

Devlore: Device Interrupt Protection for Confidential VMs

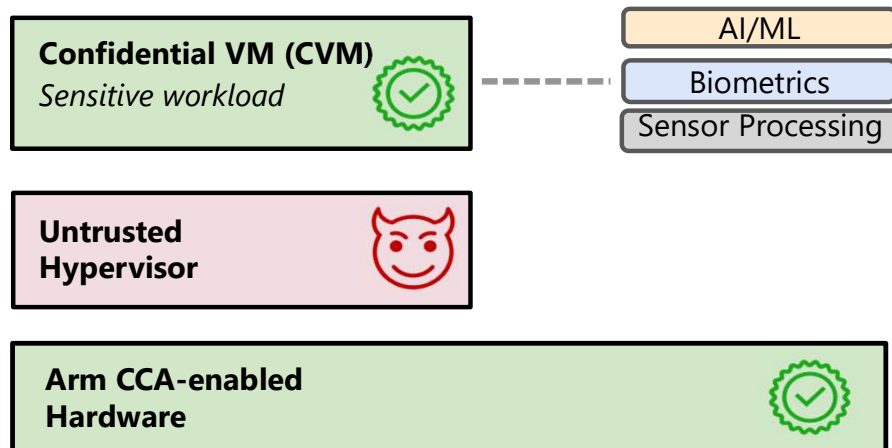
Andrin Bertschi*, Supraja Sridhara*, Mark Kuhne,
Benedict Schlüter, Friederike Groschupp, Clément Thorens,
Nicolas Dutly, Srdjan Capkun, Shweta Shinde

ETH Zurich, 13. October 2026

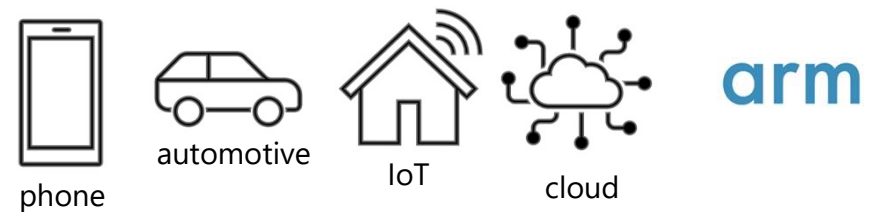
andrin.bertschi@inf.ethz.ch

Confidential Computing

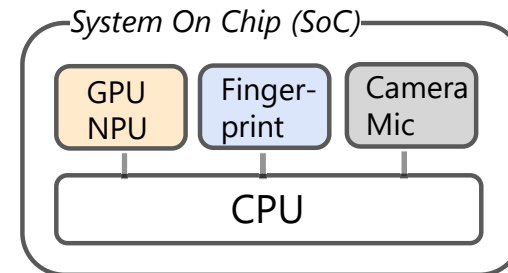
... protects sensitive workloads



... from phones to cloud



... increasingly depend on integrated devices

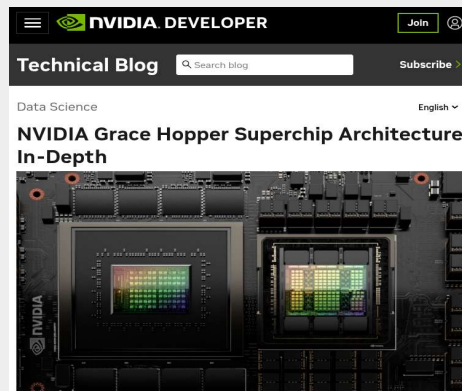


Confidential Computing

Industry:

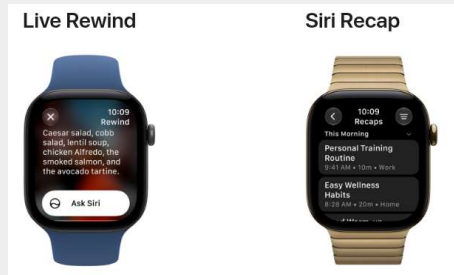
Nvidia Grace-Hopper

integrates CPU + GPU in one superchip



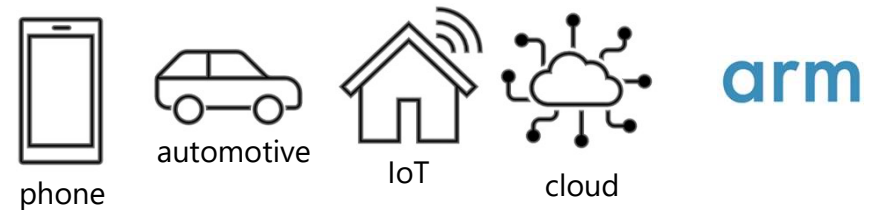
Apple Watch (2026)

Live Rewind/ Siri Recap
Microphone on-device AI

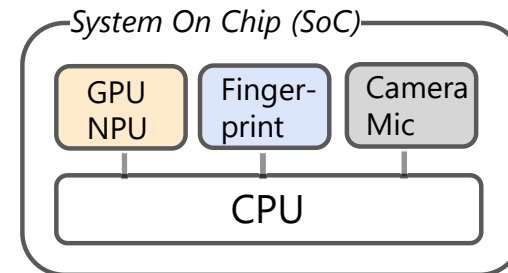


<https://developer.nvidia.com/blog/nvidia-grace-hopper-superchip-architecture-in-depth>
<https://support.apple.com/en-us/148354>

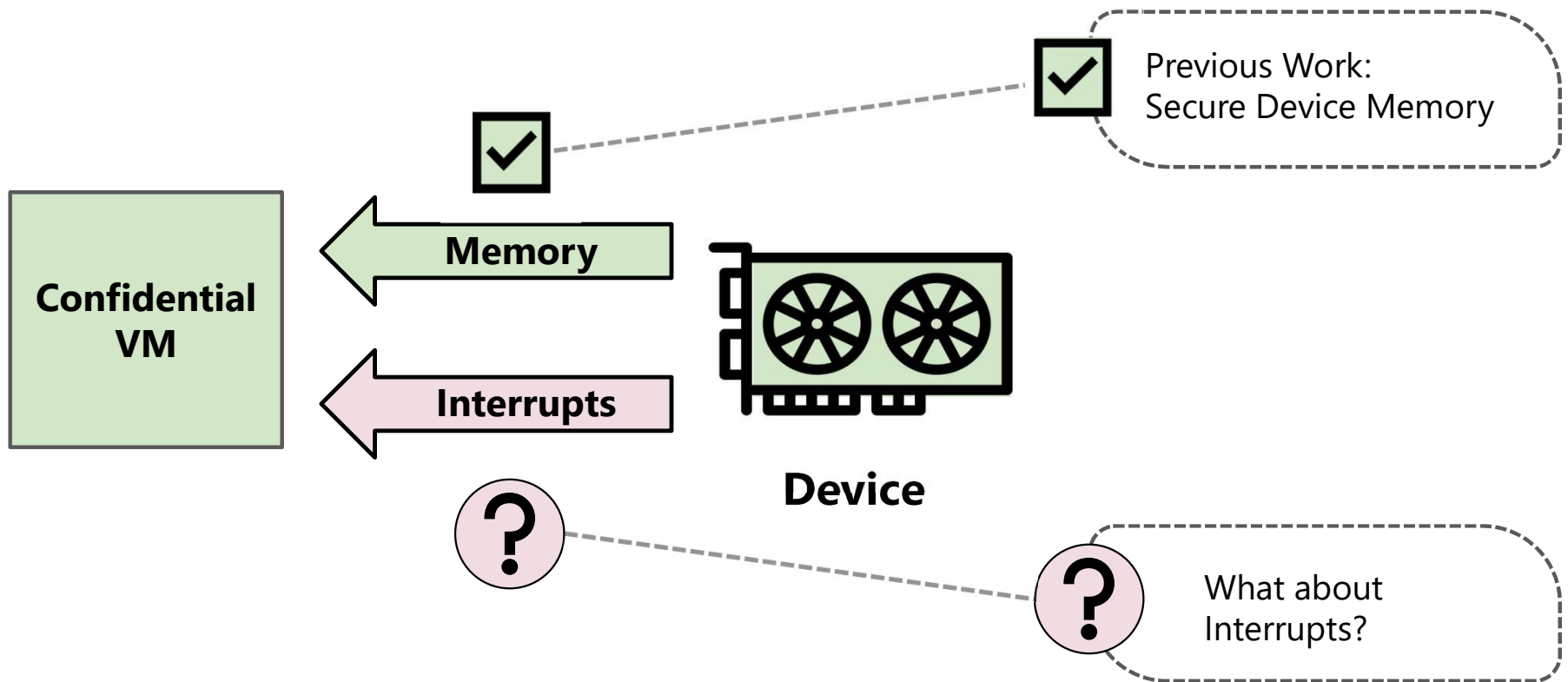
... from phones to cloud



... increasingly depend on integrated devices



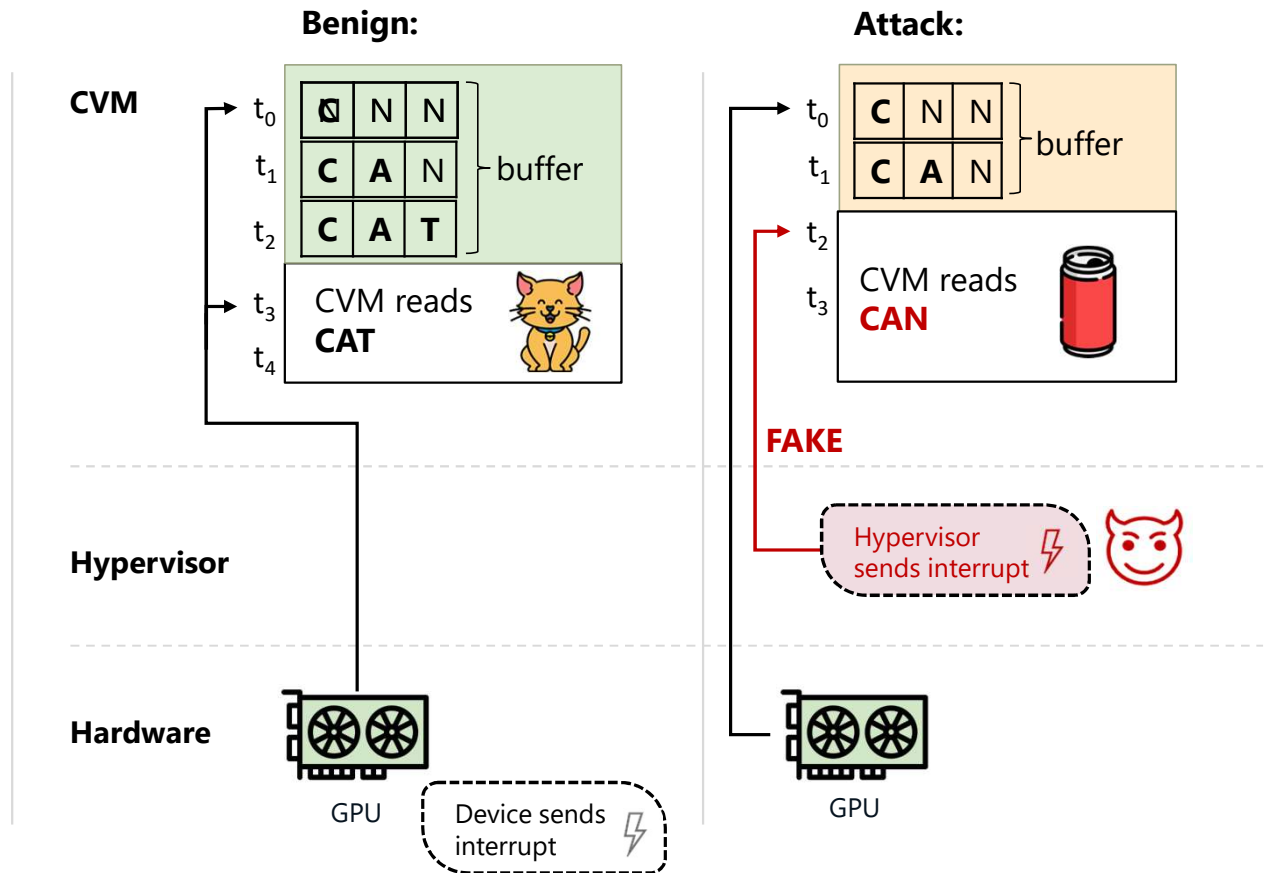
Devices are part of sensitive computation



Can Device Interrupts break Trusted Execution?

Interrupts can Break Trusted Execution

Fake interrupts can break execution **even if memory is protected**



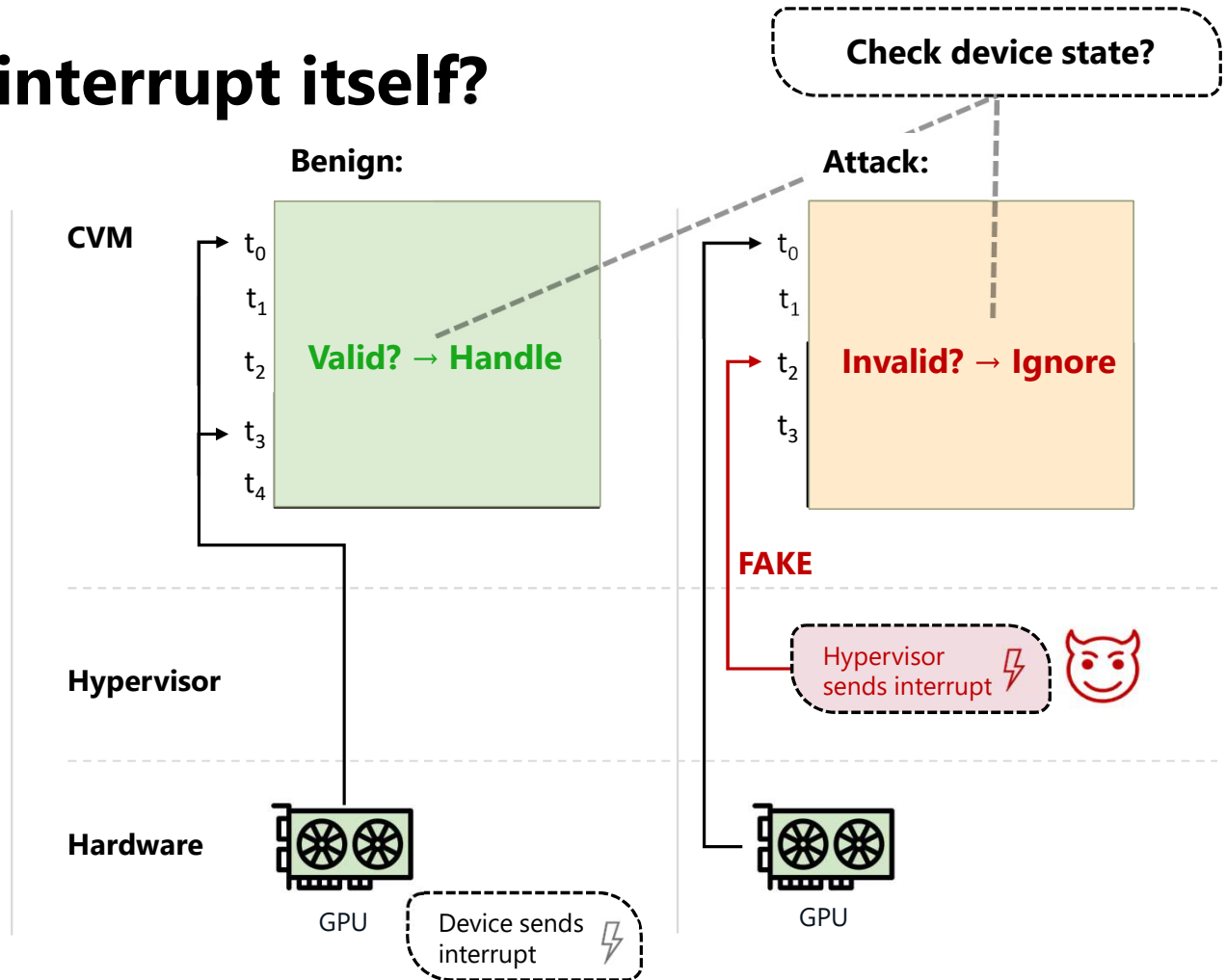
Can driver validate interrupt itself?

Can help:

- **Check** protected device **state**
- **Ignore** fake interrupts

But general solution:

- Needs **device state** (MMIO status)
- Needs **device-specific checks**
- **One unchecked driver is enough**



Devlore:

How can we provide Interrupt Protection as a System Guarantee so that it does not depend on the Driver.

Device Interrupt Protection for CVMs

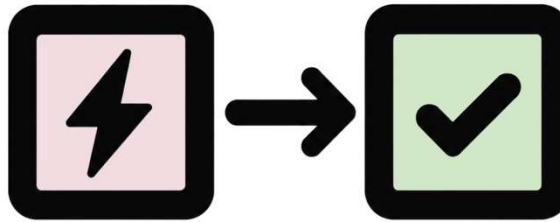
Contributions

Interrupt Isolation



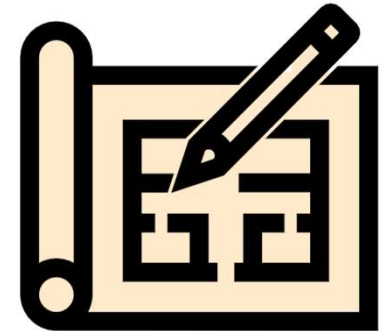
Complete **device isolation** beyond memory

Delegate but Check Design



Hypervisor **manages**
Trusted software **checks**
No driver/app changes

Practical CCA Prototype

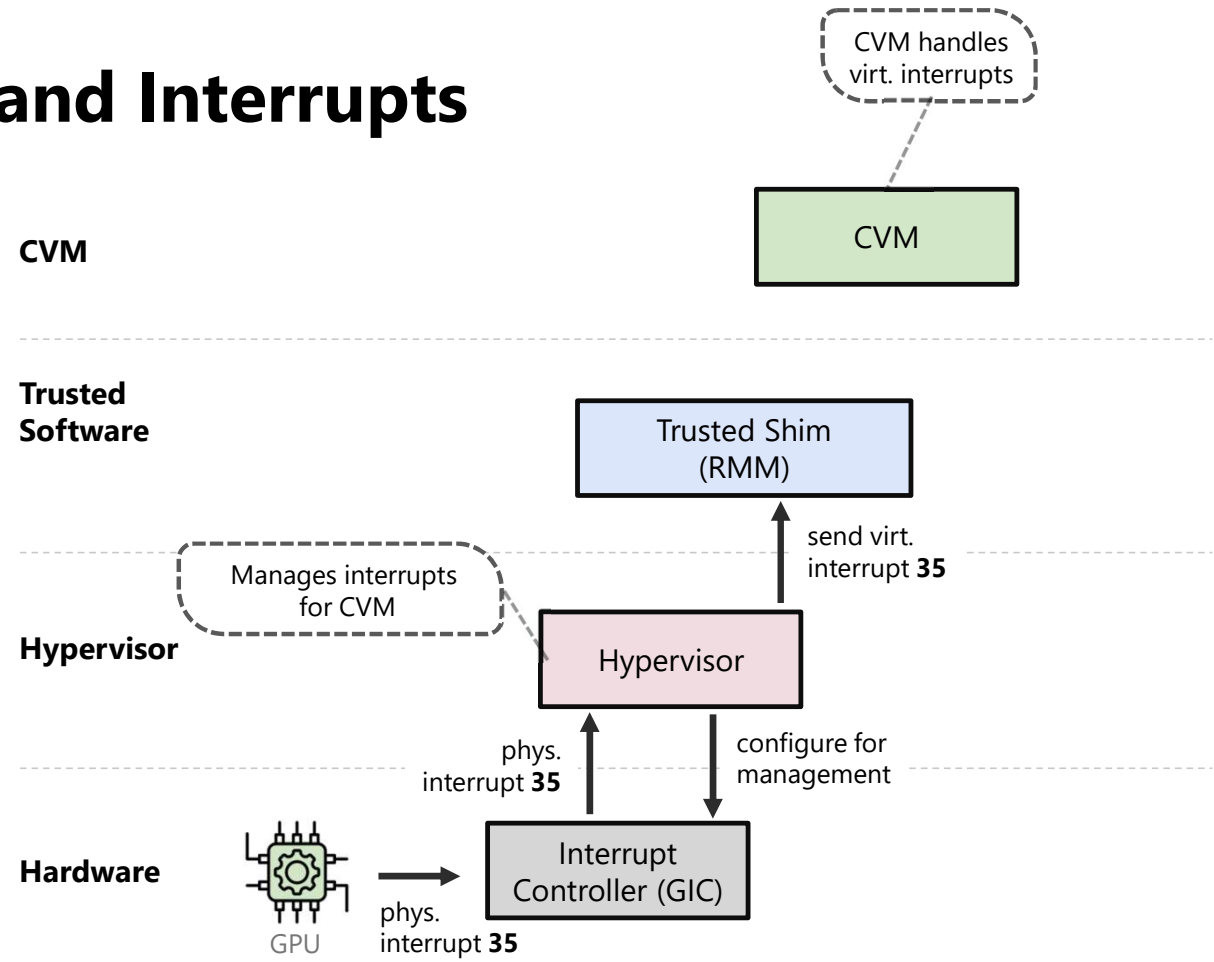


Multiple **devices**
Minimal overhead

Background: Arm CCA and Interrupts

Interrupt Flow:

- **Device raises phy. interrupt**
- Hypervisor manages interrupts
- **CVM receives virt. interrupt**



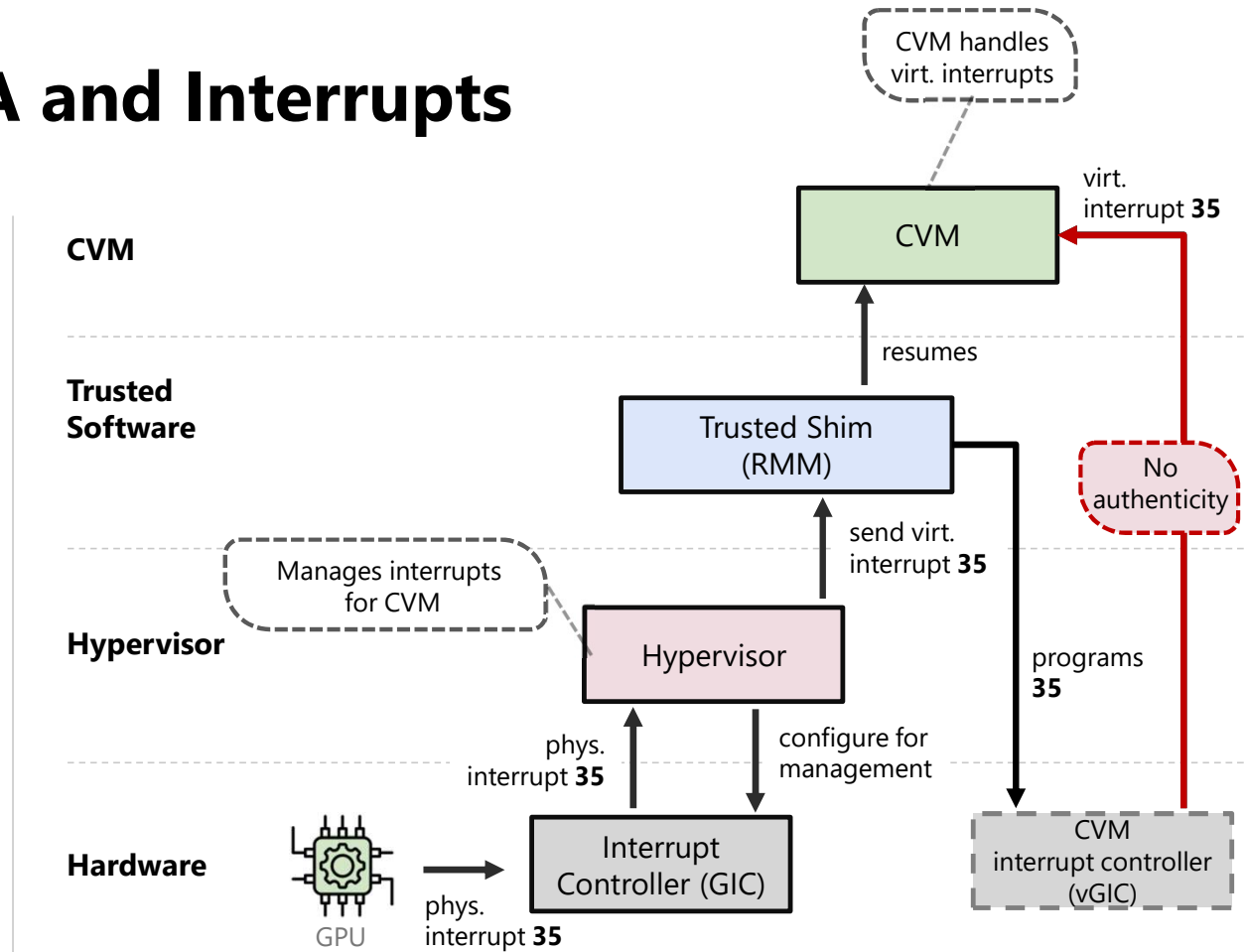
Background: Arm CCA and Interrupts

Interrupt Flow:

- **Device raises phy. interrupt**
- Hypervisor manages interrupts
- **CVM receives virt. interrupt**
- RMM programs CVM vGIC

Security Gap:

RMM cannot verify interrupt authenticity

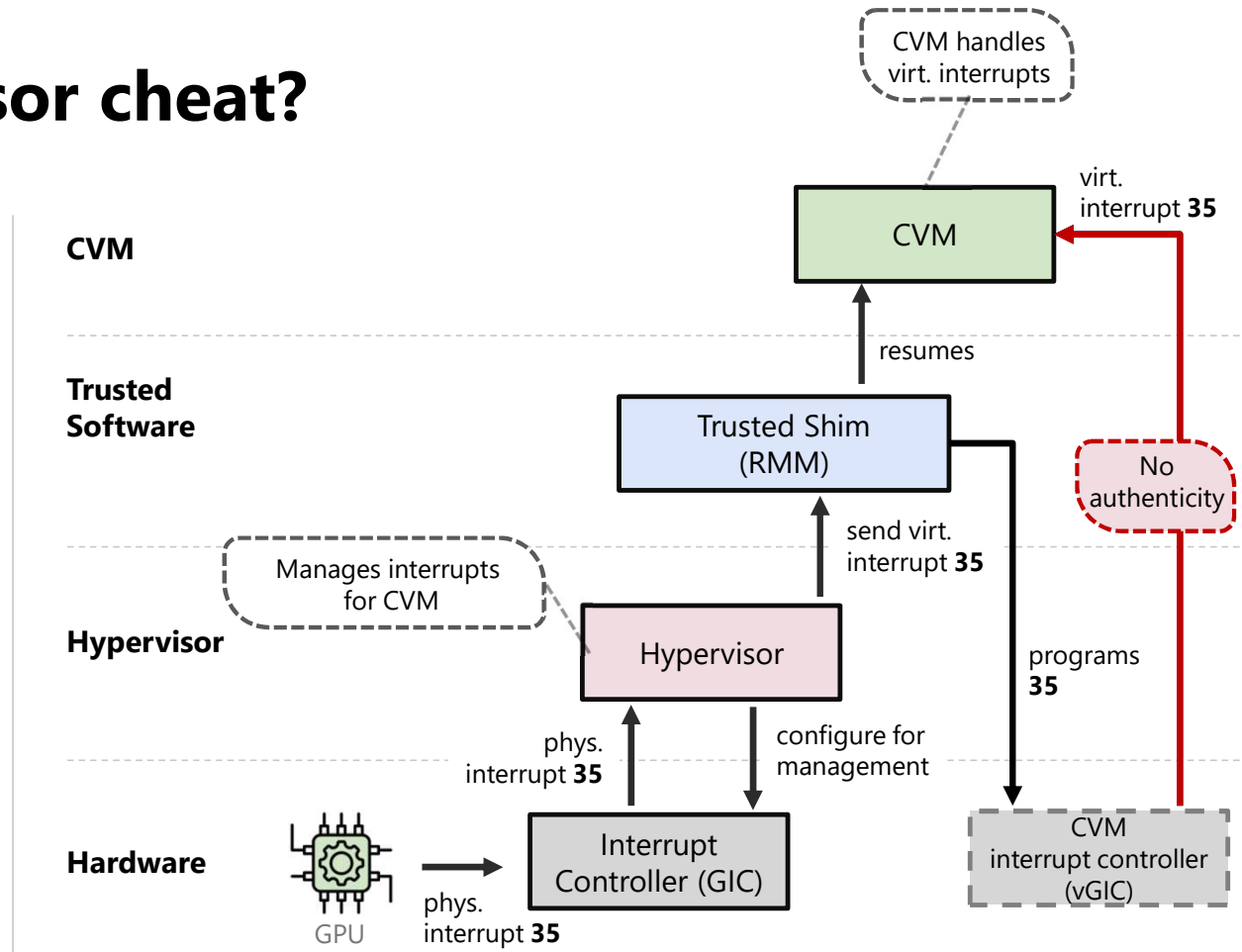


How can the Hypervisor cheat?

A malicious hypervisor can:

Security Gap:

RMM cannot verify interrupt authenticity



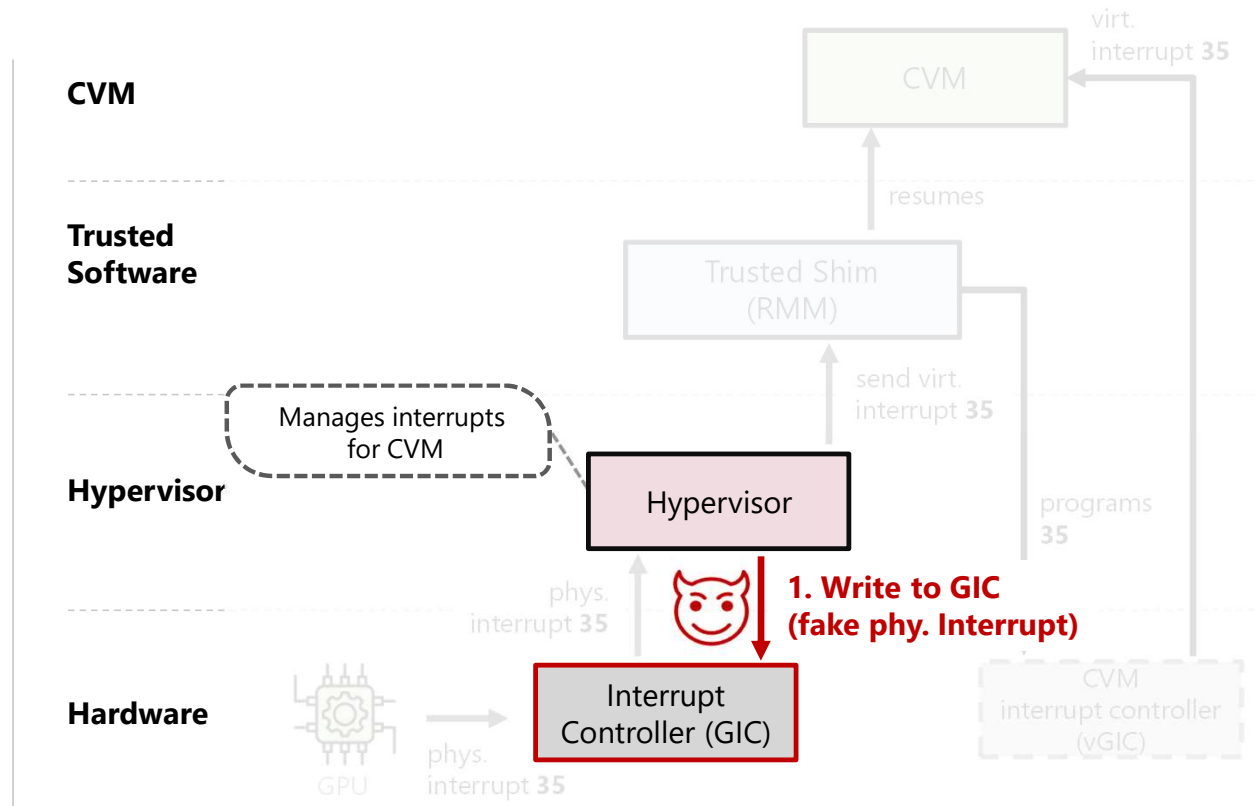
How can the Hypervisor cheat?

A malicious hypervisor can:

- **1. Fake phy. interrupt by writing to GIC**

Security Gap:

RMM cannot verify interrupt authenticity



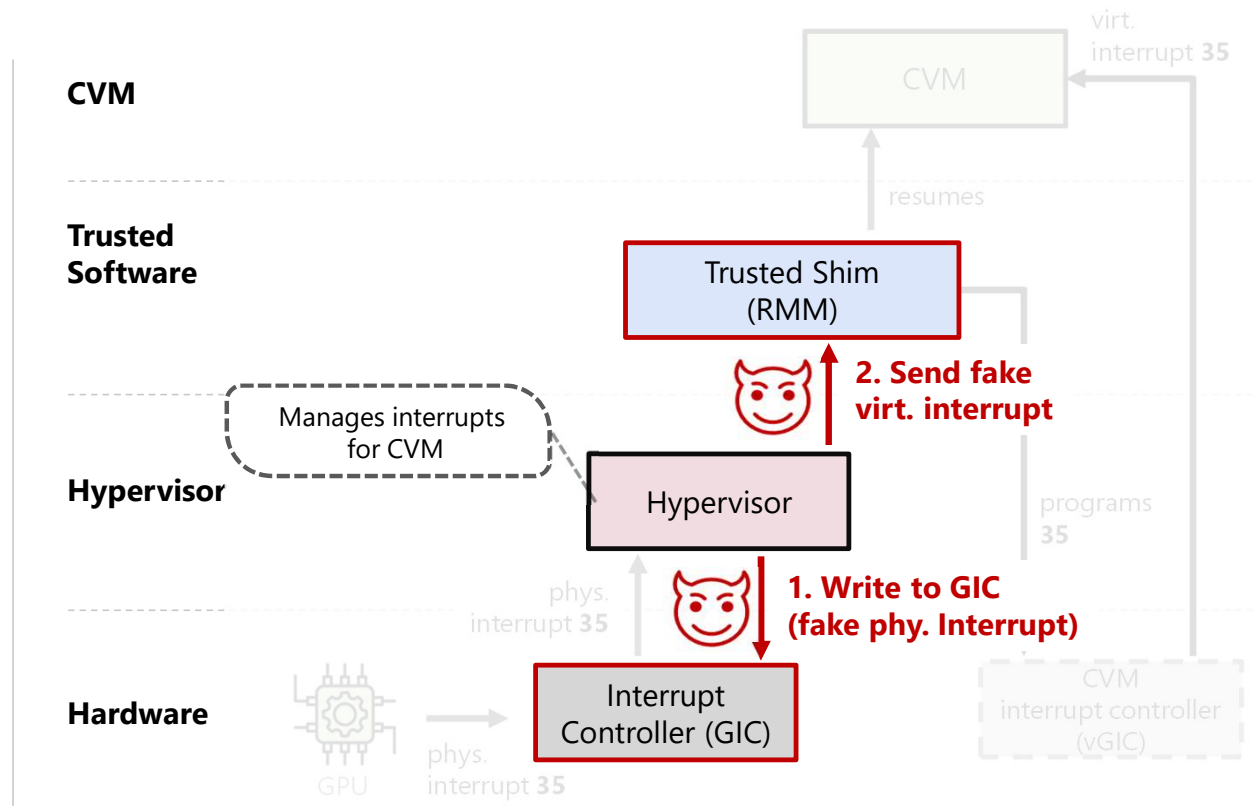
How can the Hypervisor cheat?

A malicious hypervisor can:

- **1. Fake phy. interrupt by writing to GIC**
- **2. Fake virt. interrupt**

Security Gap:

RMM cannot verify interrupt authenticity



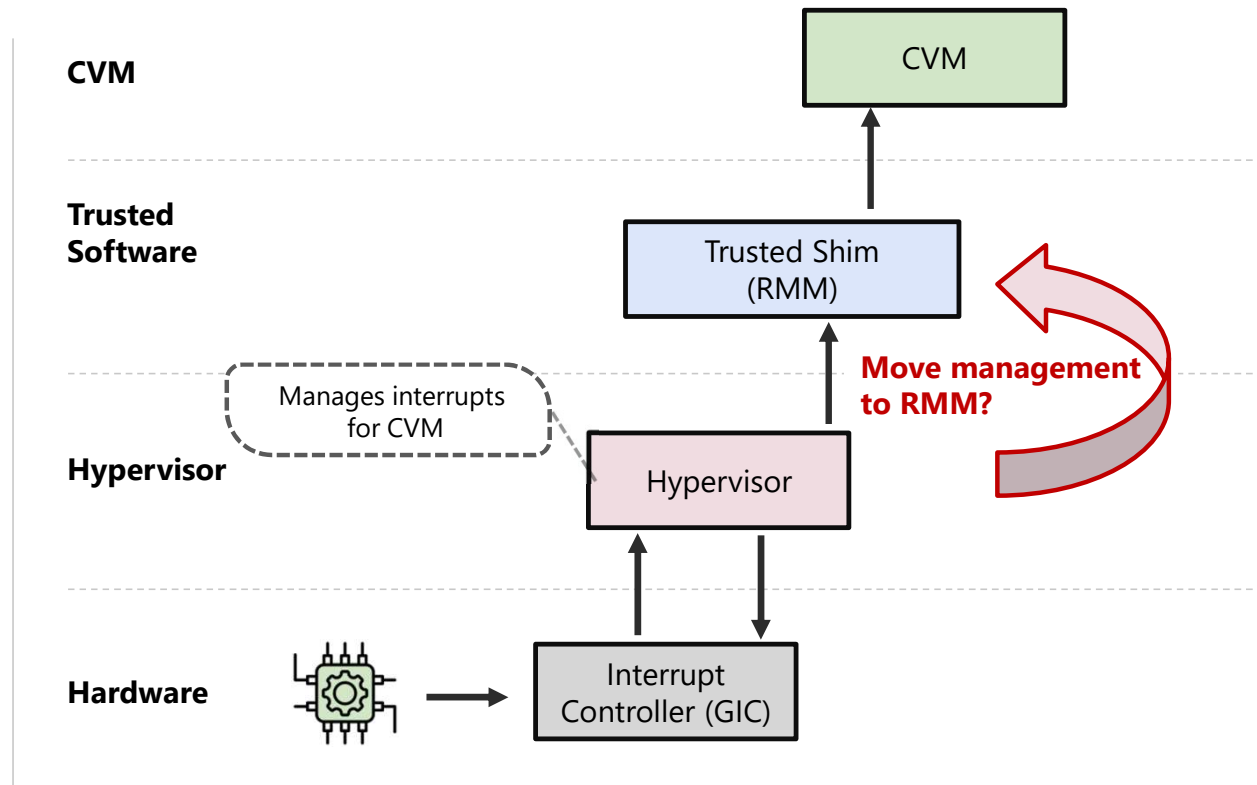
Why not move Interrupt Management to RMM?

Obvious solutions?

Interrupt management:

- depends on scheduling
- Timers/ IPIs
- Priorities
- vGIC management

**Large increase in TCB,
Breaks compatibility**

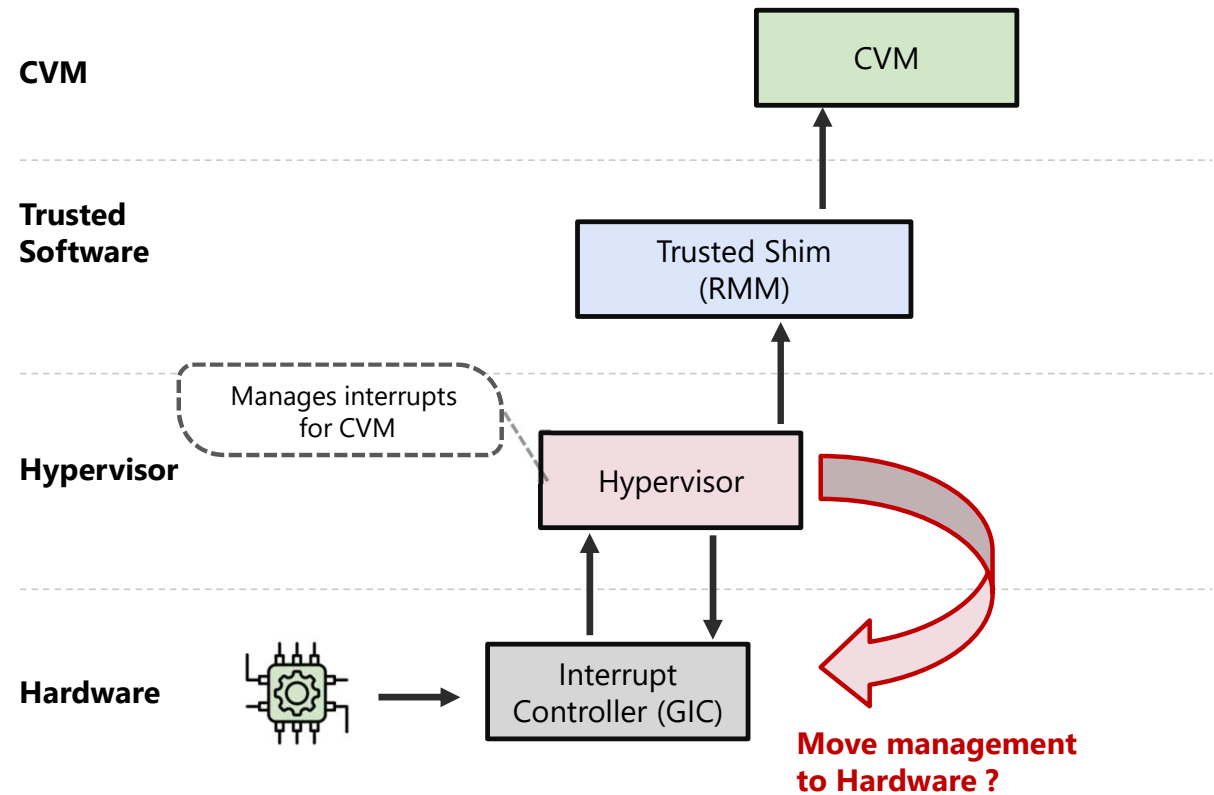


Why not move Interrupt Management to Hardware?

Obvious solutions?

- Requires hardware changes

We don't have access to Arm GIC IP



Devlore:

Can we keep existing hardware and hypervisor management, but verify correctness?

Devlore: Delegate but Check

Compatibility:

Hypervisor keeps managing interrupts



No Hardware changes

Works with standard CCA hardware



How does the RMM distinguish?

We need to establish **interrupt authenticity**

CVM

Trusted Software

Hypervisor

Hardware

Trusted Shim (RMM)

Hypervisor

Interrupt Controller (GIC)

No authenticity

CVM virtual interrupt controller (vGIC)

Devlore: Protect, Record, Check

Goal:

Tie physical interrupt to virt. interrupt

1. Protect:

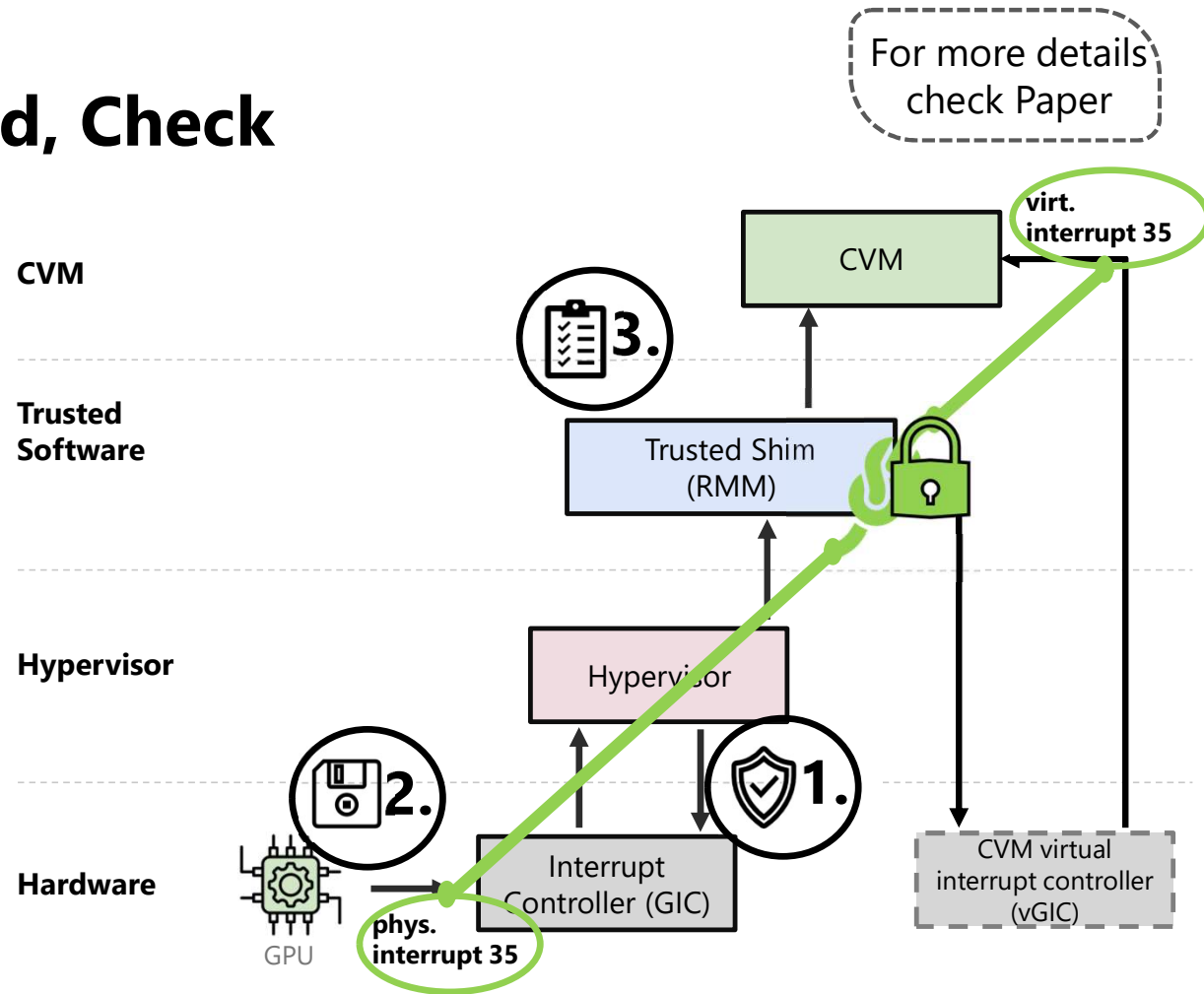
Prevent hypervisor from forging protected interrupt by **protecting GIC**

2. Record:

Record phy. interrupts as **ground truth**

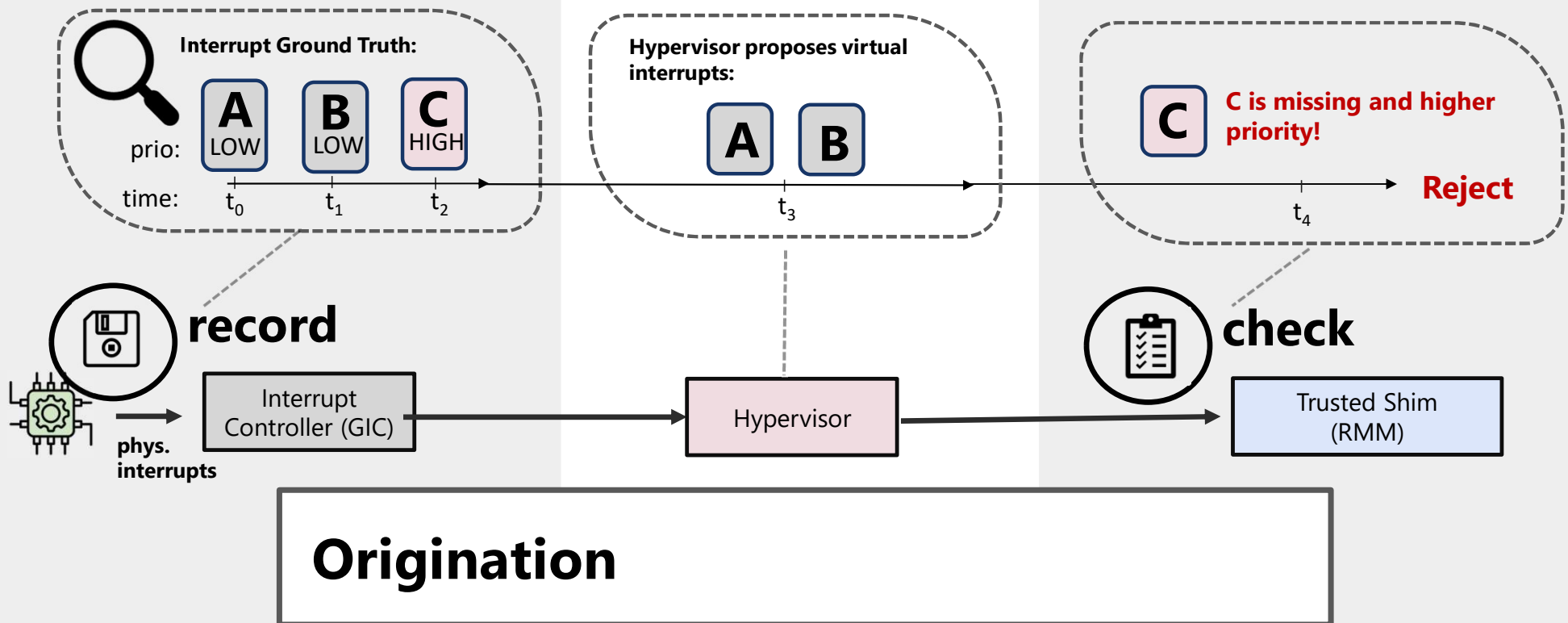
3. Check:

Only deliver virt. interrupt if it matches ground truth



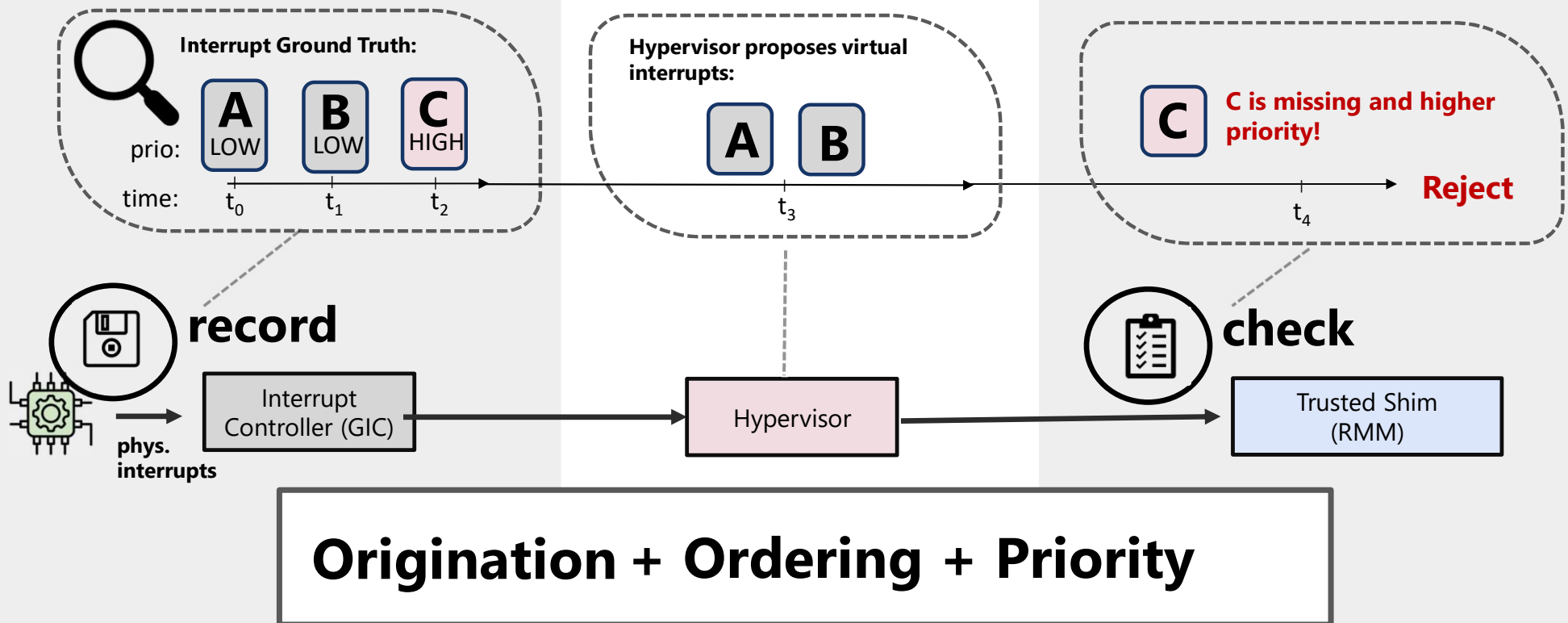
Deep dive: What does the RMM Check?

Challenge: hypervisor may legitimately delay, batch, and prioritize interrupts !



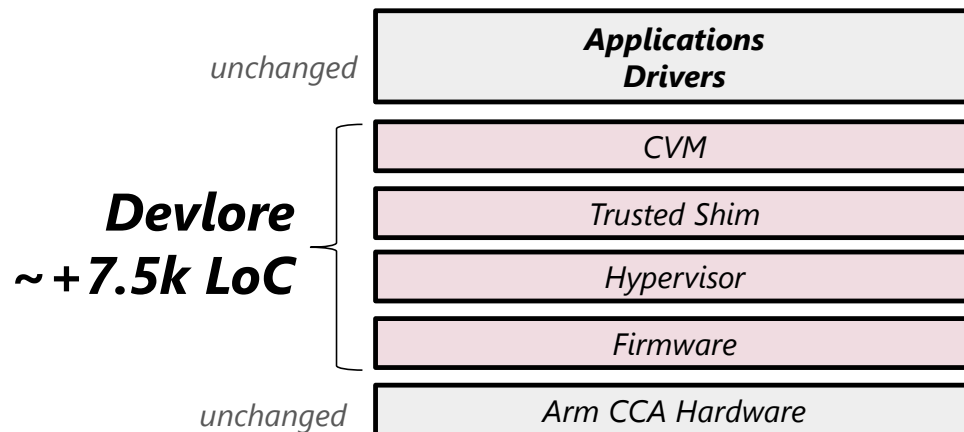
Deep dive: What does the RMM Check?

Challenge: hypervisor may legitimately delay, batch, and prioritize interrupts !



Details: End-to-End Prototype

Implementation:



- Device **assignment from prior work** (memory isolation)
- Devlore interrupt protection on top

Evaluation Platforms:

1. Functionality Prototype

- Security + compatibility evaluation

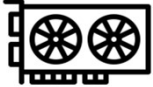
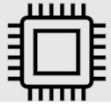
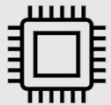


2. Performance Prototype

- Performance evaluation with real devices

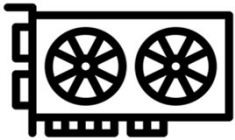


Evaluation: Compatibility with Devices

Device		Setup and Benchmark
Functionality: 4 simulated device classes	Keyboard/ Mouse 	Load amba-kmi keyboard driver Type n keys
	LED/ Buttons 	Flash LED, read button state
	GPU (nomali G76) 	Load kbase driver Run Midgard test suite
	UART 	Load amba-pl011 driver Run nano, browser, IRC chat
Performance: 2 real devices	GPU (Mali G610) 	Load panthor driver Run GPU rendering tasks (glmark2)
	UART 	Load Synopsys DesignWare driver, Fire n keys (loopback)

Unmodified Drivers

Evaluation: What does Interrupt Isolation cost?



Mali G610
GPU

Single GPU
Interrupt

+ 5%

End to End
interrupt latency

Sustained
interrupt load

< 1%

Sustained interrupts
on GPU/UART

Real application
impact

0.06%

Geomean slowdown
glmark2 testsuite

Evaluation on performance prototype **indicates small impact on real workloads**



Devlore: Device Interrupt Protection for Confidential VMs

- Close interrupt gap for confidential devices
- Architectural/system guarantee for interrupt authenticity
- No changes to drivers/apps
- End-to-End prototype for Arm CCA



Paper & Source

<https://sectrs-devlore.github.io>

andrin.bertschi@inf.ethz.ch
<https://abertschi.ch>