github.io

# Sovereign Smartphone for next generation





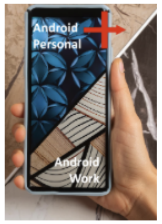
# Sovereign Smartphone

To Enjoy Freedom We Have to Control Our Phones.

Modern smartphones are complex systems where control over phone resources is exercised by phone manufacturers, operators, OS vendors, and users. Users are generally limited in what applications they can install on their devices, in the privacy model and trust assumptions of the existing applications, and in the functionalities that applications can have. Our goal is to enable users to install sensitive applications in isolated domains with protected access to selected peripherals alongside an existing OS.

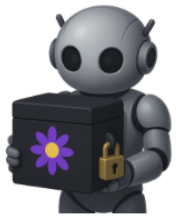
Under the umbrella of the sovereign smartphone project, we explore designs that maintain compatibility with the existing smartphone ecosystem but allow the balancing of the control over the phones. Our architectures implement novel TEE-based mechanisms that allow the users to flexibly choose which resources (including peripherals) to dedicate to different isolated domains to securely execute applications.

## Projects




**TEEtime**

A new smartphone architecture that gives control of execution back to users.

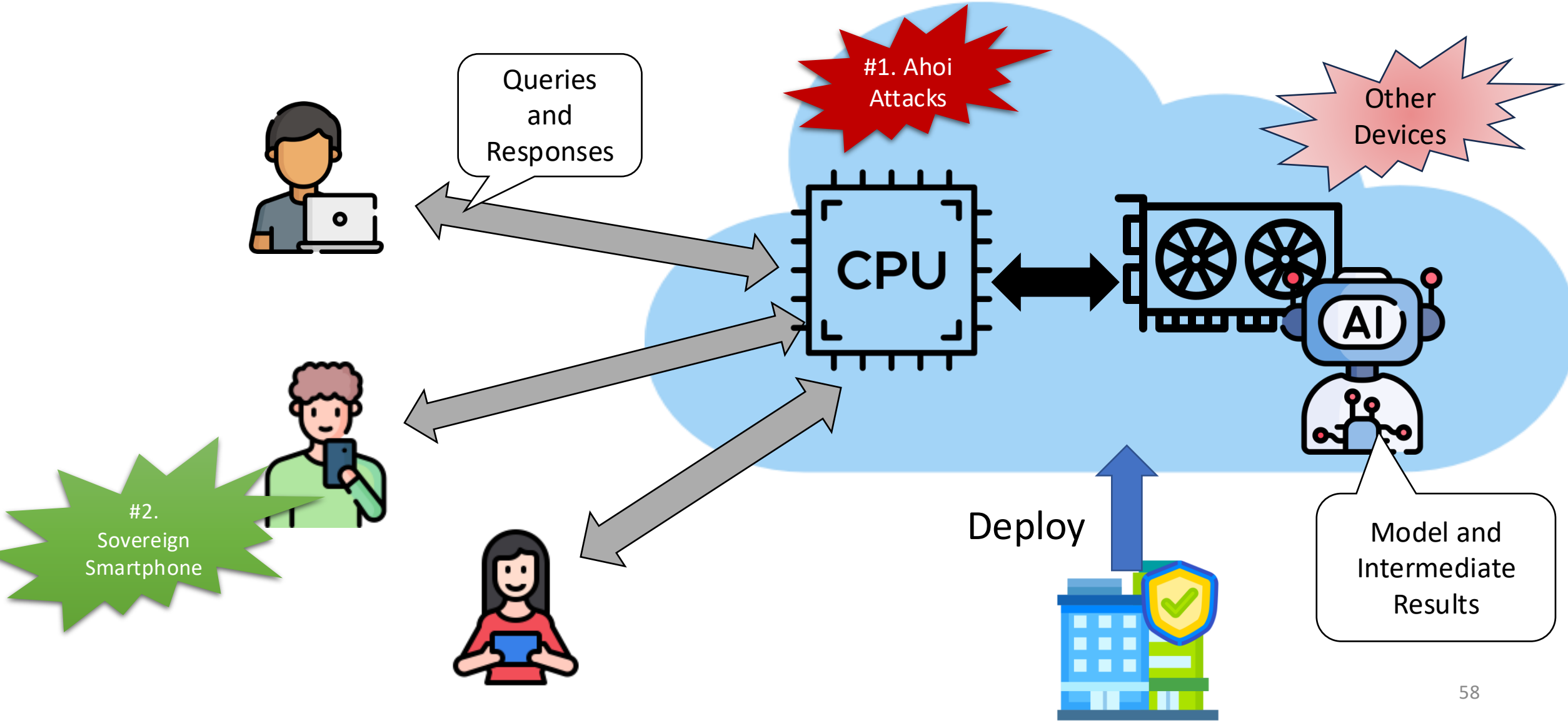


**Aster**

Enabling confidential VMs for Android-based devices using Arm CCA.

https://sovereign-smartphone.github.io

# Have we made progress?

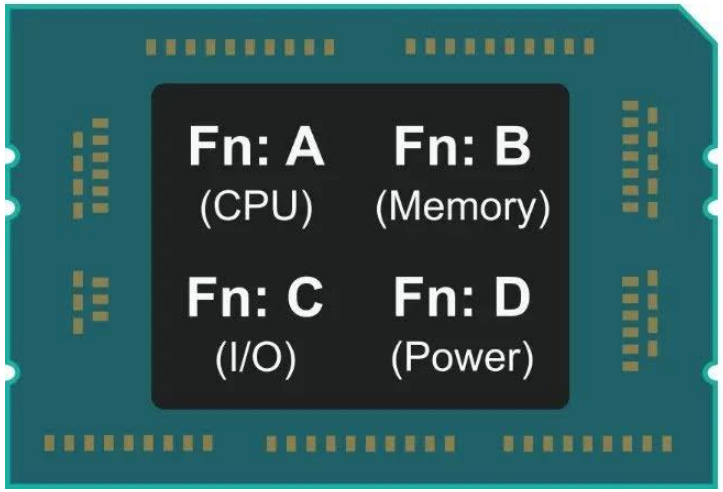


# The way “chips” are built has changed…

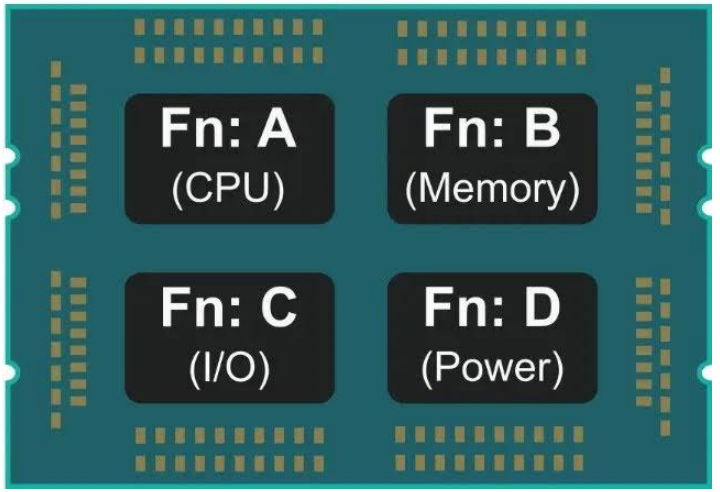
*“It may prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected. The availability of large functions, combined with functional design and construction, should allow the manufacturer of large systems to design and construct a considerable variety of equipment both rapidly and economically.”*

*Gordon Moore, 1965 (from the original paper and prediction that became Moore’s Law)*

Monolithic Design Example




Chiplet Design Illustration




https://semiengineering.com/

# AMD System-on-Chip (SoC)

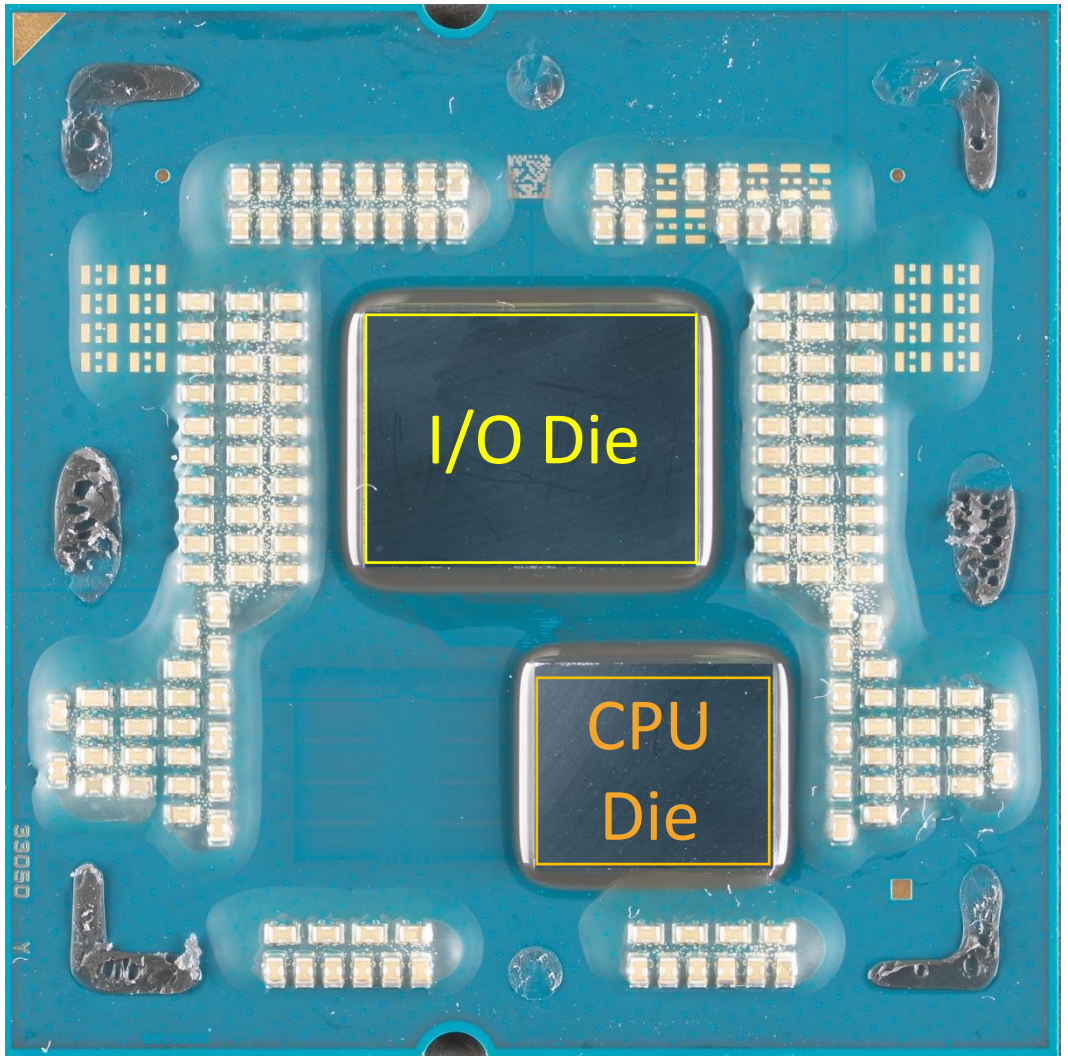
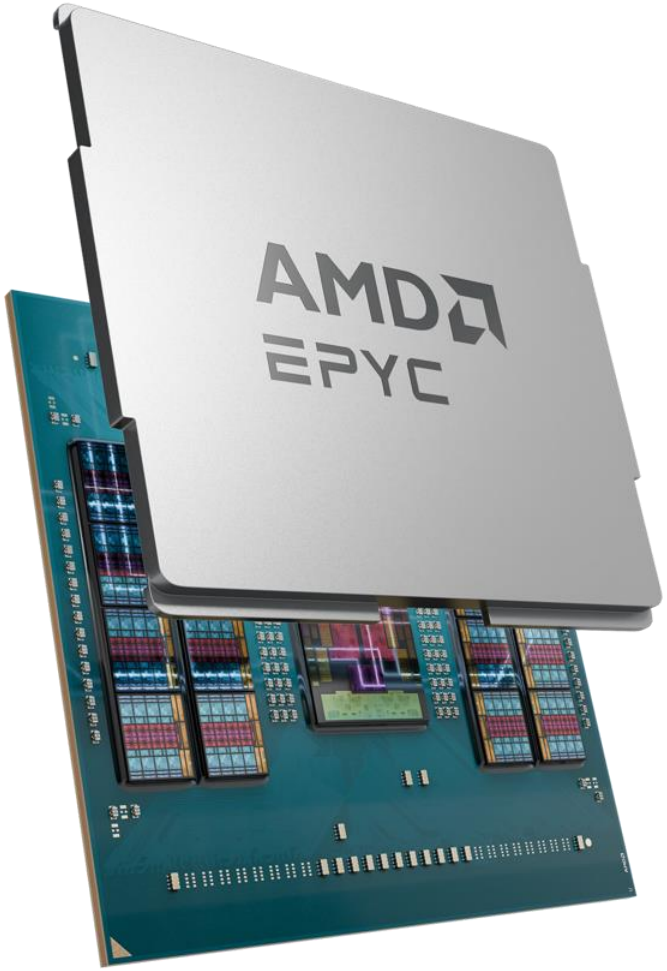
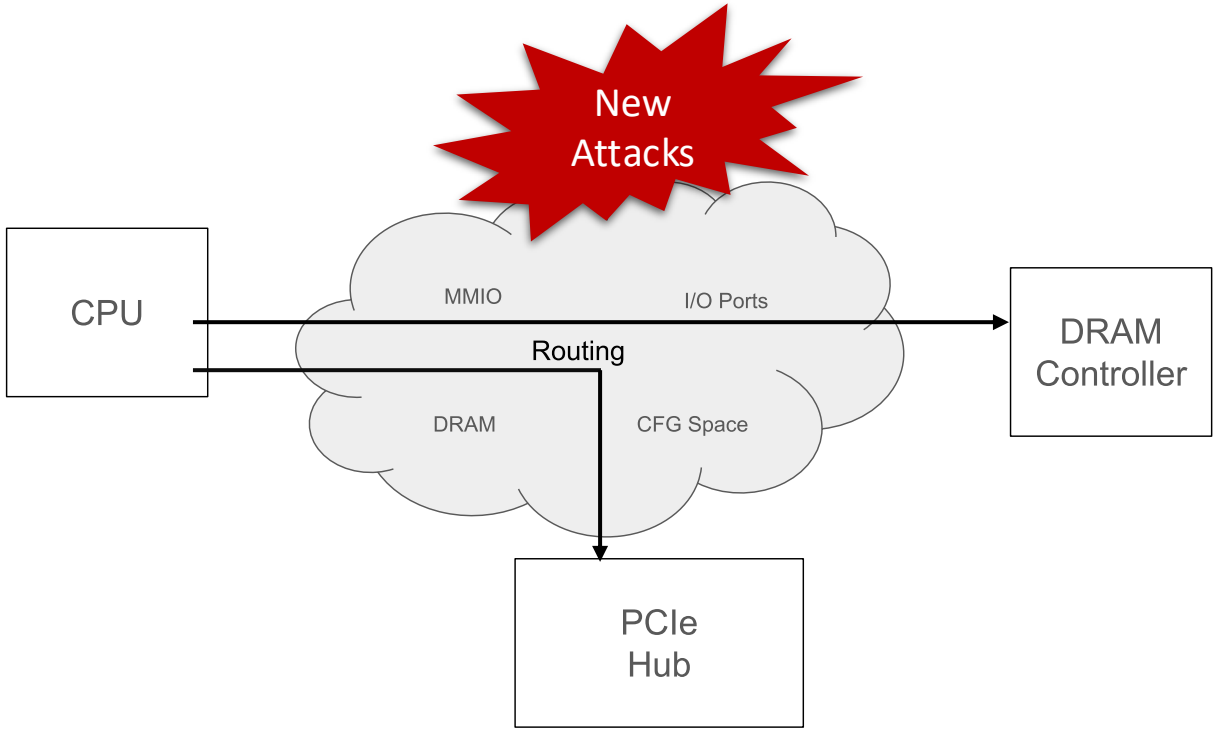


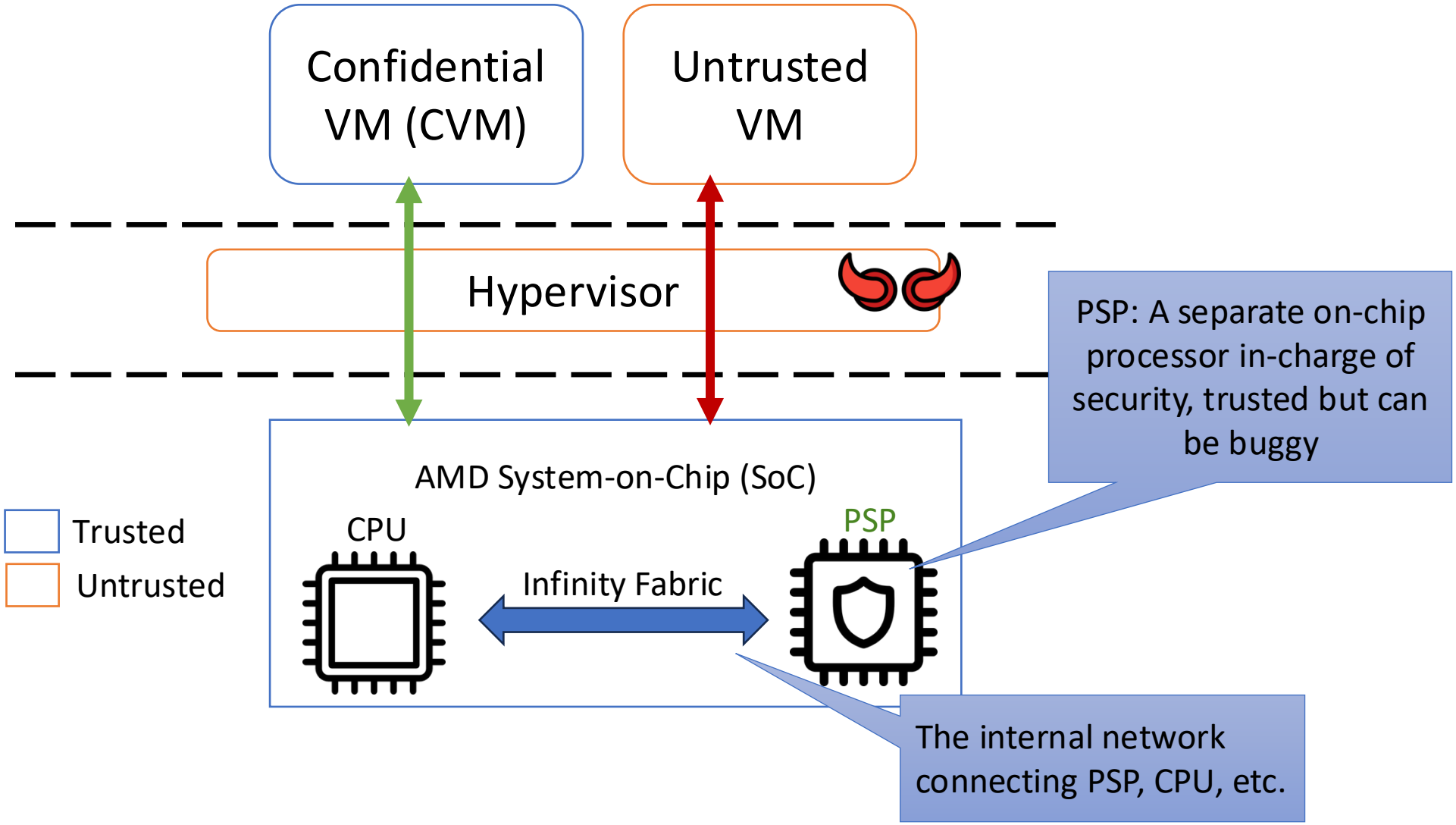

Image: Fitzchens Fitz

# The way we reason about their security hasn't

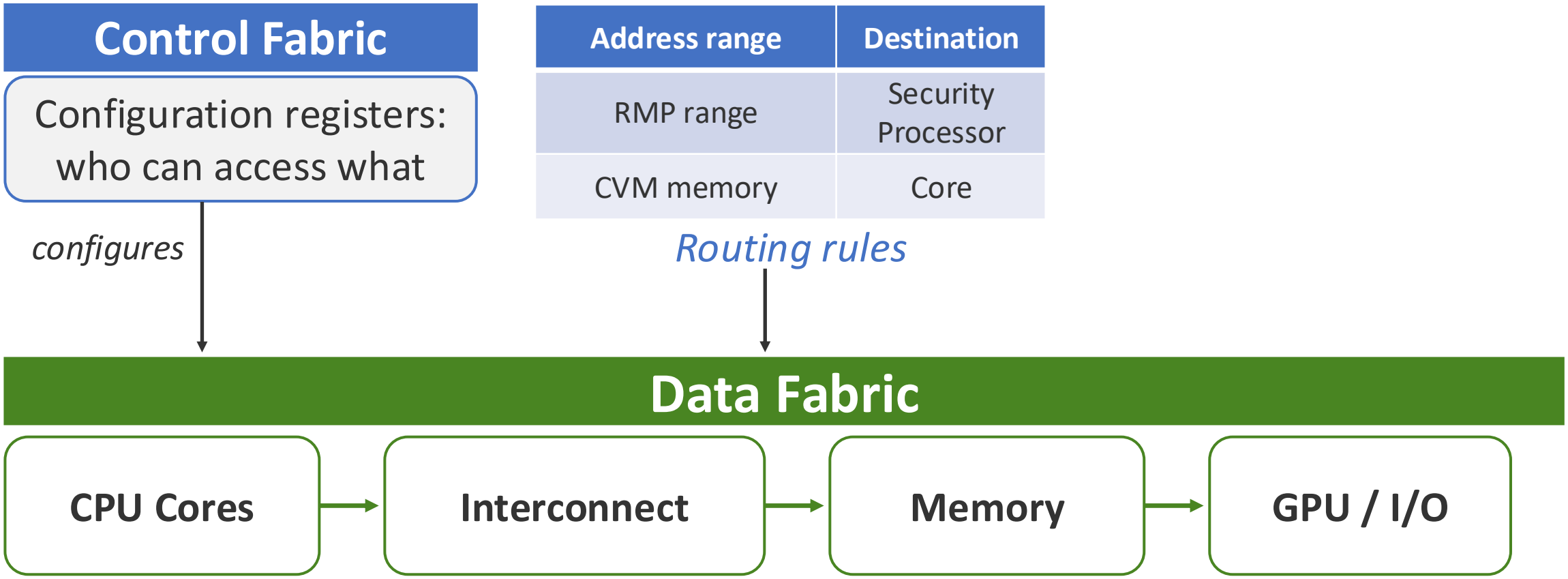


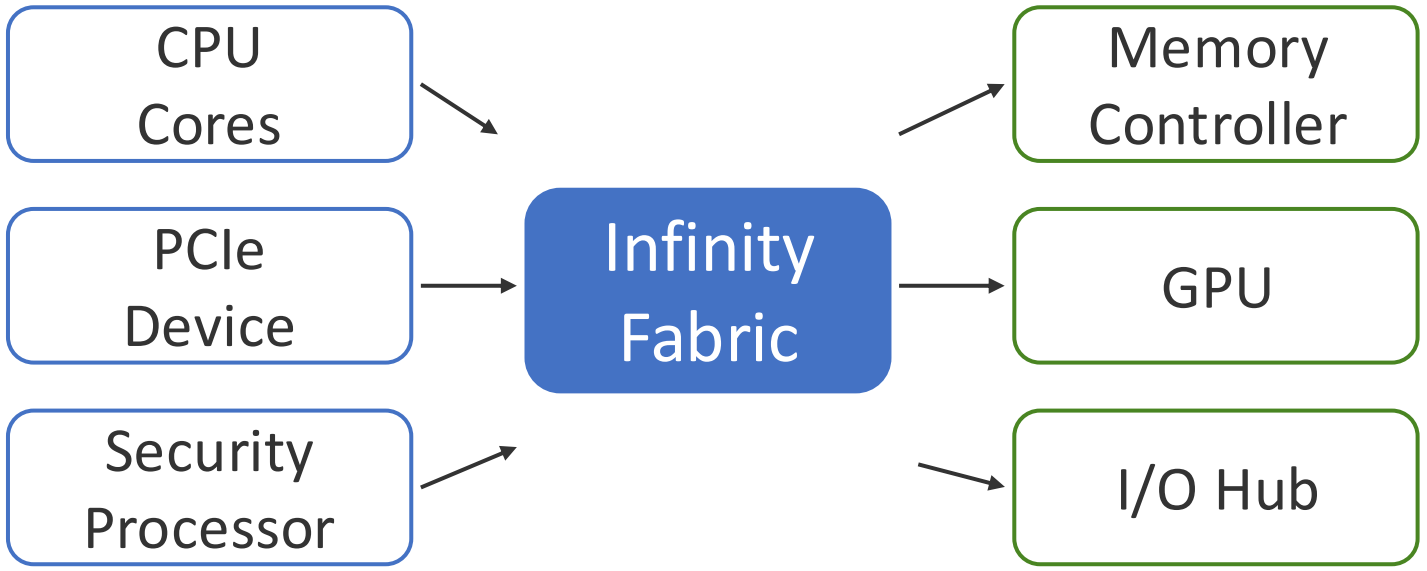
# AMD Infinity Fabric & Confidential Computing



# Infinity Fabric: Control and Data Planes

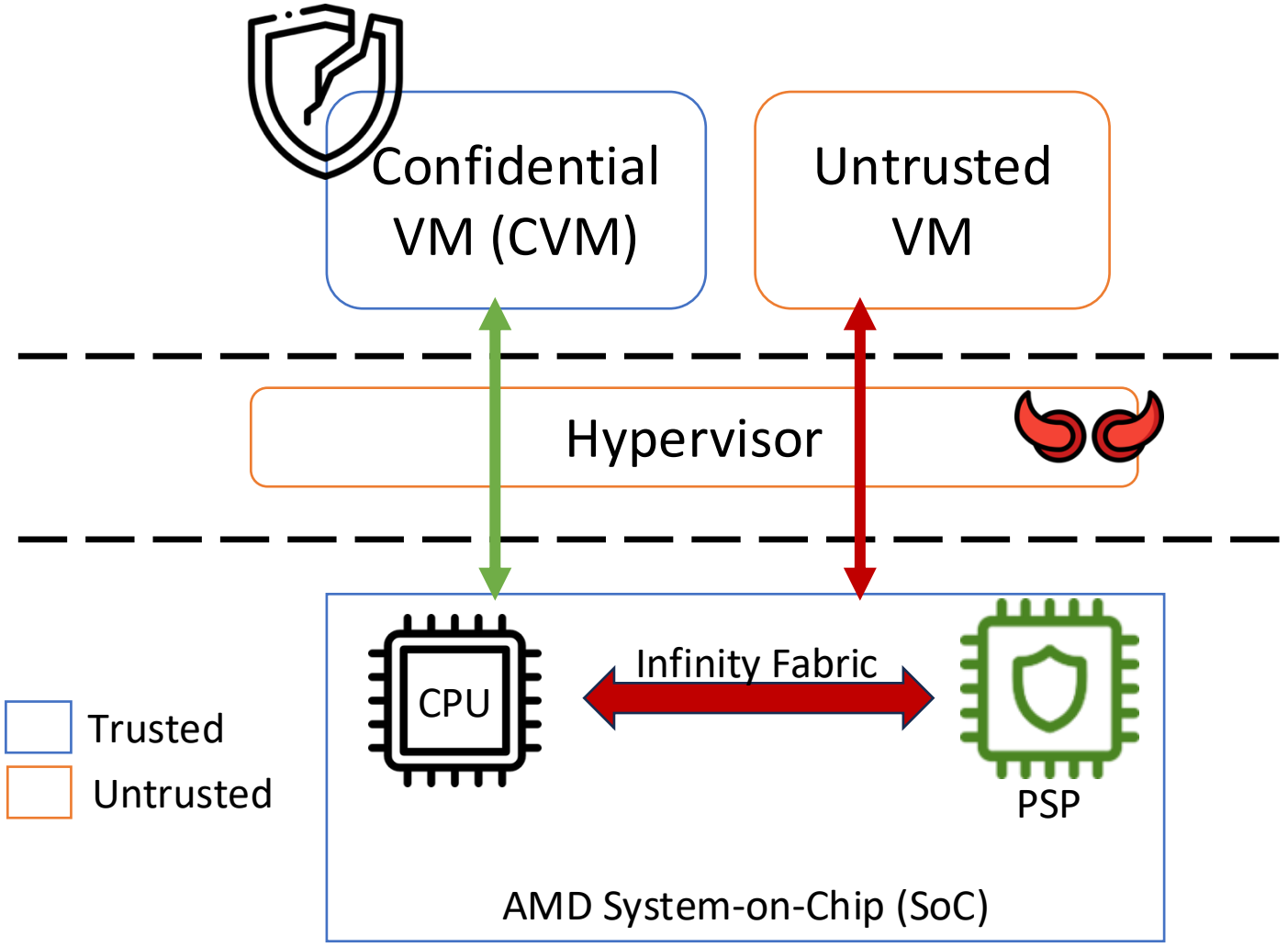


# AMD's beautifully engineered routing



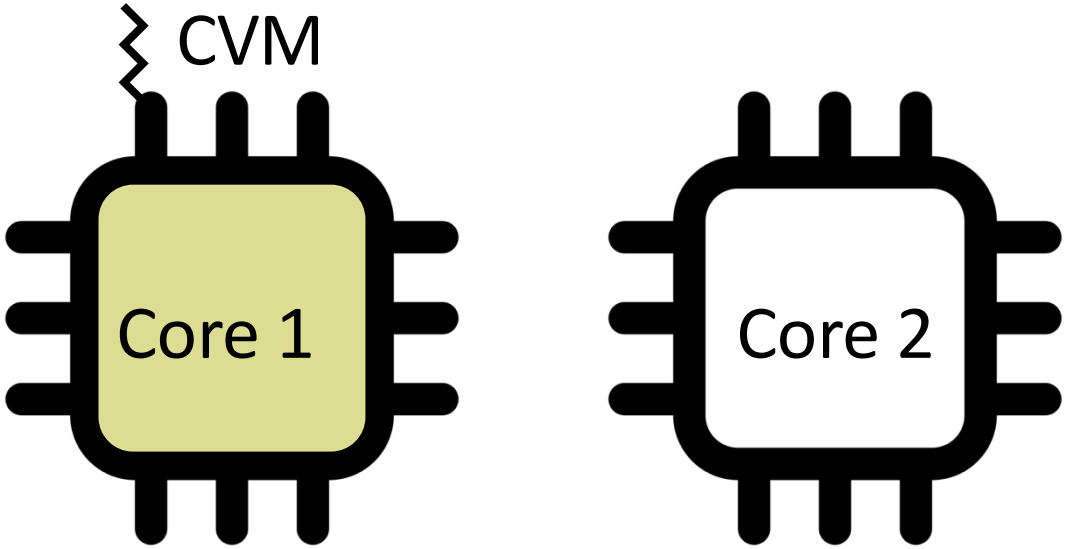

| Address range | Destination |
|---|---|
| 0xa000–b000 | Memory |
| 0xb000–c000 | GPU |

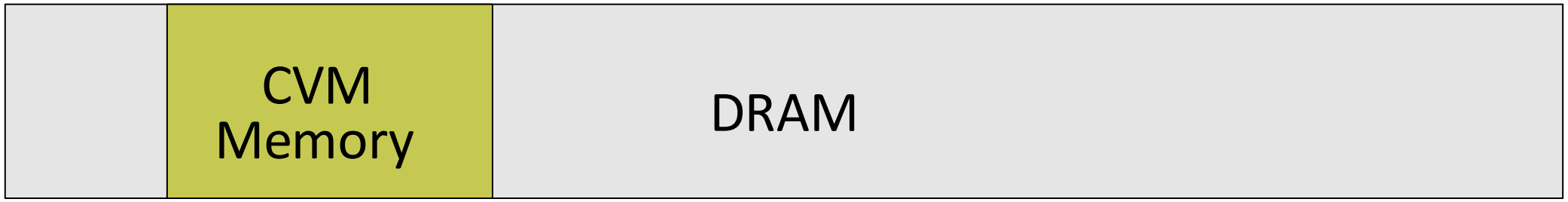
# Interconnect Corruption Attacks (XCA)



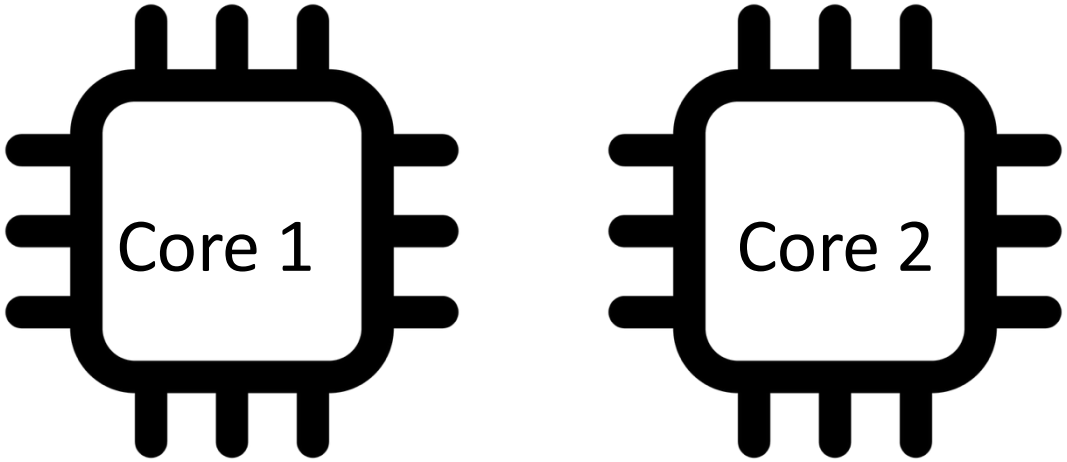
# XCA #1: Fabricked

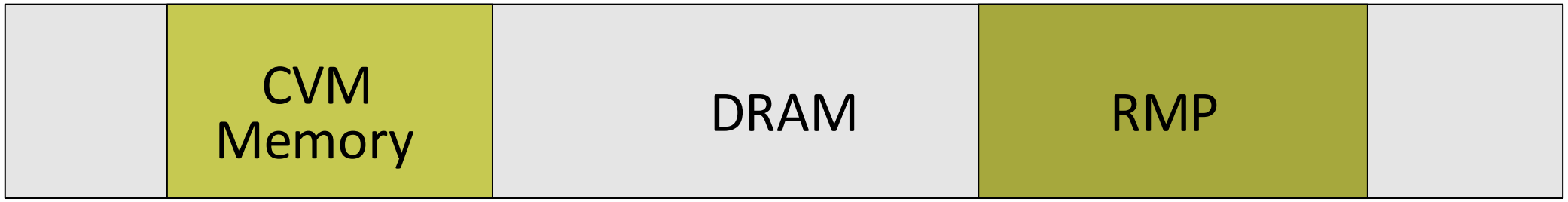
# AMD Confidential Computing



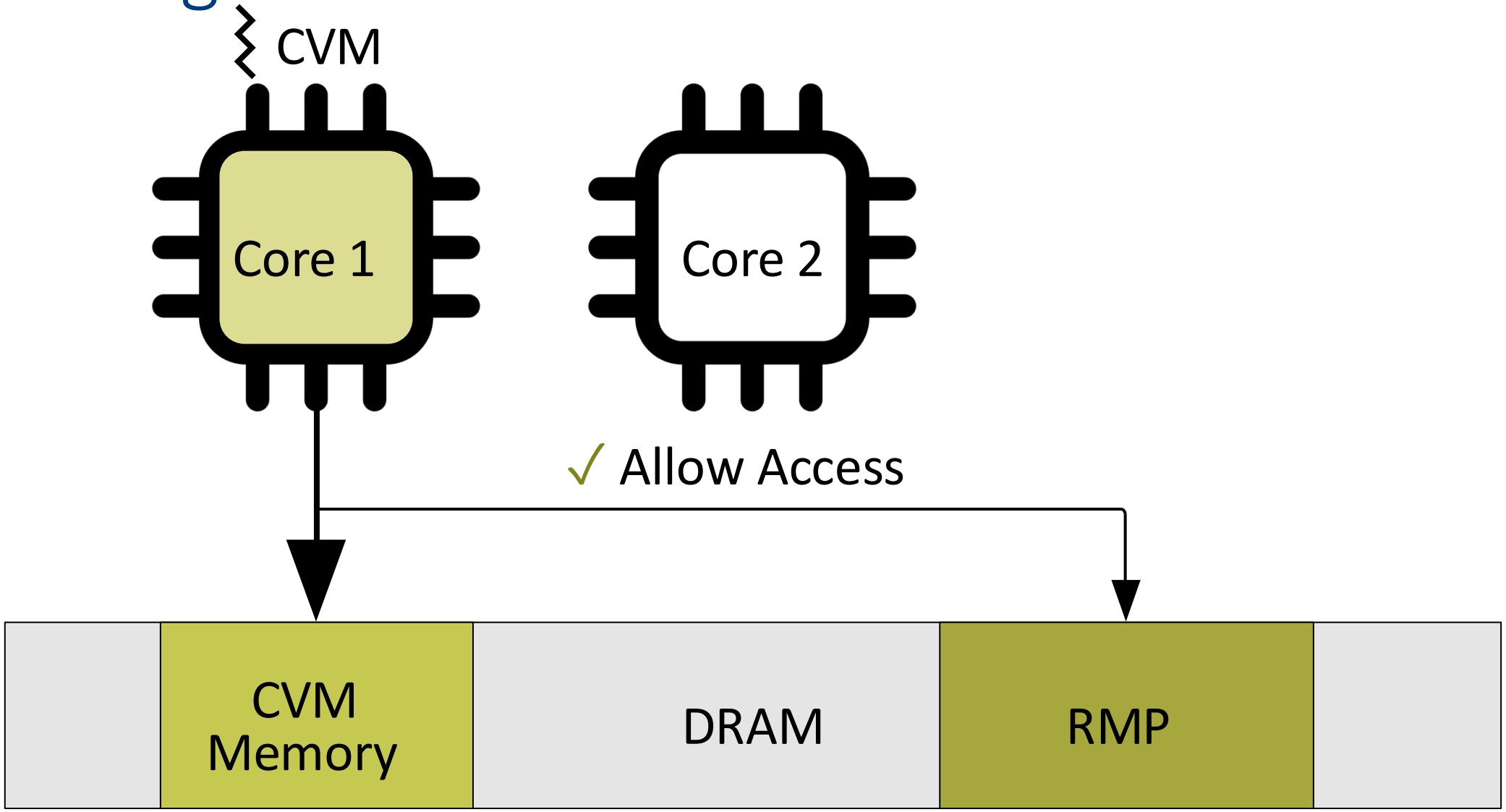



# Critical Data Structure: Reverse Map Table (RMP)

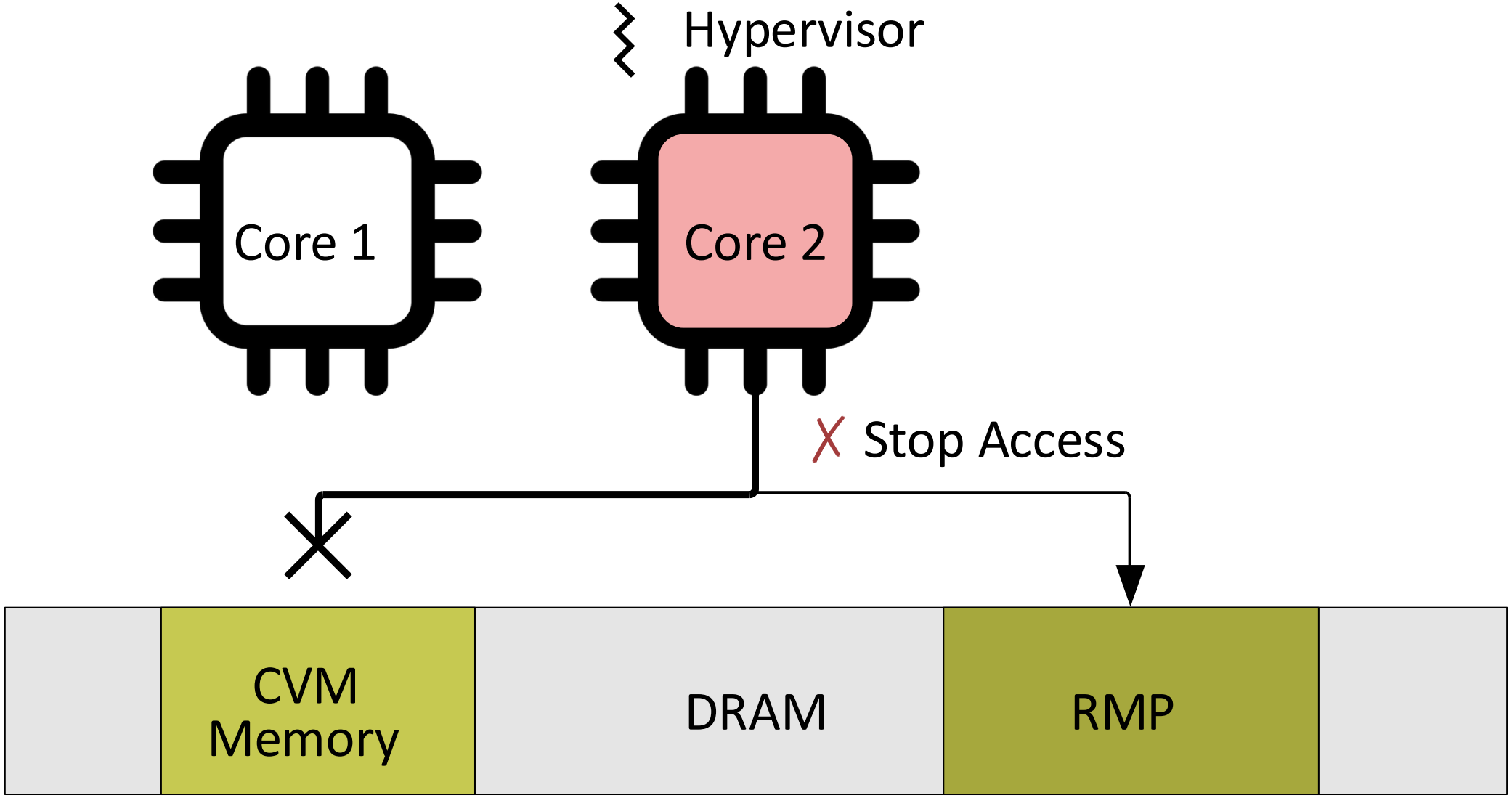


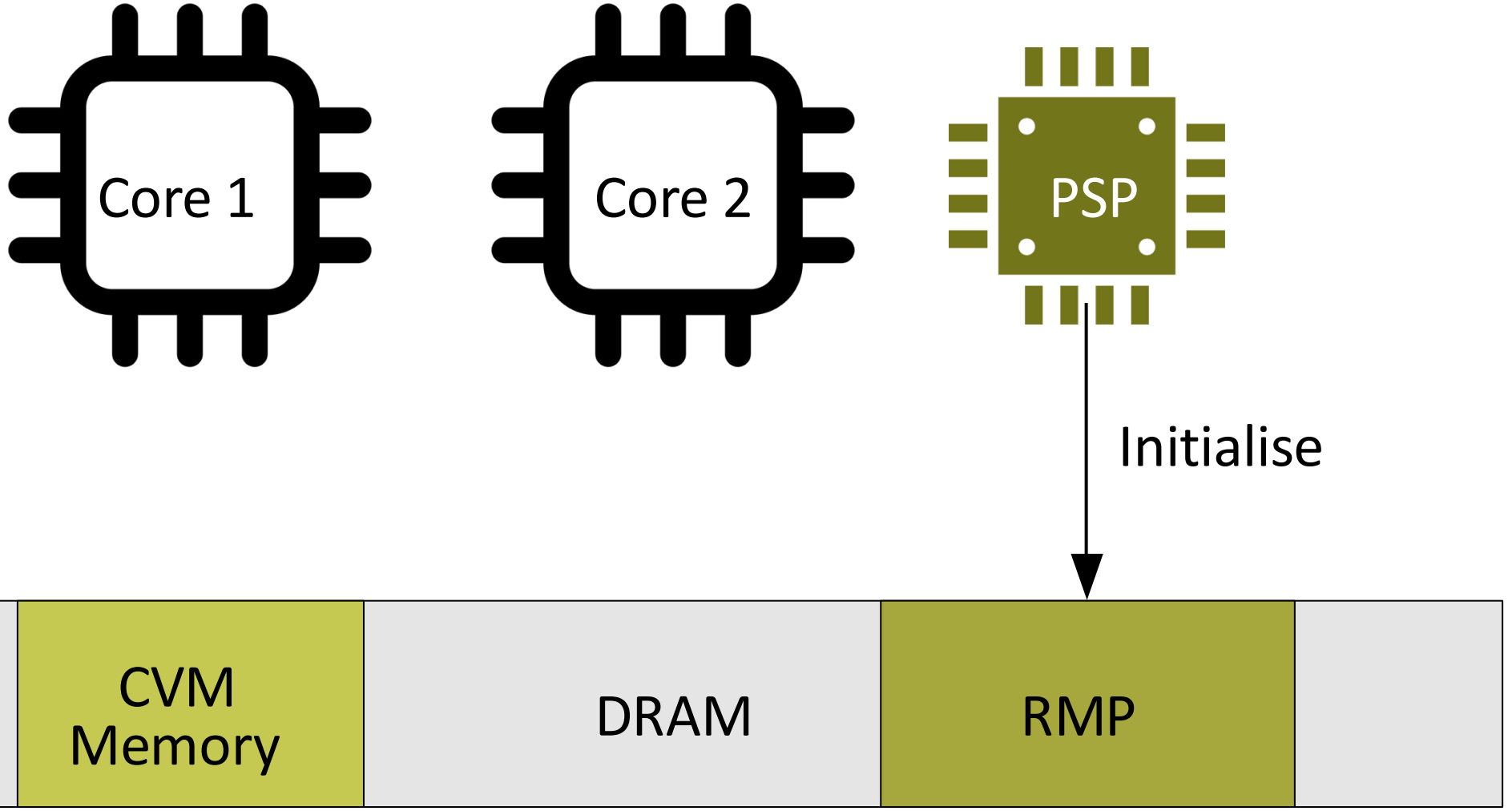
# Allowing CVM Access



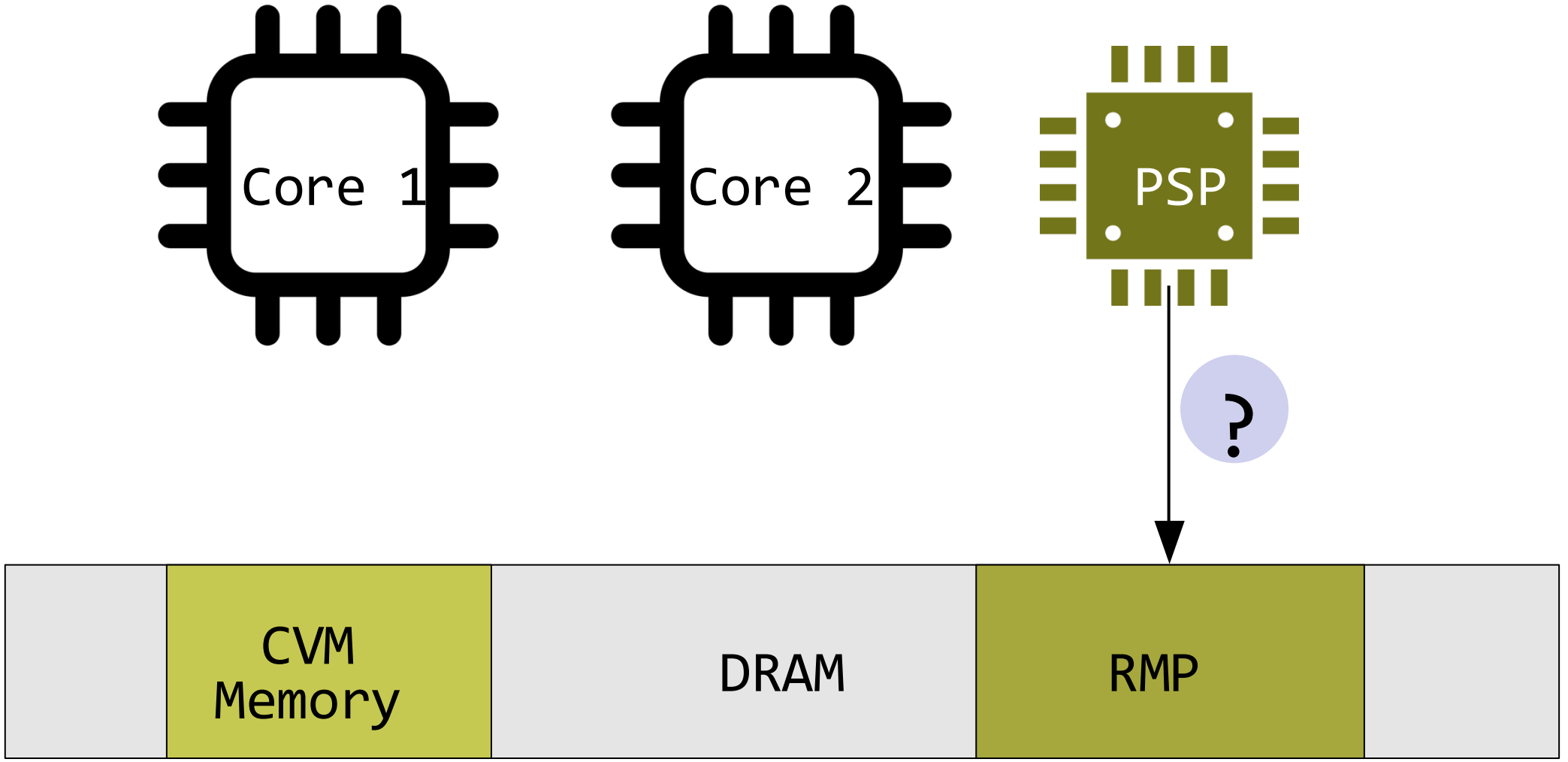
# Blocking Unauthorized Access



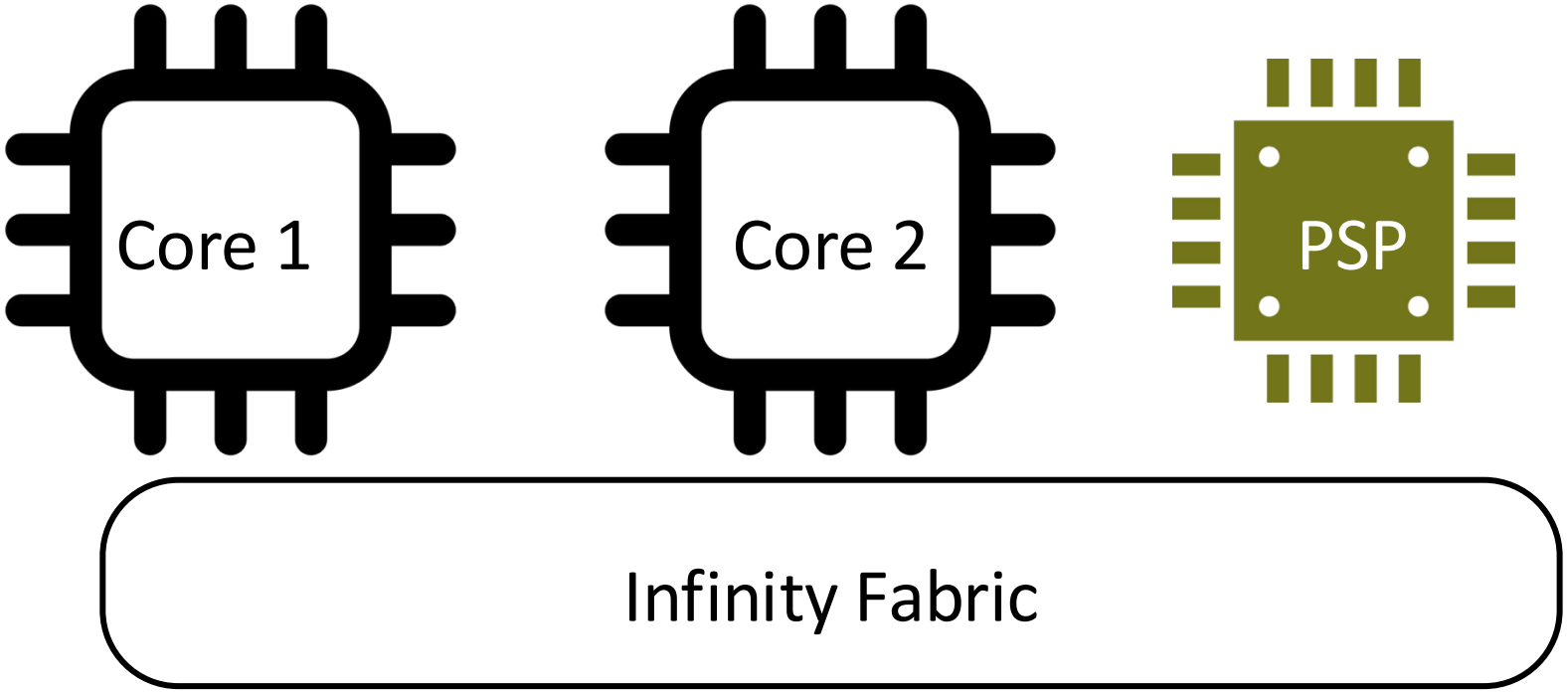
# Security Processor Initializes RMP

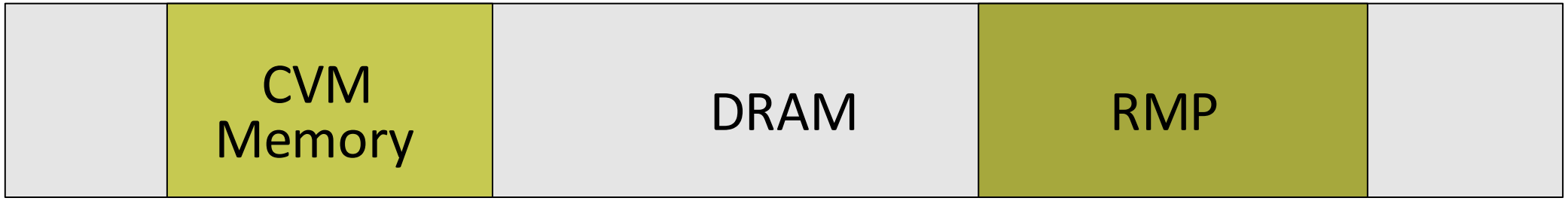


# RMP Initialization

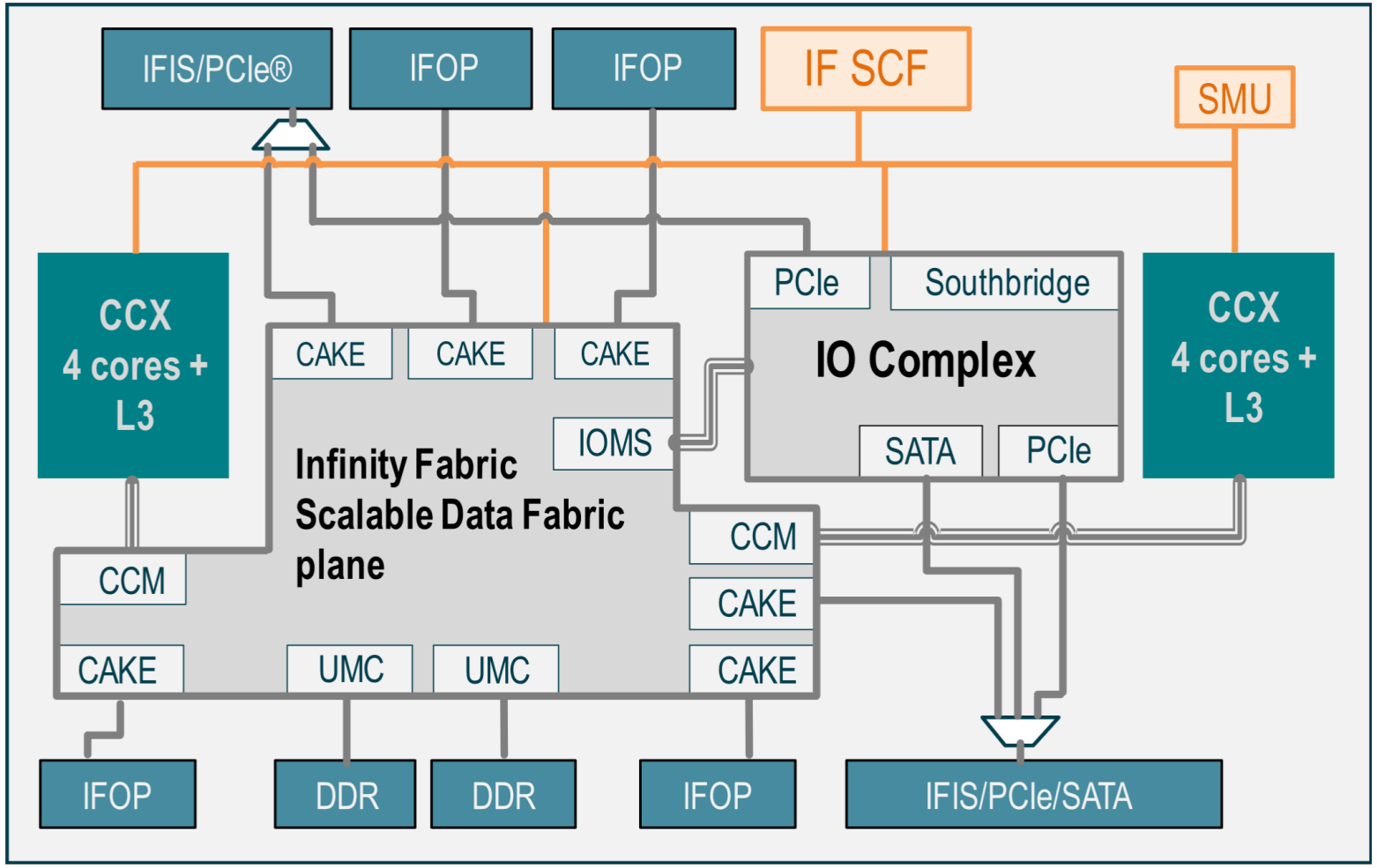


# AMD Memory Routing



# Challenge: Reverse Engineering Infinity Fabric




Thomas Burd, Noah Beck, Sean White, Milam Paraschou, Nathan Kalyanasundharam, Gregg Donley, Alan Smith, Larry Hewitt, and Samuel Naffziger. "Zeppelin": An SoC for Multichip Architectures. IEEE Journal of Solid-State Circuits, 2019

# Infinity Fabric: Question 1

Question 1:

Who configures the Infinity Fabric routing?

# Infinity Fabric: Question 1

Question 1:

Who configures the Infinity Fabric routing?

Answer:

- Security Processor (PSP)

# Infinity Fabric: Question 1

Question 1:

Who configures the Infinity Fabric routing?

Answer:

- Security Processor (PSP)
- In one case, BIOS

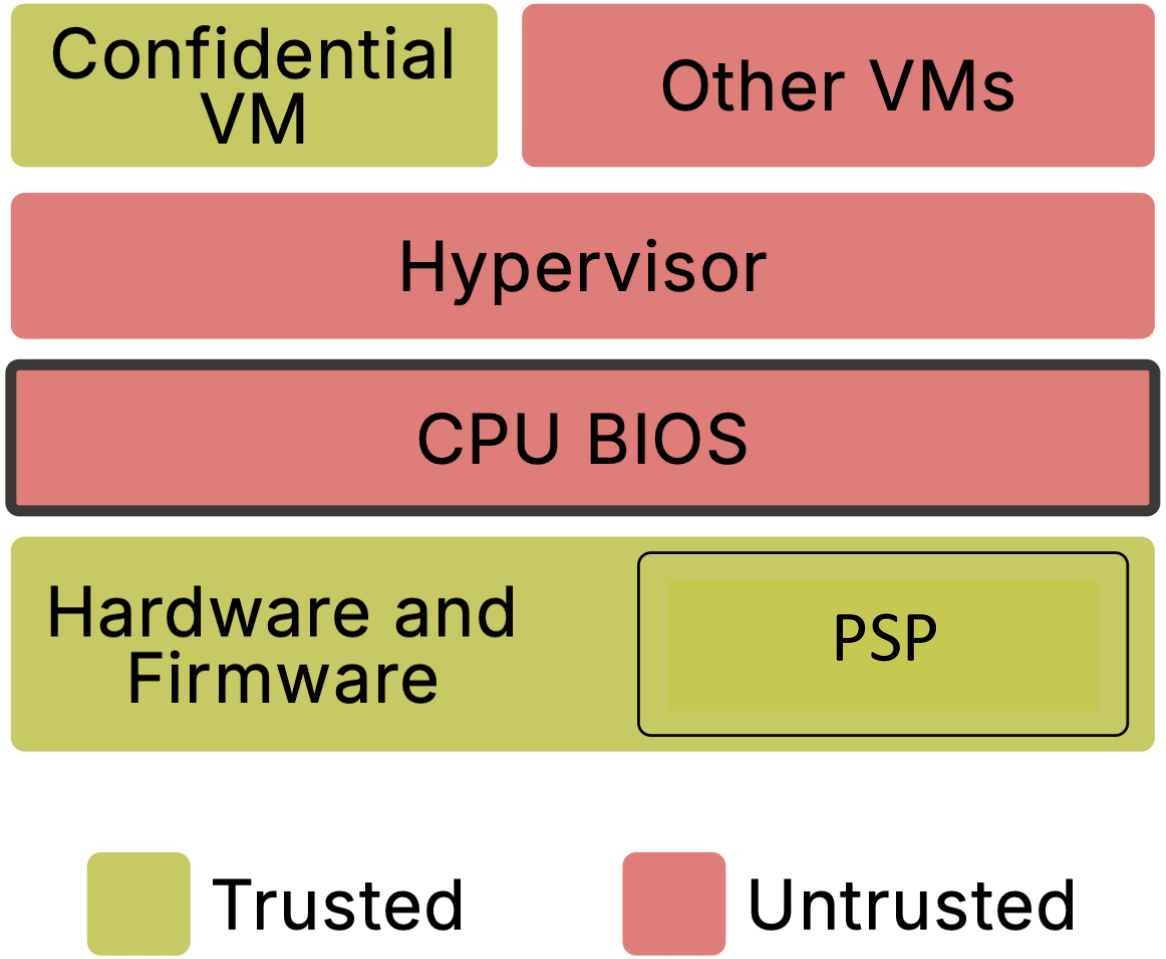
# Infinity Fabric: Question 1

Question 1:

Who configures the Infinity Fabric routing?

Answer:

- Security Processor (PSP)
- In one case, BIOS



# Infinity Fabric: Question 2

Question 1:

Who configures the Infinity Fabric routing?

Question 2:

What happens if BIOS (maliciously) misconfigures routing?

Answer:

- Security Processor (PSP)
- In one case, BIOS

# Infinity Fabric: Question 2

Question 1:

Who configures the Infinity Fabric routing?

Answer:

- Security Processor (PSP)
- In one case, BIOS

Question 2:

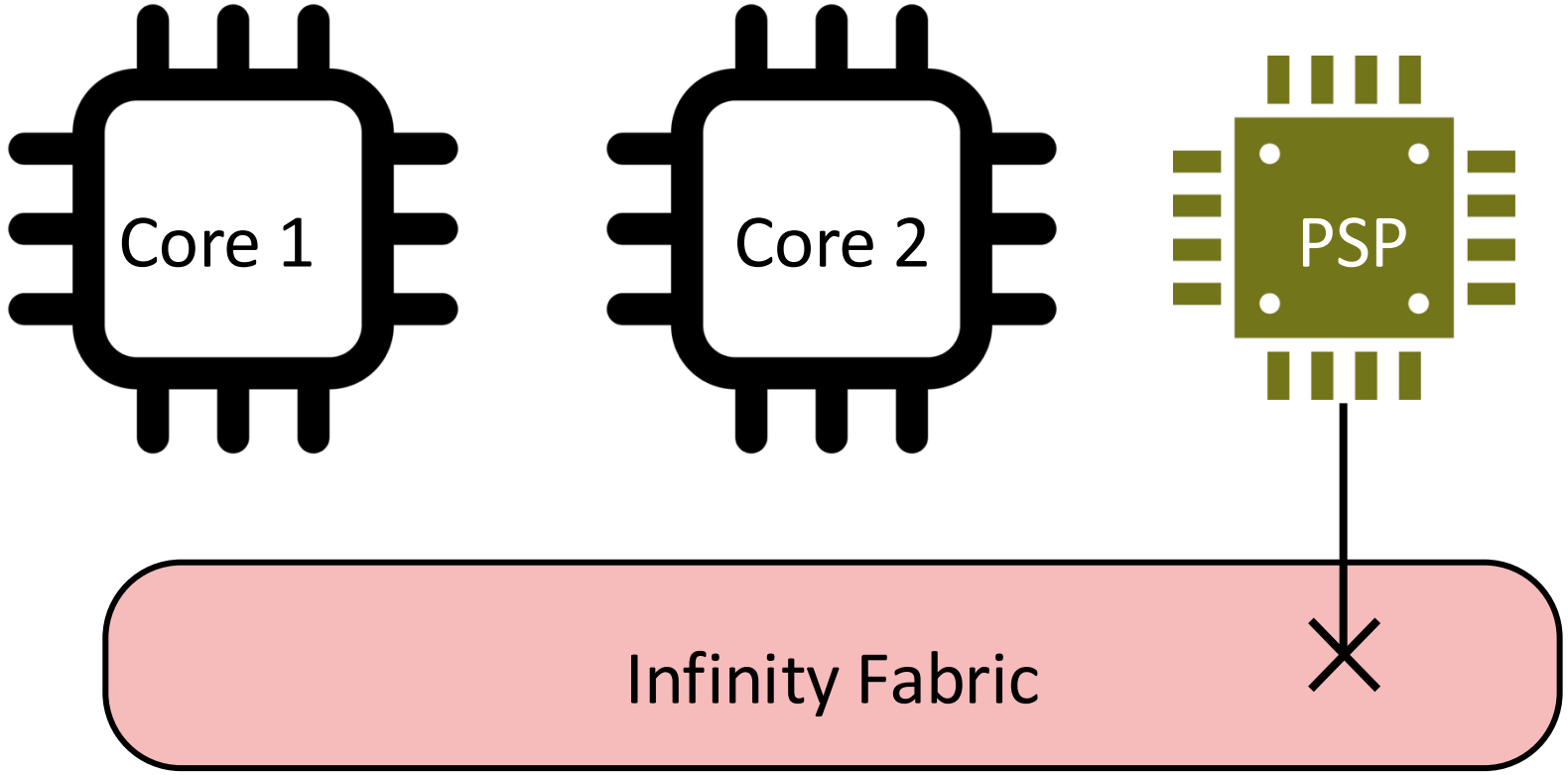
What happens if BIOS (maliciously) misconfigures routing?

Answer:

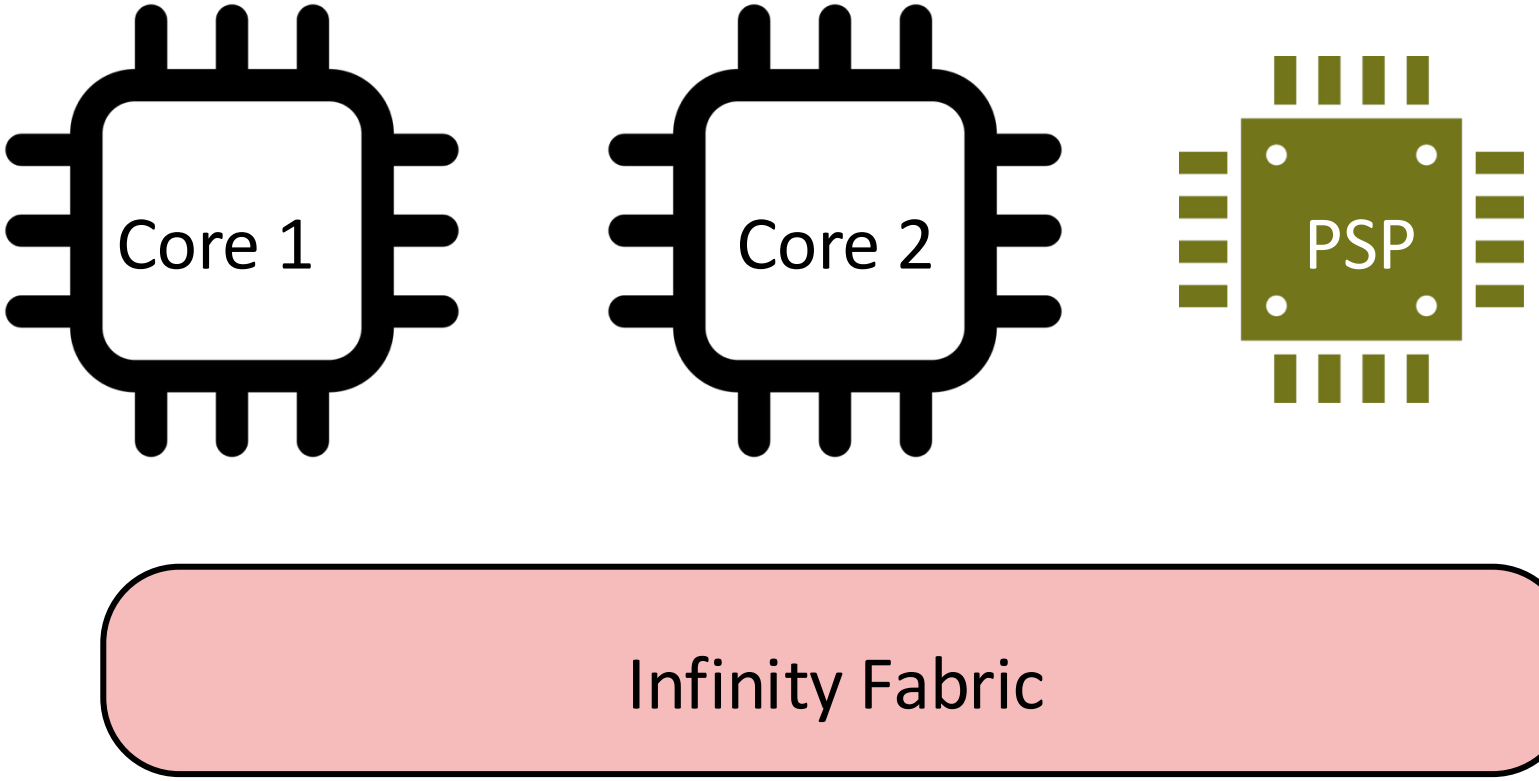
Fabricked Attack



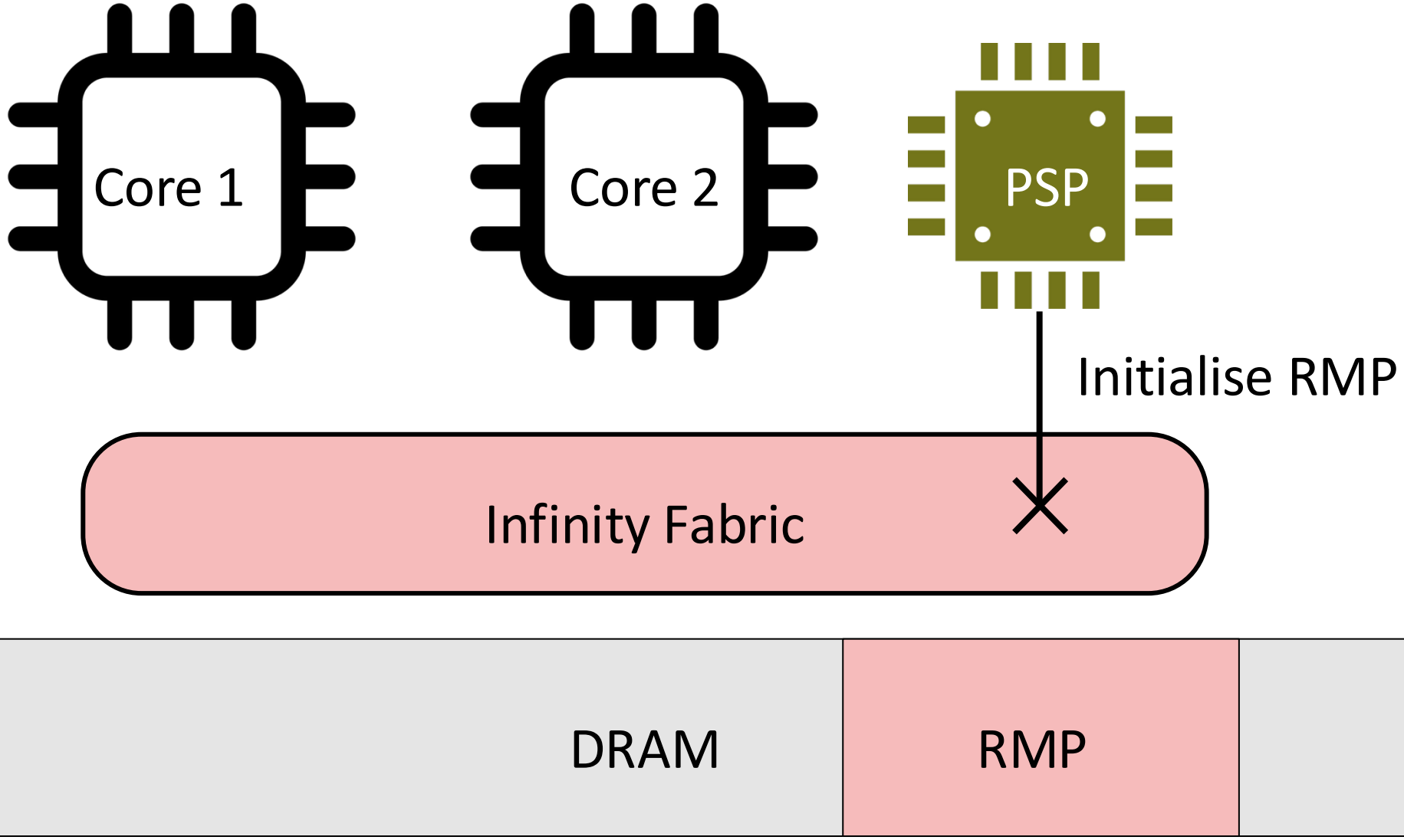
# Fabricked Attack: Dropping Writes



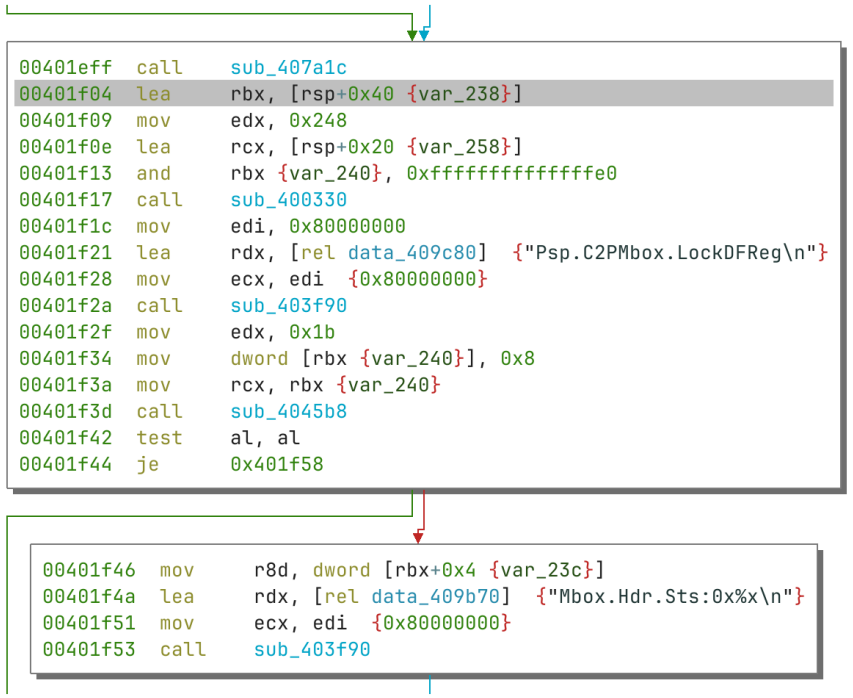
# Fabricked Attack: Compromising RMP Initialization



# Fabricked Attack: Compromising RMP Initialization



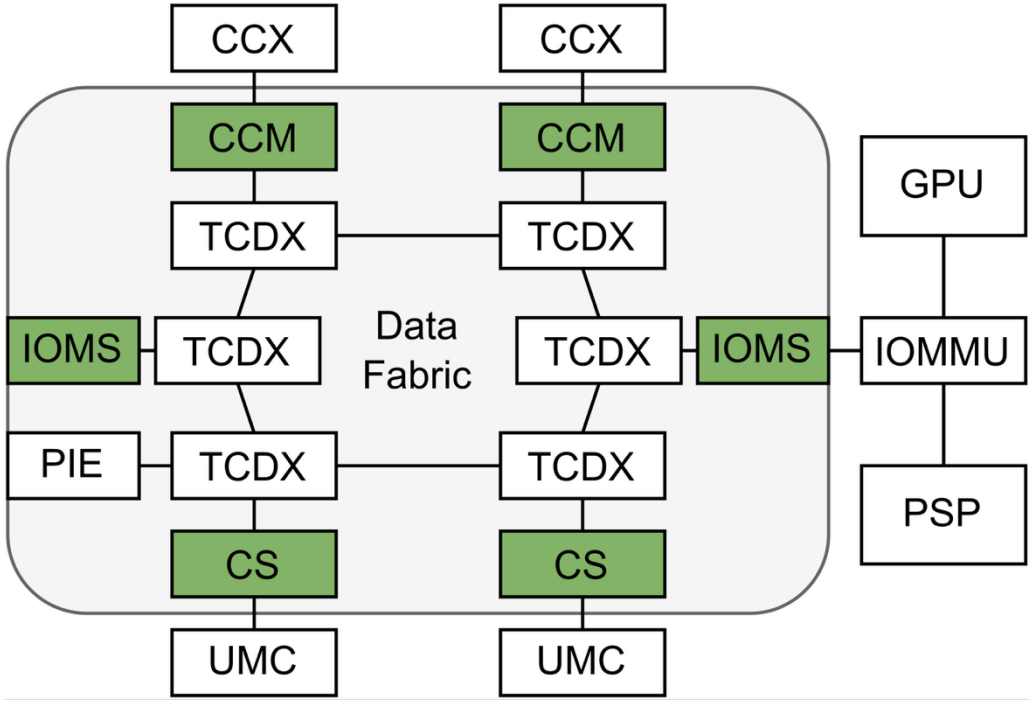
# Additional Challenges



```
00401eff  call    sub_407a1c
00401f04  lea     rbx, [rsp+0x40 {var_238}]
00401f09  mov     edx, 0x248
00401f0e  lea     rcx, [rsp+0x20 {var_258}]
00401f13  and     rbx {var_240}, 0xfffffffffffffe0
00401f17  call    sub_400330
00401f1c  mov     edi, 0x80000000
00401f21  lea     rdx, [rel data_409c80]  {"Psp.C2PMbox.LockDFReg\n"}
00401f28  mov     ecx, edi  {0x80000000}
00401f2a  call    sub_403f90
00401f2f  mov     edx, 0x1b
00401f34  mov     dword [rbx {var_240}], 0x8
00401f3a  mov     rcx, rbx {var_240}
00401f3d  call    sub_4045b8
00401f42  test    al, al
00401f44  je      0x401f58
```

```
00401f46  mov     r8d, dword [rbx+0x4 {var_23c}]
00401f4a  lea     rdx, [rel data_409b70]  {"Mbox.Hdr.Sts:0x%x\n"}
00401f51  mov     ecx, edi  {0x80000000}
00401f53  call    sub_403f90
```

BIOS Reversing




Inferring Routing Behaviour

# Fabricked Breaks AMD SEV-SNP

- Demonstrated two end-to-end case studies:
  - Full guest read/write
  - Forged attestation report
- AMD released firmware patches

# Interconnect Corruption Attacks (XCA)



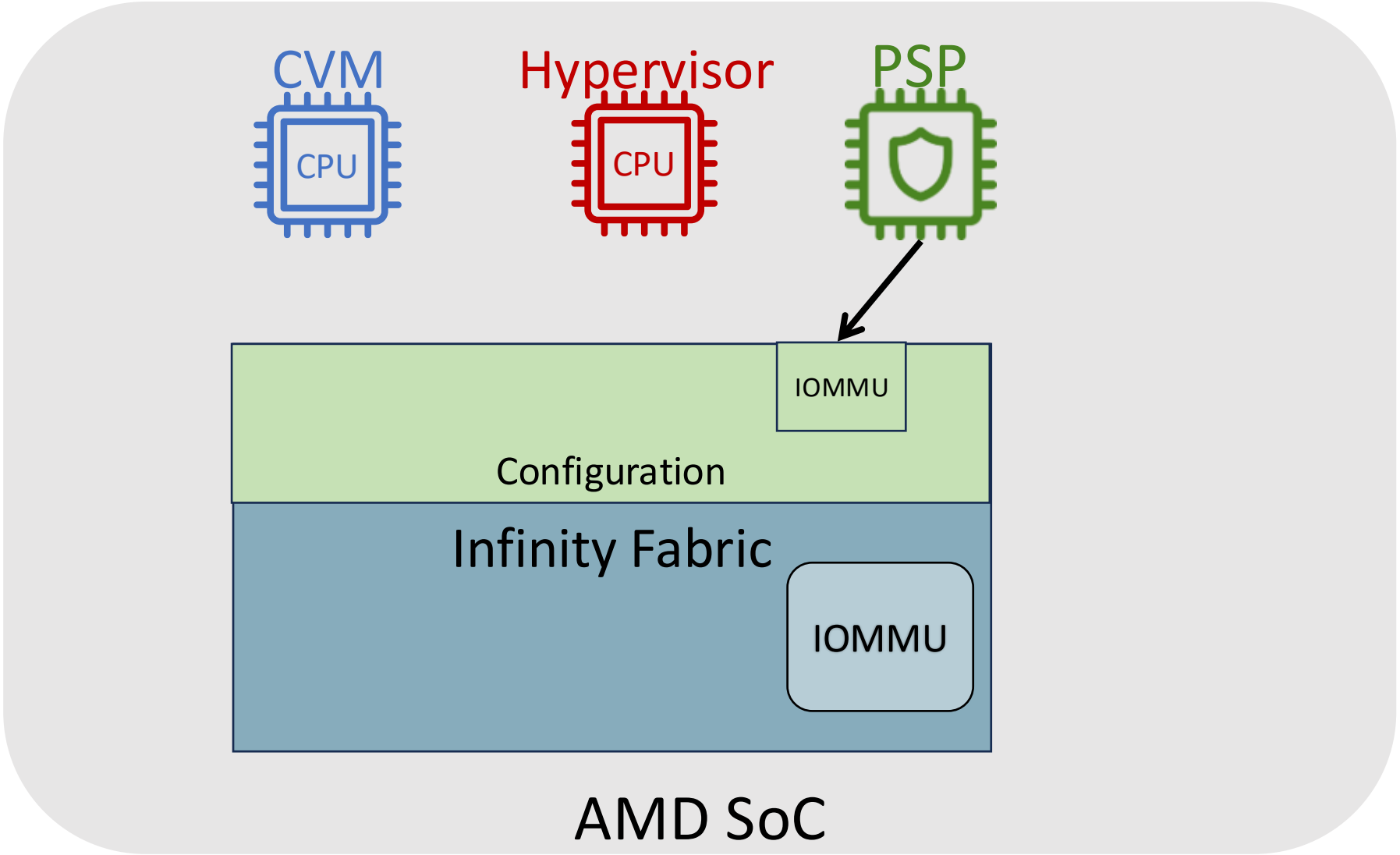
Fabricked

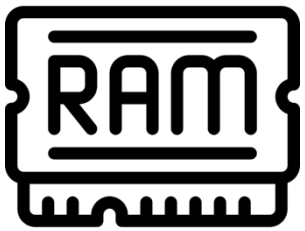
Corrupts the Data Plane

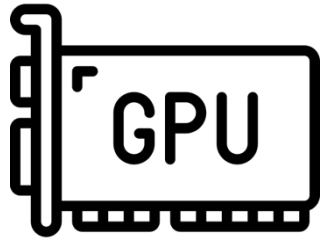
CVE-2025-54510

# XCA #2: BreakFast

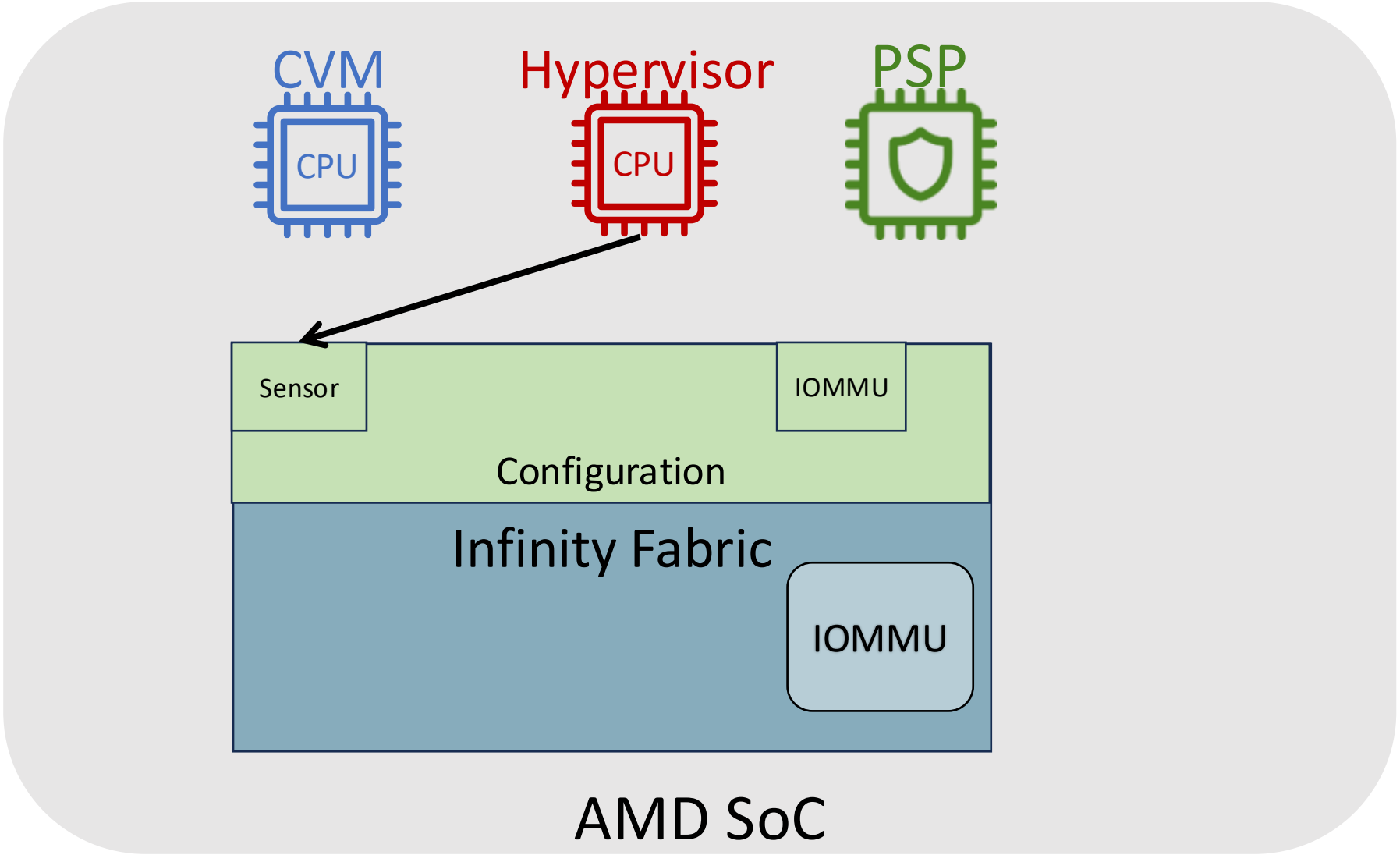
# Infinity Fabric Configuration: by PSP

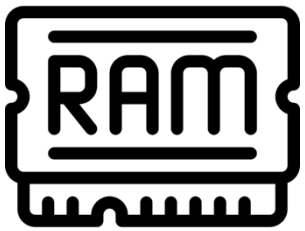


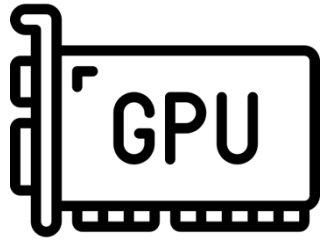



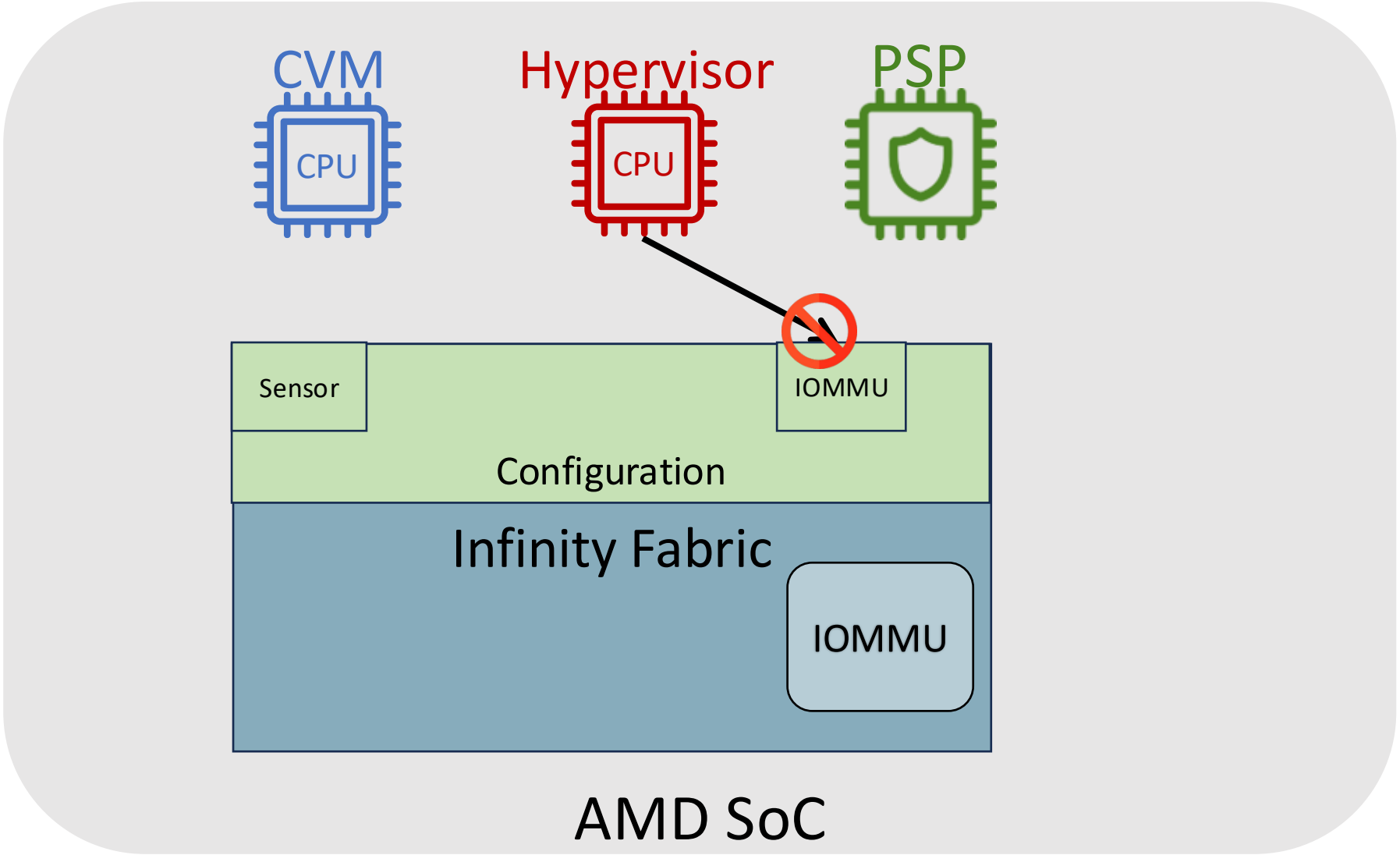
# Infinity Fabric Configuration: by hypervisor

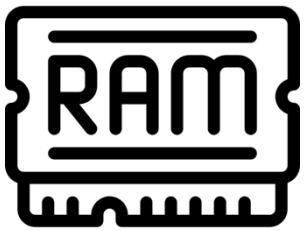


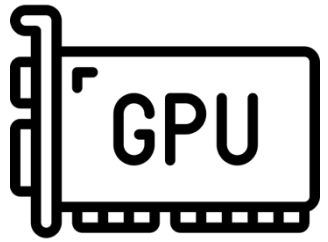



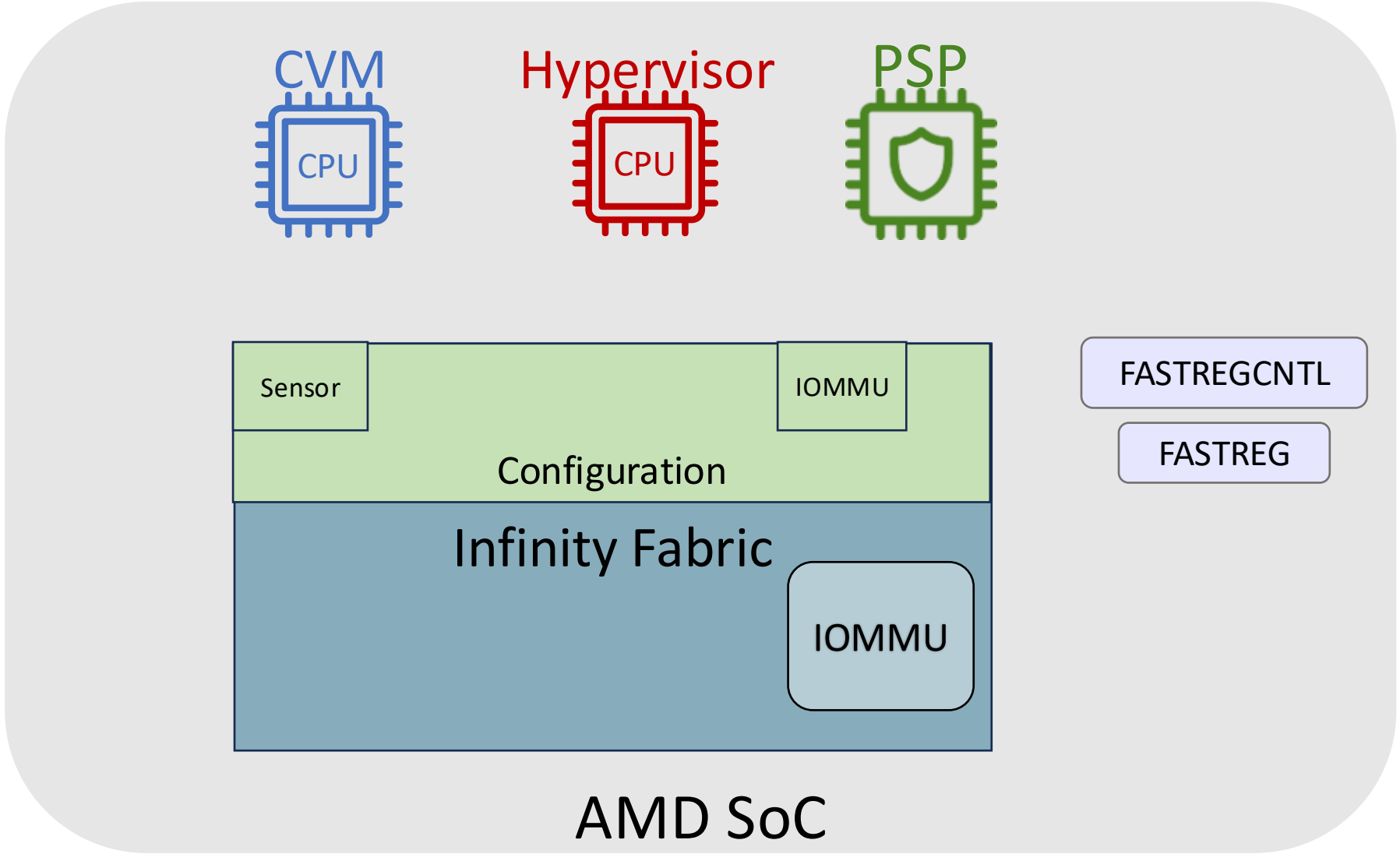
# Infinity Fabric Configuration: Access Control

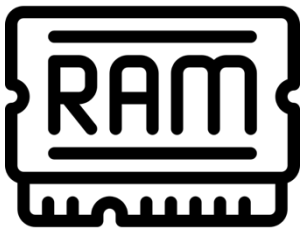


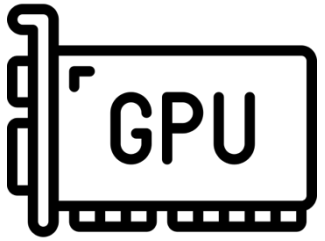



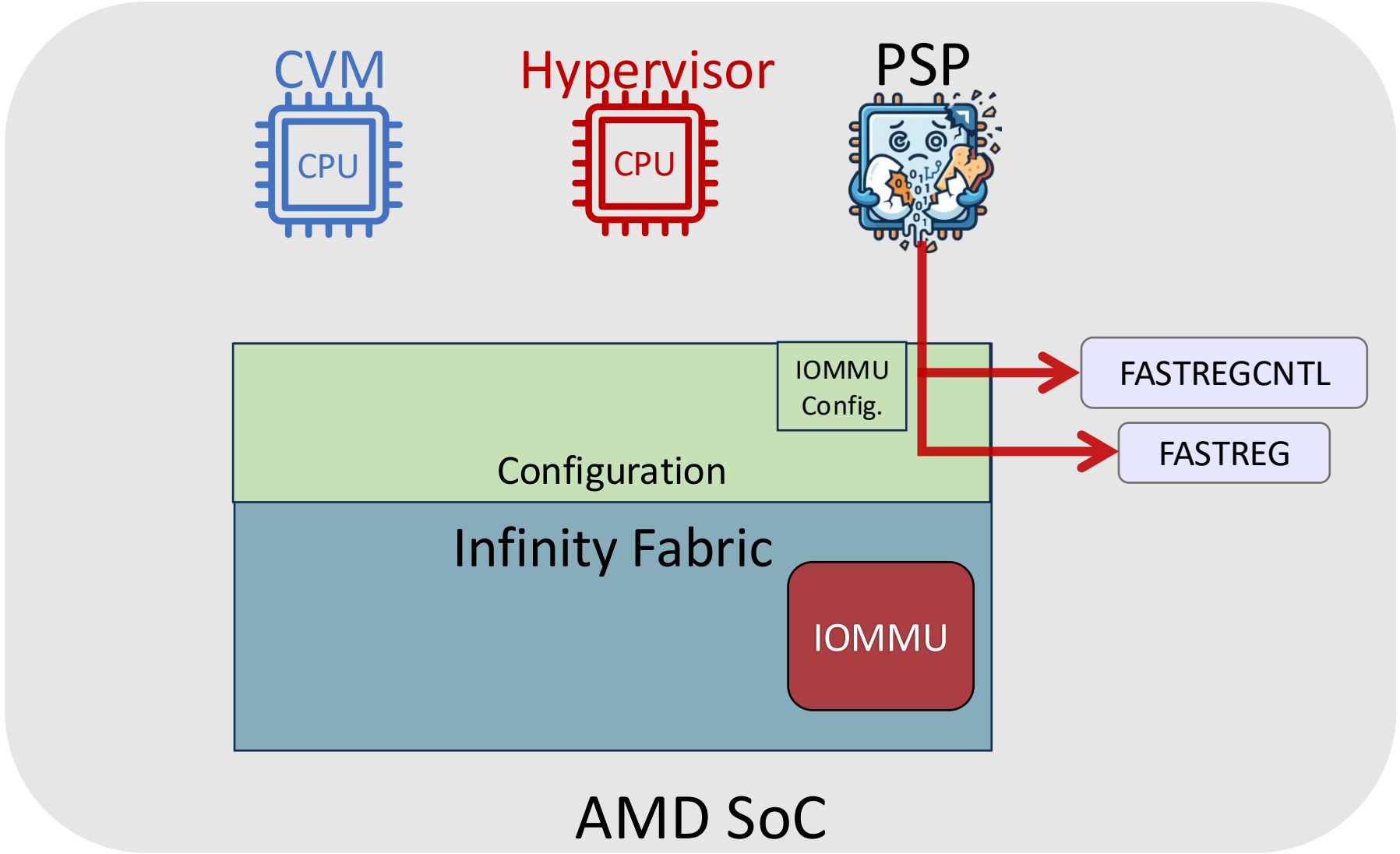
# Other components on the AMD SoC

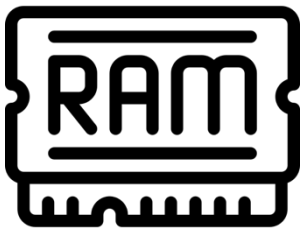


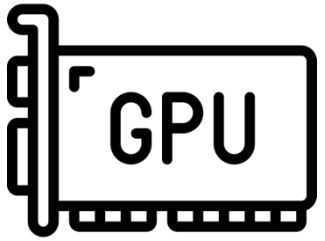



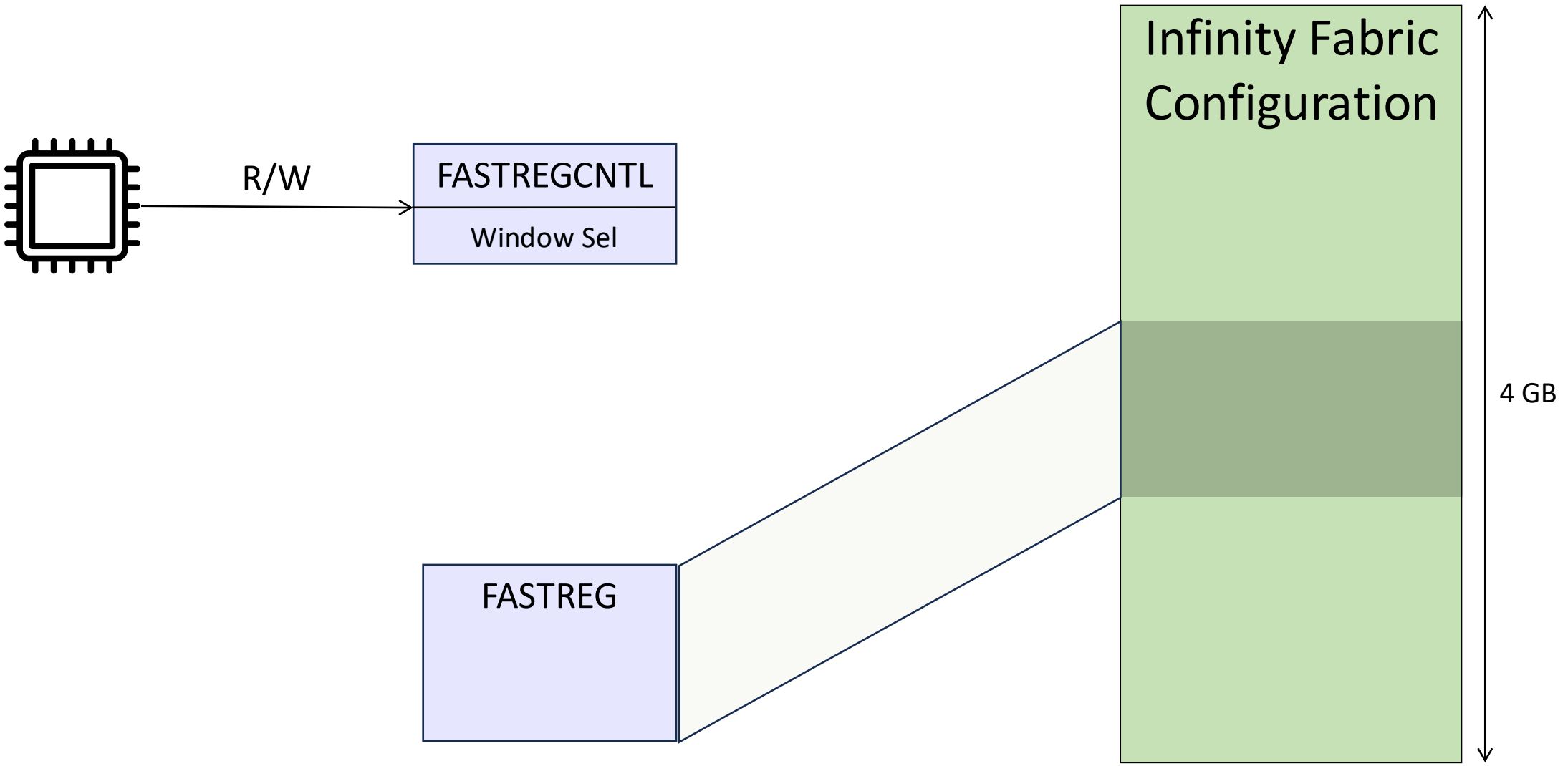
# Confusing the PSP to write attacker's data



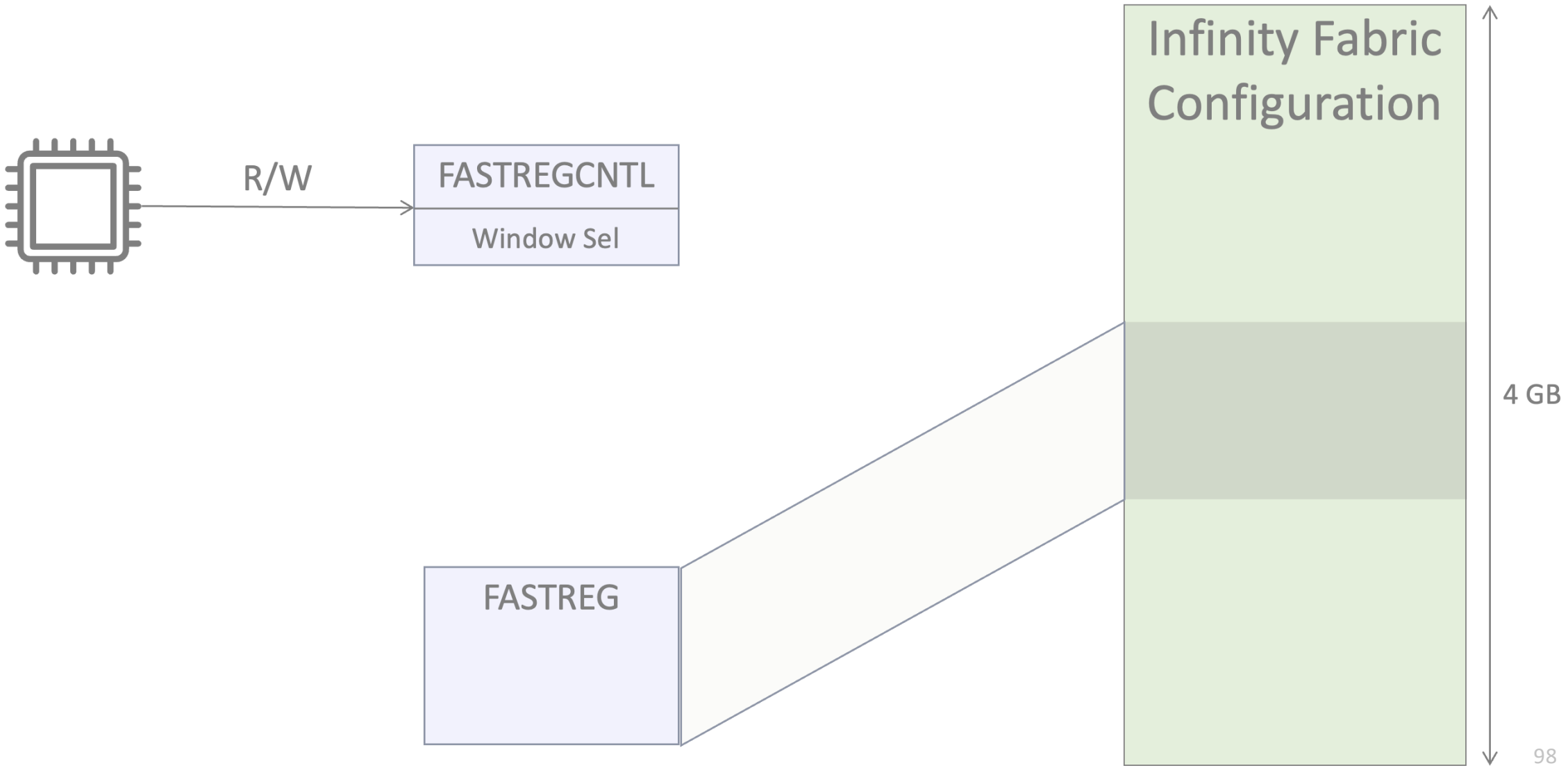



# What's FASTREGCNTL & FASTREG?

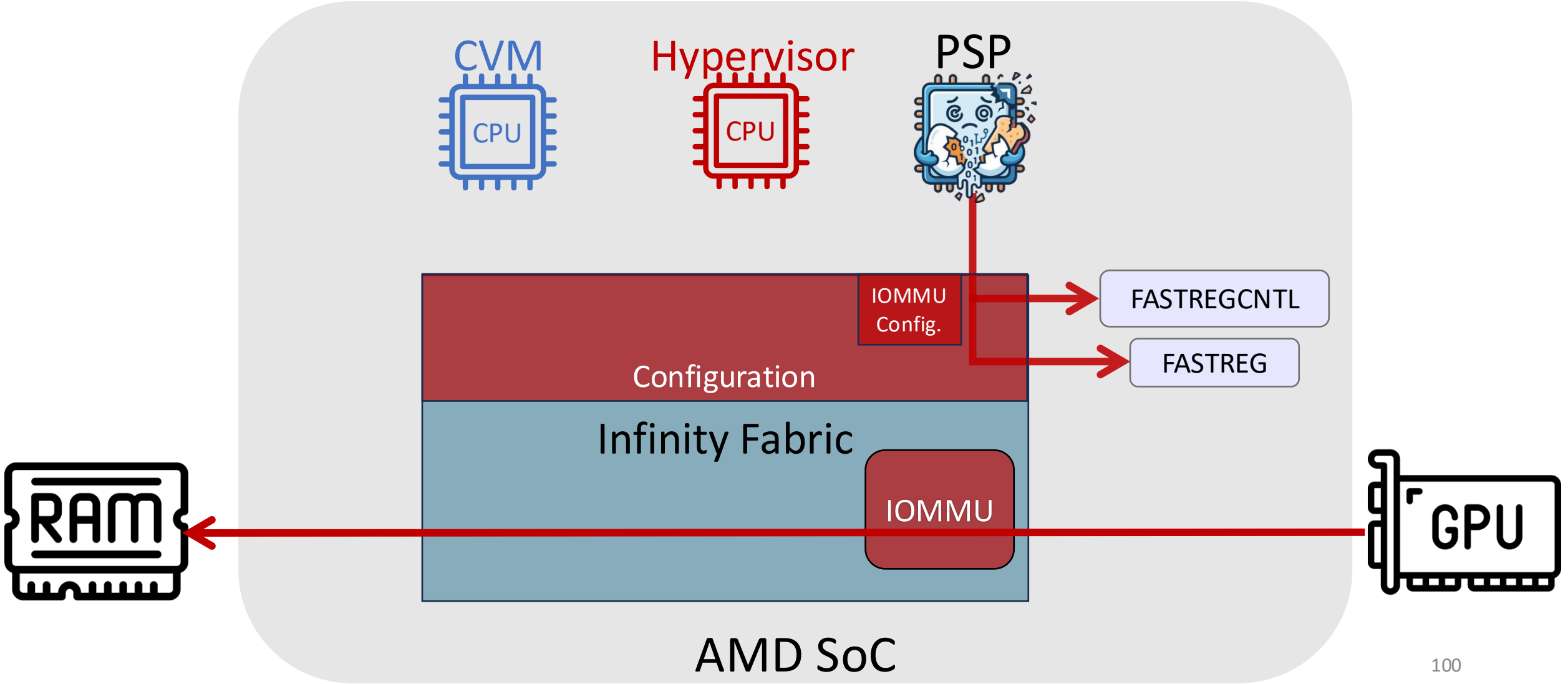


# FASTREGCNTL adjusts the window
# FASTREG indexes into the window



# After reverse engineering…
# Arbitrary write to configuration

# Attacker can change the entire control plane



# Interconnect Corruption Attacks (XCA)





## Fabricked

Corrupts the Data Plane

CVE-2025-54510

## BreakFAST

Corrupts the Control Plane

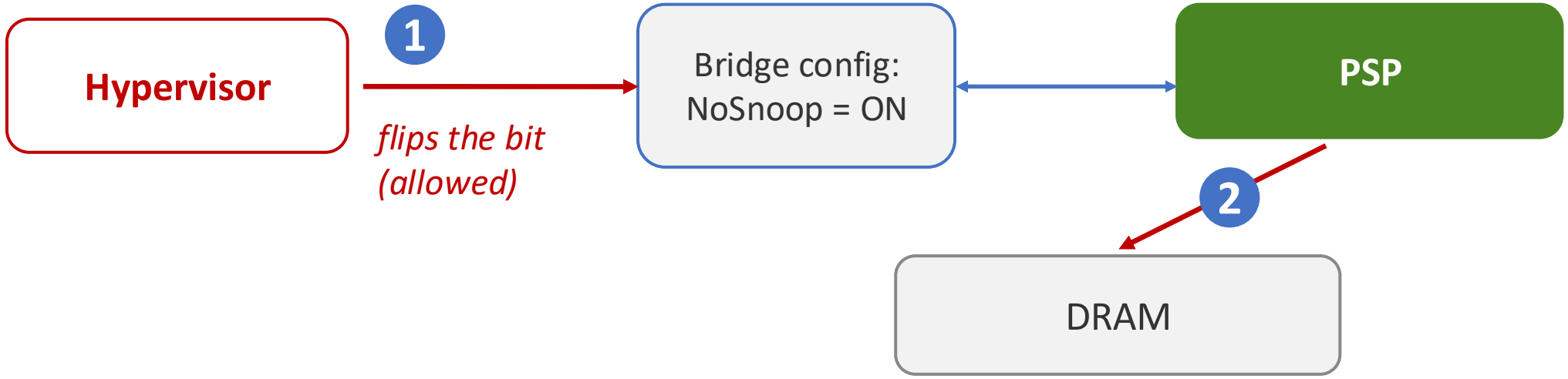
CVE-2025-61971/2025-61972

# XCA #3: Staleus

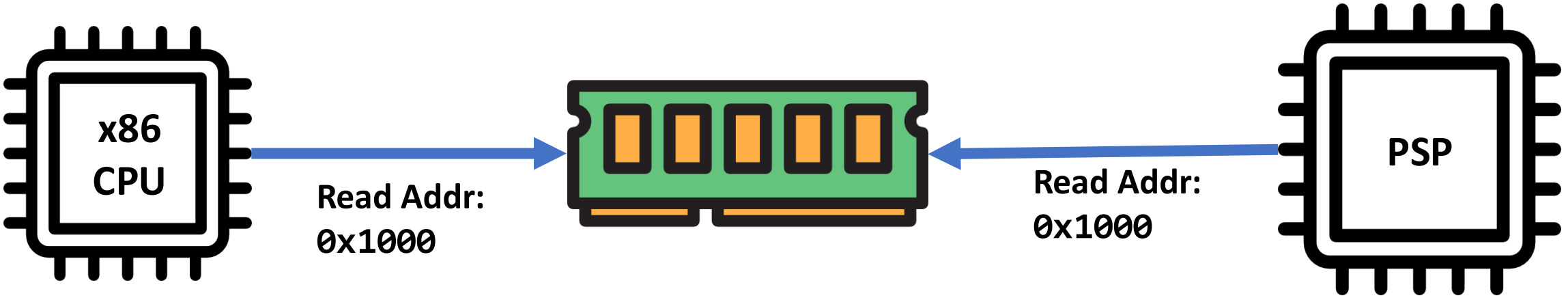
AMD enforces coherence between the x86 cores and the PSP via …

**Infinity Fabric**

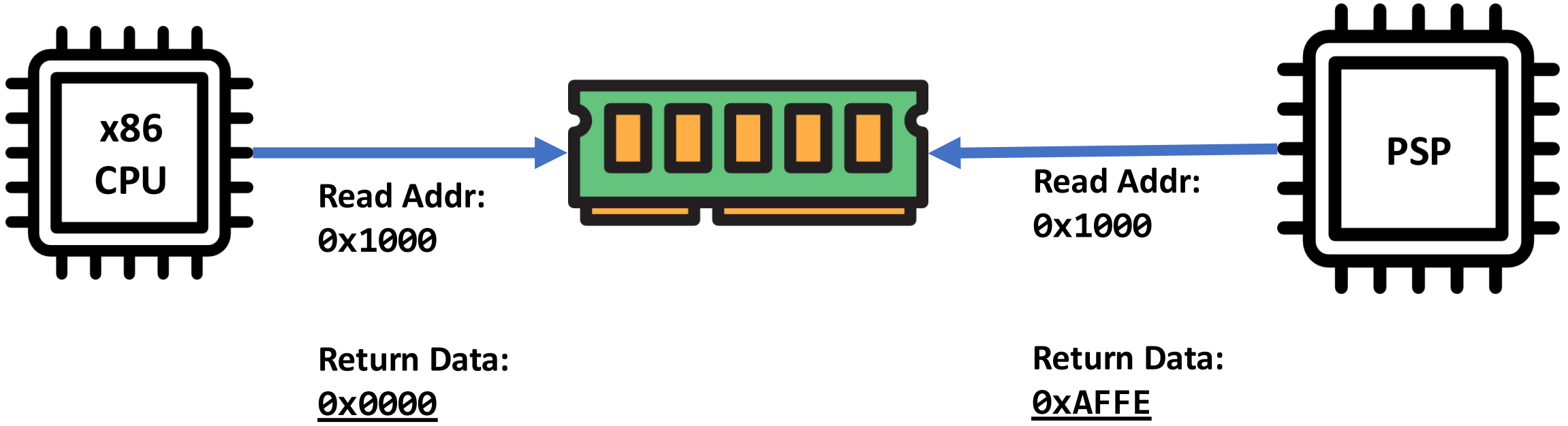
# Staleus: Turns off coherency



# Staleus: Turns off coherency



# Induce Split View Between x86 CPU and PSP



# Using Split View to break SEV-SNP

What can we do with forcing the PSP to be non-coherent?

1. Force the PSP to read stale (already overwritten) data
2. Overwrite data that the PSP assumes is read-only

Then, forge a critical metadata page:

- Gives us arbitrary read and write confidential VM memory
- Breaks **ALL** of SEV-SNP security guarantees
- Cannot be detected by the confidential VM guest

# Interconnect Corruption Attacks (XCA)



## Fabricked

Corrupts the Data Plane

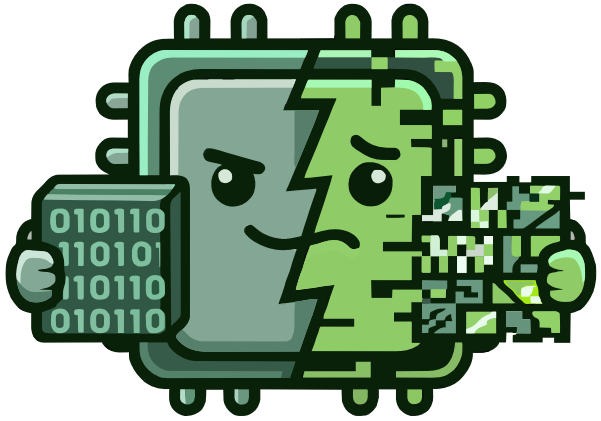
CVE-2025-54510



## BreakFAST

Corrupts the Control Plane

CVE-2025-61971/2025-61972




## Staleus

Corrupts Coherency

CVE-2025-54509

https://xca-attacks.github.io

Systems

# Europe built sovereign clouds to escape US control. Then forgot about the processors

## Intel ME and AMD PSP: The silicon layer nobody certifies

Kim Loohuis

66

Sat 16 May 2026 // 10:30 UTC

AMD faces the same architectural question. On April 14, 2026, researchers demonstrated the Fabricked attack against AMD's SEV-SNP confidential computing technology, achieving a 100 percent success rate with a software-only exploit. The Platform Security Processor proved vulnerable to the same class of compromise.

# How can we detect & patch such vulnerabilities early on?

- Modern SoCs use standardized interconnect protocols

- AXI, OBI, APB, AHB, TileLink
  - Advanced eXtensible Interface (AXI) is an on-chip bus protocol, used in AMD Xilinx, PULP, and open-source projects
  - TileLink is used in Google OpenTitan

- Staleus stems from a potential issue in the AXI-To-SDP Bridge

# Catching bugs before they ship

- Chips are assembled from standard building blocks
  - They talk to each other using shared rules
- When a block breaks those rules,
  - it becomes a security hole.

**eXpect**

Finds where real building blocks don't follow the rules

**135 violations in 7 designs**

**Xray**

Tests real interconnect setups for exploitable vulnerabilities

**41 vulnerabilities in 7 interconnects**

# How can we patch such vulnerabilities early on?

If we can capture and express the expected behavior faithfully,

Then,

#1. we can synthesize the necessary assertions

#2. if there is an assertion violation, we can synthesize a patch

Generalize to multiple protocols, leverage LLMs

# Stitch: Assertion-Guided Patching using LLMs



**Stitch**

Automated Patching of On-chip protocols implementations



Patched 11 implementations across 5 protocols

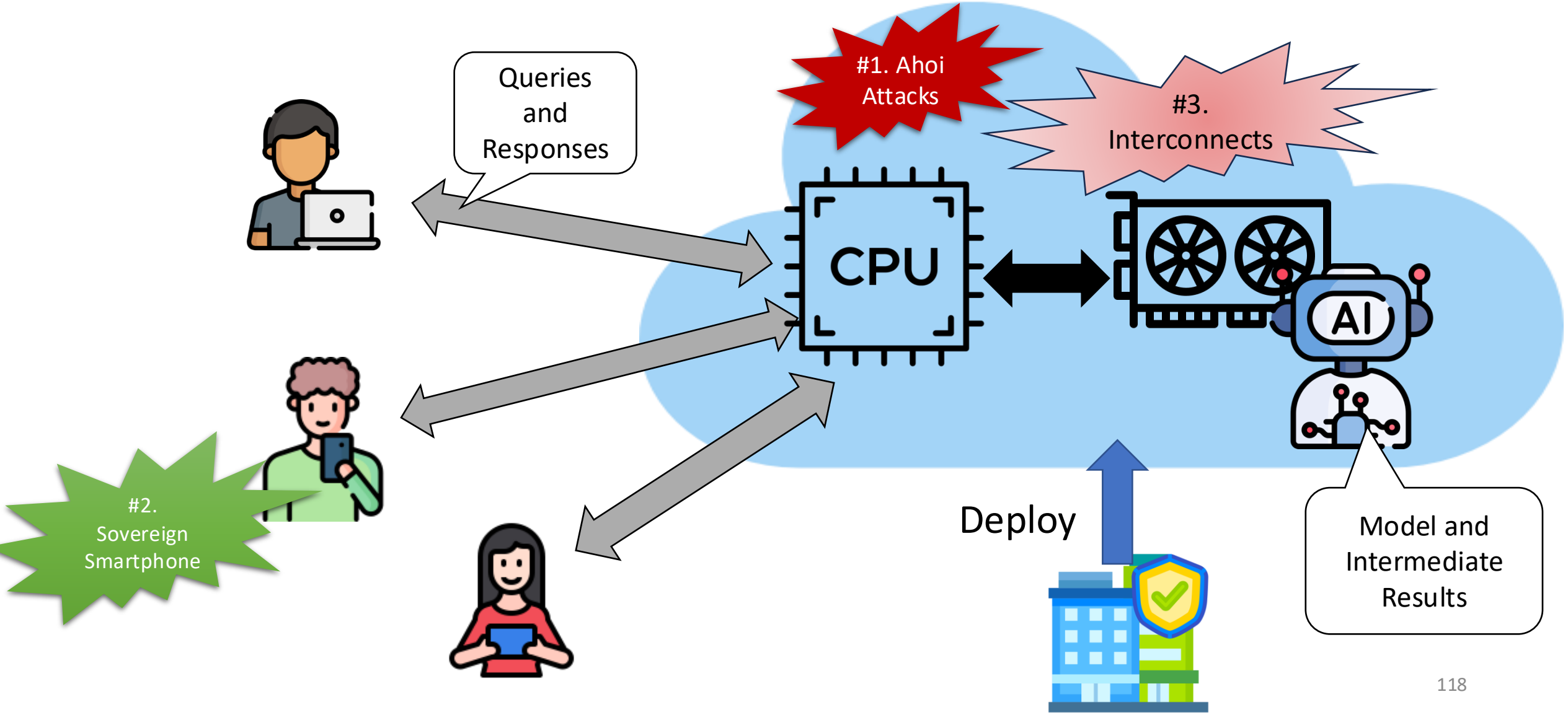


Dataset of 100 violations of on-chip communication protocols

*Capture the expected behavior once, an LLM can synthesize the fix*

xprotocols-lab.github.io/stitch

# Where do we go from here?



# Digital Sovereignty is an ongoing process

It needs both:

**Build systems we can control**

Example: Sovereign Smartphone
from 165M to 10K lines of trusted code

+

**Challenge systems we are asked to trust**

Example: New attacks on Intel and AMD,
Tools that detect and patch chip bugs early on

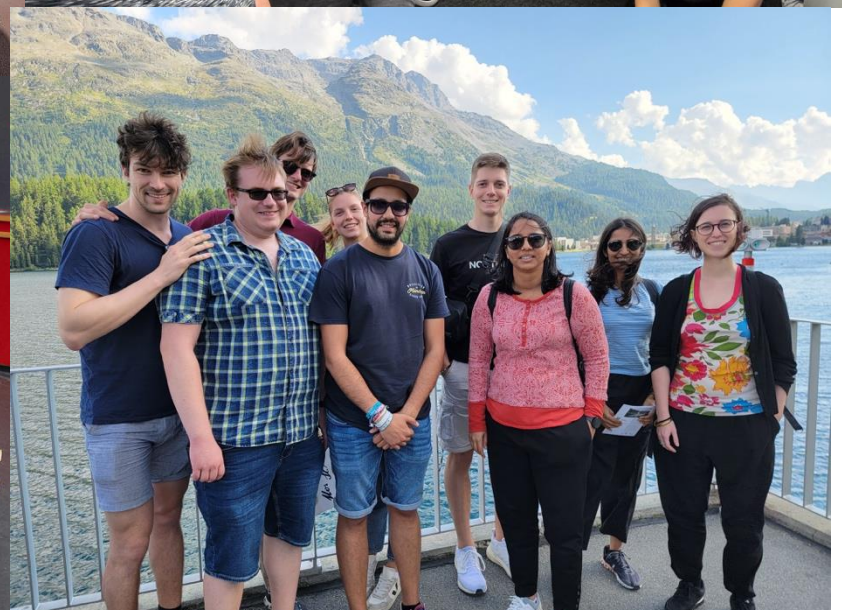
# Thanks to the team behind all this work!



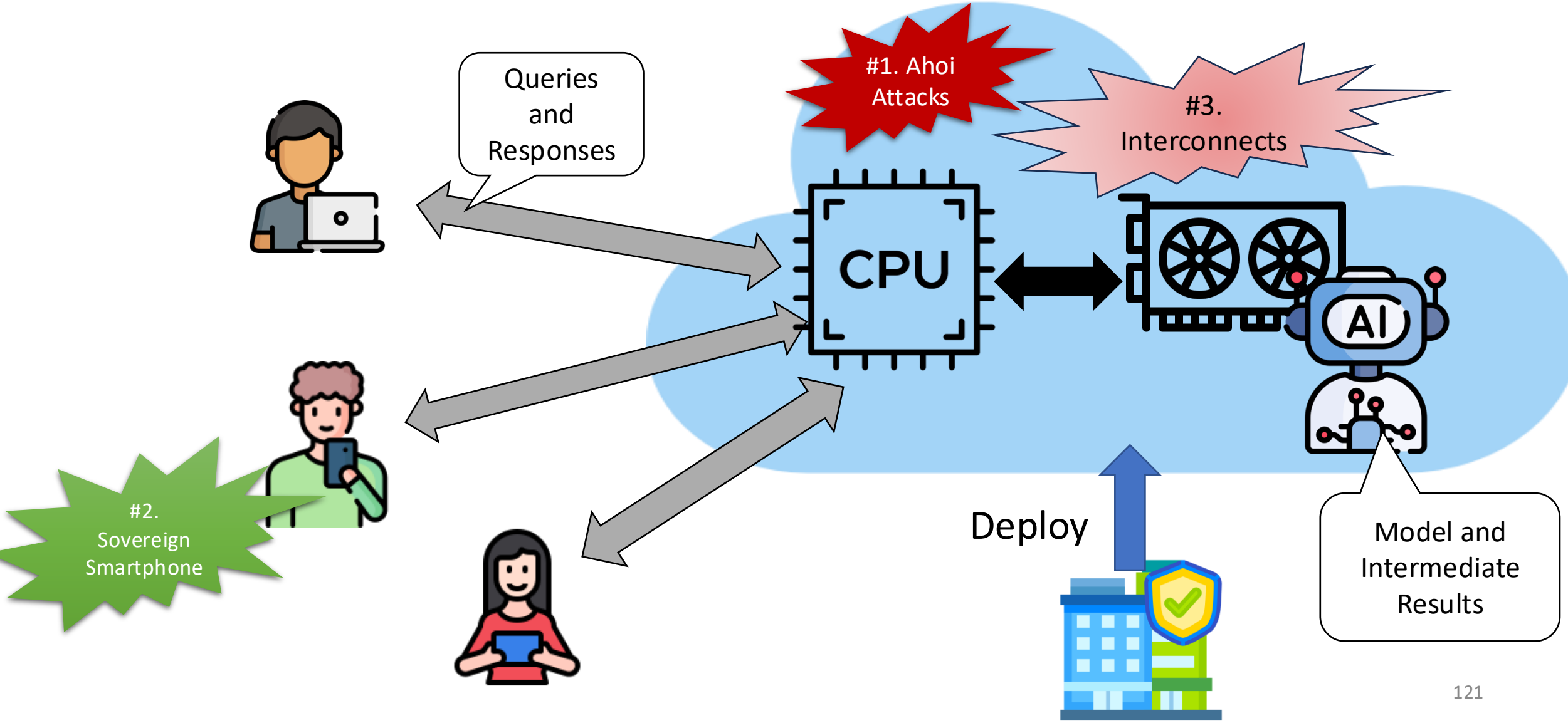






# Summary


Queries and Responses
#1. Ahoi Attacks
#3. Interconnects
CPU
AI
#2. Sovereign Smartphone
Deploy
Model and Intermediate Results

# Thank you!