Linaro and Arm CCA Tech Event

Deep dive into the Arm Confidential Compute Architecture

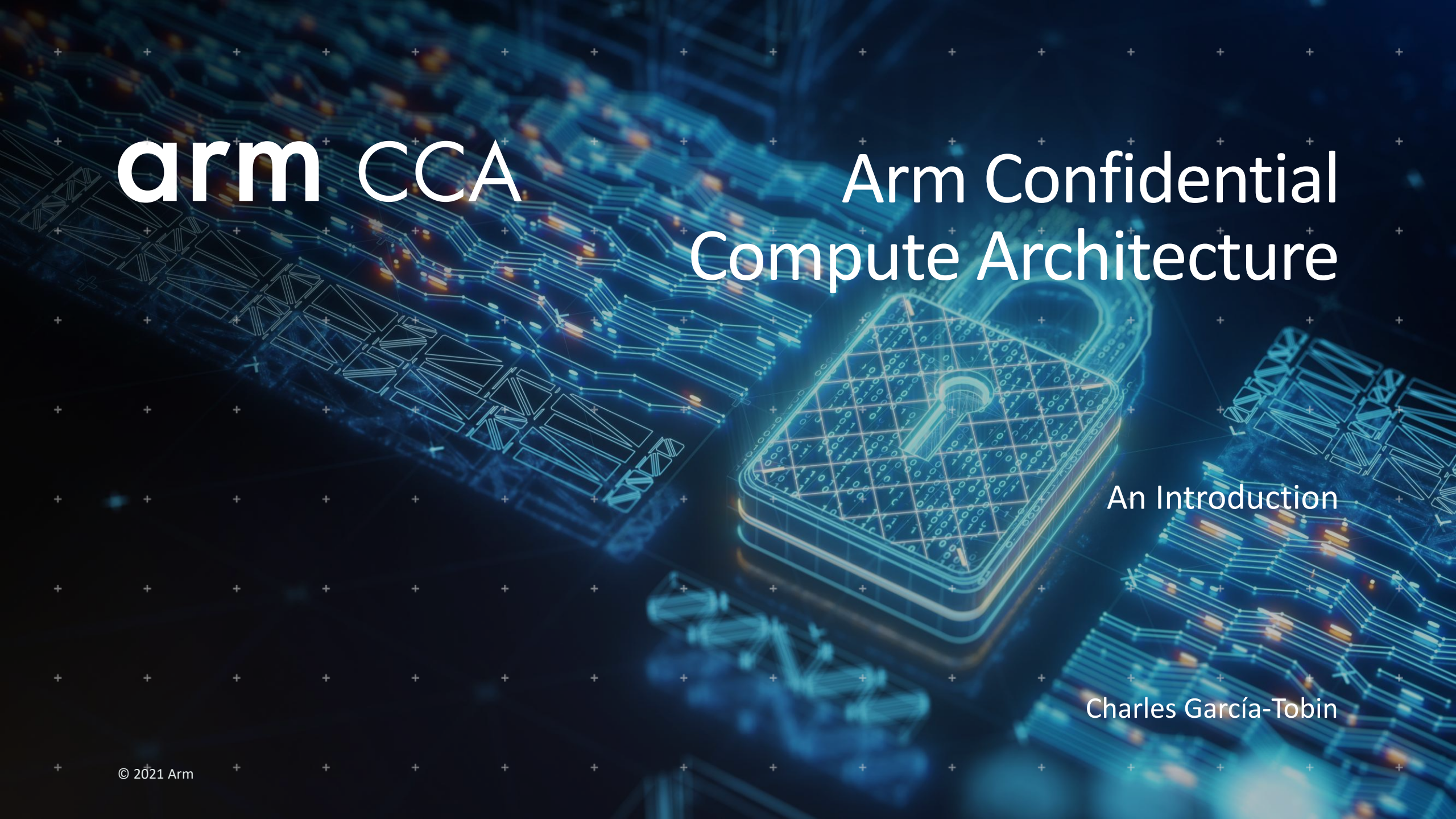
Charles García-Tobin

23 June 2021

Linaro & Arm CCA Tech Event

Agenda

Time (UTC)	Session	
14:00	Charles García-Tobin	Introduction to Arm CCA
14:50		Ten minute break
15:00	Gareth Stockwell	Software & Firmware Architecture
15:40		Five minute break
15:45	Simon Frost	Attestation
16:15		Five minute break
16:20	Matteo Carlini	Developer Resources
16:50		Fifteen minute break
17:05	Joakim Bech	Panel Q&A
17:50		Five minute break
17:55-18:25	Soby Mathew	TF-A Monitor Firmware (deep dive)



Arm Confidential Compute Architecture

An Introduction

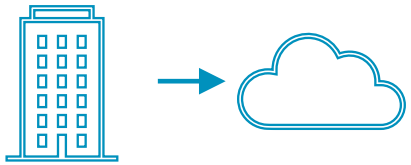
Charles García-Tobin

Strong industry trend towards confidential computing

There are a number of drivers

- Increasing regulation around the use of private data
  - GPDR, CCPA, CRPA, SHIELD, ePrivacy directive, HIPAA…

- Increased use of sensitive personal data and ambient compute e.g ML models



- Desire to move sensitive on-premise workloads to the cloud

“By 2025, 50% of large organizations will adopt privacy-enhancing computation for processing data in untrusted environments and multiparty data analytics use cases.”- Gartner

Hardware-backed Isolation Between All Workloads

Mobile / PC / DTV

Social Media

Email

Browsing

Content Protection

OS Services

Enterprise Apps

Health / Fitness Apps

IoT

Machine Learning

Ambient Compute

Cloud

Tenant 1

Tenant 2

Edge Compute

Machine Learning

Multi-Vendor Environment

Automotive

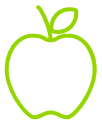
OEM Apps

Personal Data

Autonomous Driving

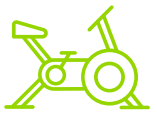
Content Protection

High end Wearables



Health

Smart locks



Fitness

Medical

Proliferation of confidential compute solutions

Redhat
Enarx

Azure Confidential computing

Microsoft
OpenEnclave

Google Asylo
Google OAK

AWS Nitro Enclaves

Baidu MesaTEE

Apache Teaclave

Google confidential VMs

Veracruz
Veraison

Confidential Compute Consortium

- Linux foundation

Members

Premier

arm, 蚂蚁集团 ANT GROUP, HUAWEI, Microsoft, accenture, facebook, Google, intel, Red Hat

General

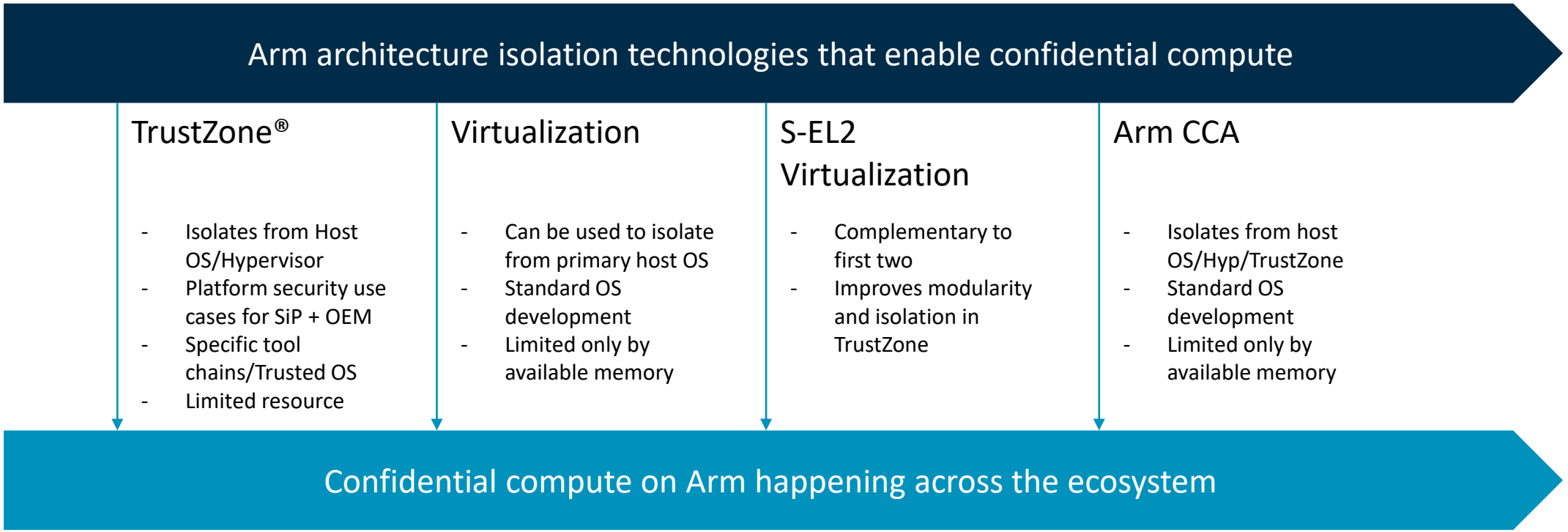
AMD, ByteDance, CYSEC, KINDITE, PHALA NETWORK, AMPERE, Anqlave, cosmian, OASIS LABS, NVIDIA, r3, anjuna, decentriq, Fortanix, swisscom, vmware, XILINX, EDGELESS SYSTEMS, MADANA, iExec, CRUST, CISCO, Western Digital

Associate

Linaro, IoTeX, CONFIDENTIAL COMPUTING CONSORTIUM

Confidential compute in Arm

Confidential Computing is the protection of data ***in use,*** by performing computation in a hardware-based secure environment, to shield portions of code and data from access or modification, even from privileged software.



Arm Confidential Compute Architecture – Overview

Requirements

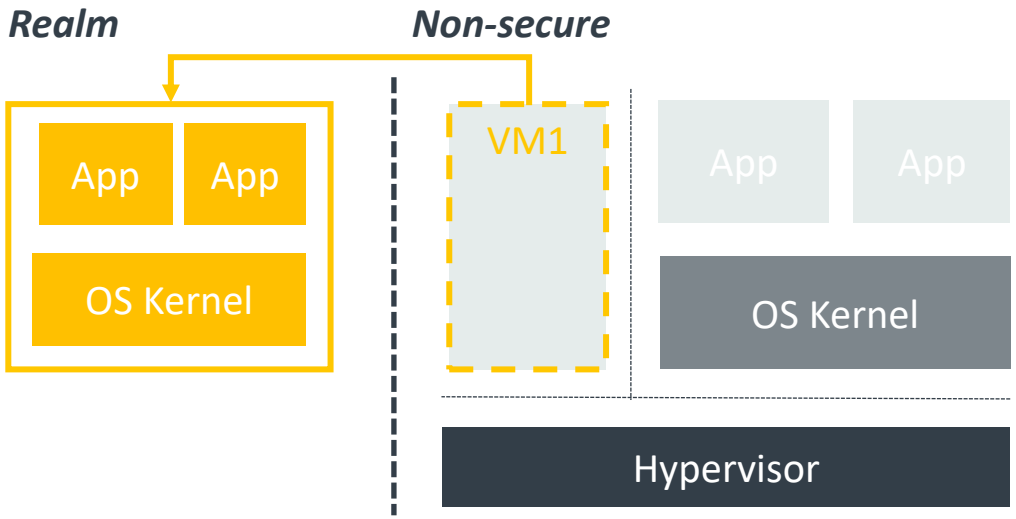
1. Secure execution environment that isolates content from more privileged SW:
   Host OS / Hypervisor, or TrustZone
2. Scale: There should be no specific limits on resources for these environments
3. Standard OS agnostic programming model – no platform specific drivers / toolchains
4. Attestable

Architecture

1. **Armv9-A Realm Management Extension** (FEAT_RME) introduces **Realms** and new isolation boundaries so that Realm content cannot be accessed by other Realms, by Host OS / Hypervisor, or by TrustZone
2. Dynamic memory: available memory dynamically moved between Realms or other use, e.g regular processes / VMs
3. Standard ABI to manage Realms
4. System HW architecture and standard ABI specifications to support attestation for Realms

Realms at the Virtual Machine Level

- Realms are supported at the virtual machine (VM) level
  - Very similar to AMD SEV-SNP / Intel TDX
- Hypervisor manages the resources of a Realm VM (memory, scheduling), but cannot access those resources
- Memory is protected in two ways:
  - Isolation: invalid accesses result in faults
    e.g. Hypervisor access of Realm memory causes a fault to that hypervisor
  - Encryption: mainly for reboot attacks
- Protection covers device DMA and processors



Realm Threat Model

Confidentiality	Mitigated by
Hypervisor/Host OS/TZ reads private Realm memory or register state	Arch
Device DMA reads private Realm memory or register state	
Integrity	
Hypervisor/Host OS/TZ state modifies private Realm memory or register state Examples: • Modify saved context • Writing to Realm pages • Memory remapping or aliasing	Arch
Device DMA modifies private Realm memory or register state	Arch
Availability	
Denial of service to a Realm – it is scheduled by OS/Hyp	⊘
Realm mounts a DoS attack on the hypervisor	Arch

Indirect SW attacks	Mitigated by
Known SW error injection – E.g.: CLKSCREW	Arch
Known side channels E.g.: Spectre / Meltdown	Arch Realm
Direct HW attacks	
Physical DRAM probe and replay	HW

Arch — Mitigated by Arm CCA (processor/SMMU/system and FW)

HW — Mitigation requires additional HW

Realm — SW in the Realm has the tools to protect itself

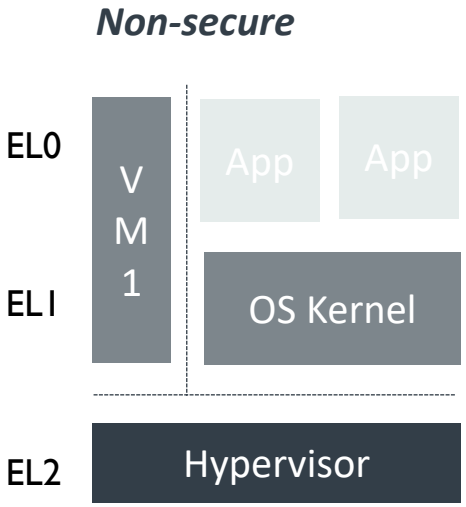
⊘ — Not mitigated

Arm CCA hardware architecture

Realm Management Extension (RME)

Charles García-Tobin

Arm Architecture



Like most architectures, we provide different levels of privilege

- EL0: User mode
- EL1: Kernel mode
- EL2: Hypervisor mode

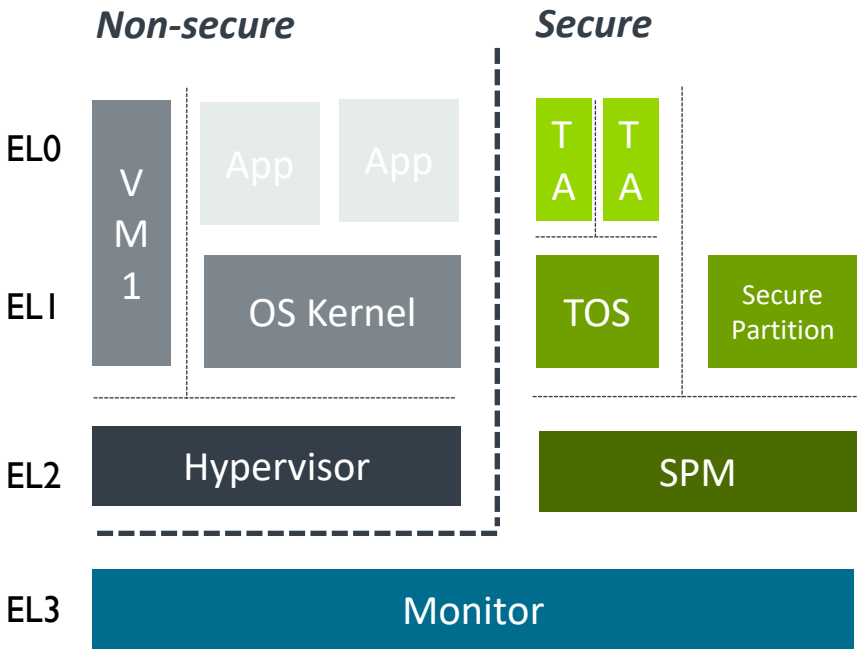
There are two stages of address translation:

- Stage 1 to translate VA → IPA (Intermediate Physical Address, a.k.a. Guest Physical Address)
- Stage 2 to translate IPA → PA

Note:

- Armv8.1 introduced the Virtual Host Extensions that supports running the kernel in EL2
- This is how Linux/KVM is run in many deployments today

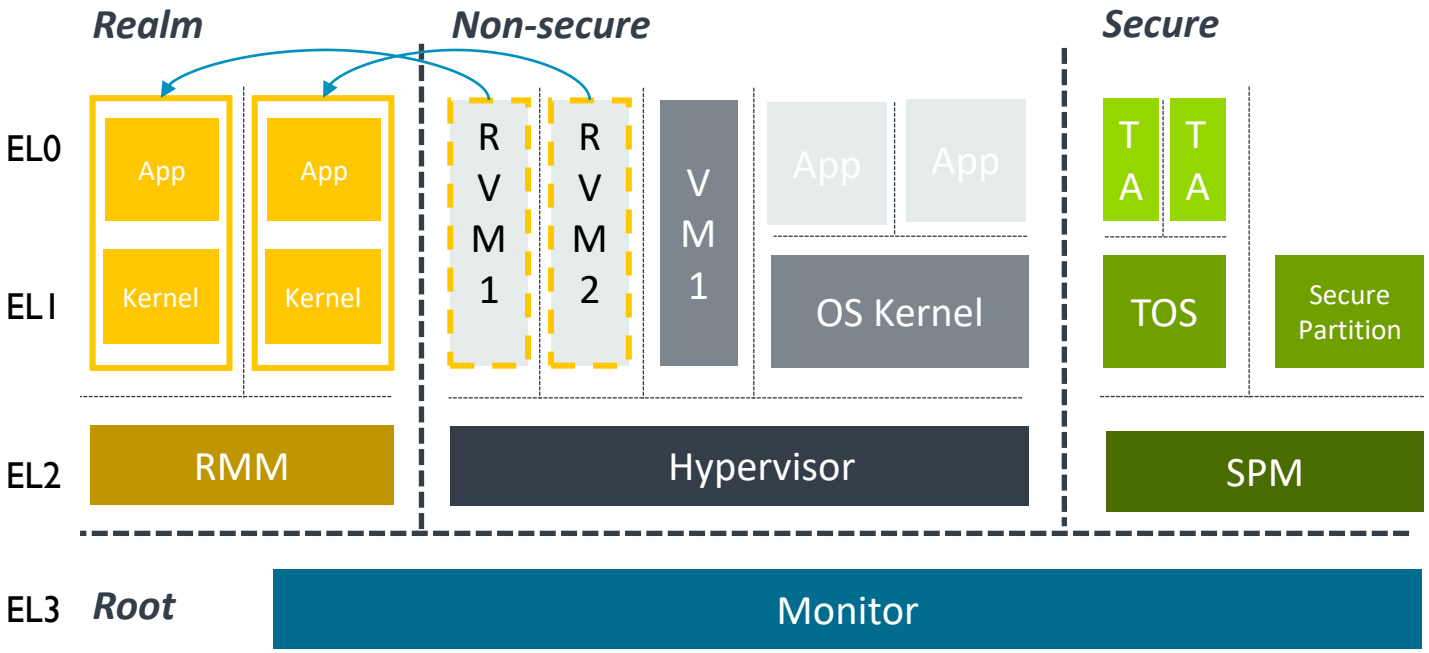
Arm CCA Hardware Architecture – TrustZone



Security State/PA space	Non-Secure PA	Secure PA
Non-secure	Allow	Block
Secure	Allow	Allow

- Two physical address spaces: Secure and Non-Secure
- At any point in time a processor can be in one of two security states: Secure or Non-Secure
  - Orthogonal to Exception level
    e.g a processor can be in Non-Secure-EL1 or Secure EL1
- Isolation boundaries based on security state
- EL3: monitor mode - used mainly to switch between security states
- No architectural mechanism to dynamically move memory /devices between Secure and Non-Secure PA
  - Memory for Secure PA is typically statically carved out
- TrustZone typically used by device vendors to provide platform security use case – includes processor and device support

Arm CCA Hardware Architecture – What does RME add?

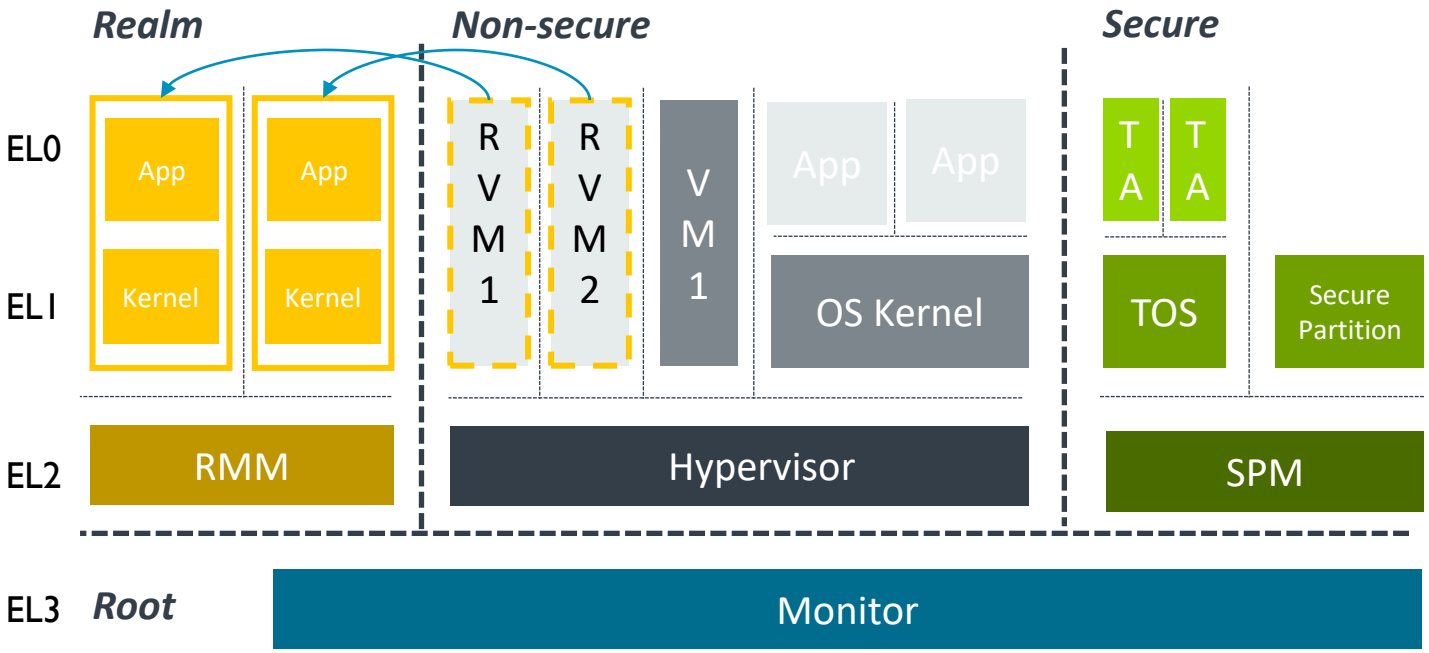


Security State/PA space	Non-Secure PA	Secure PA	Realm PA	Root PA
Non-secure	Allow	Block	Block	Block
Secure	Allow	Allow	Block	Block
Realm	Allow	Block	Allow	Block
Root	Allow	Allow	Allow	Allow

RME adds another two security states and physical address spaces

- Realm: new space for confidential compute
  - Realm VMs run in Realm state, but are managed from Non-secure state
  - Realm VMs can access their own private Realm physical address space, and share data via Non-secure physical address space
  - Secure and Realm are mutually distrusting
- Root: EL3 FW gets its own private address space

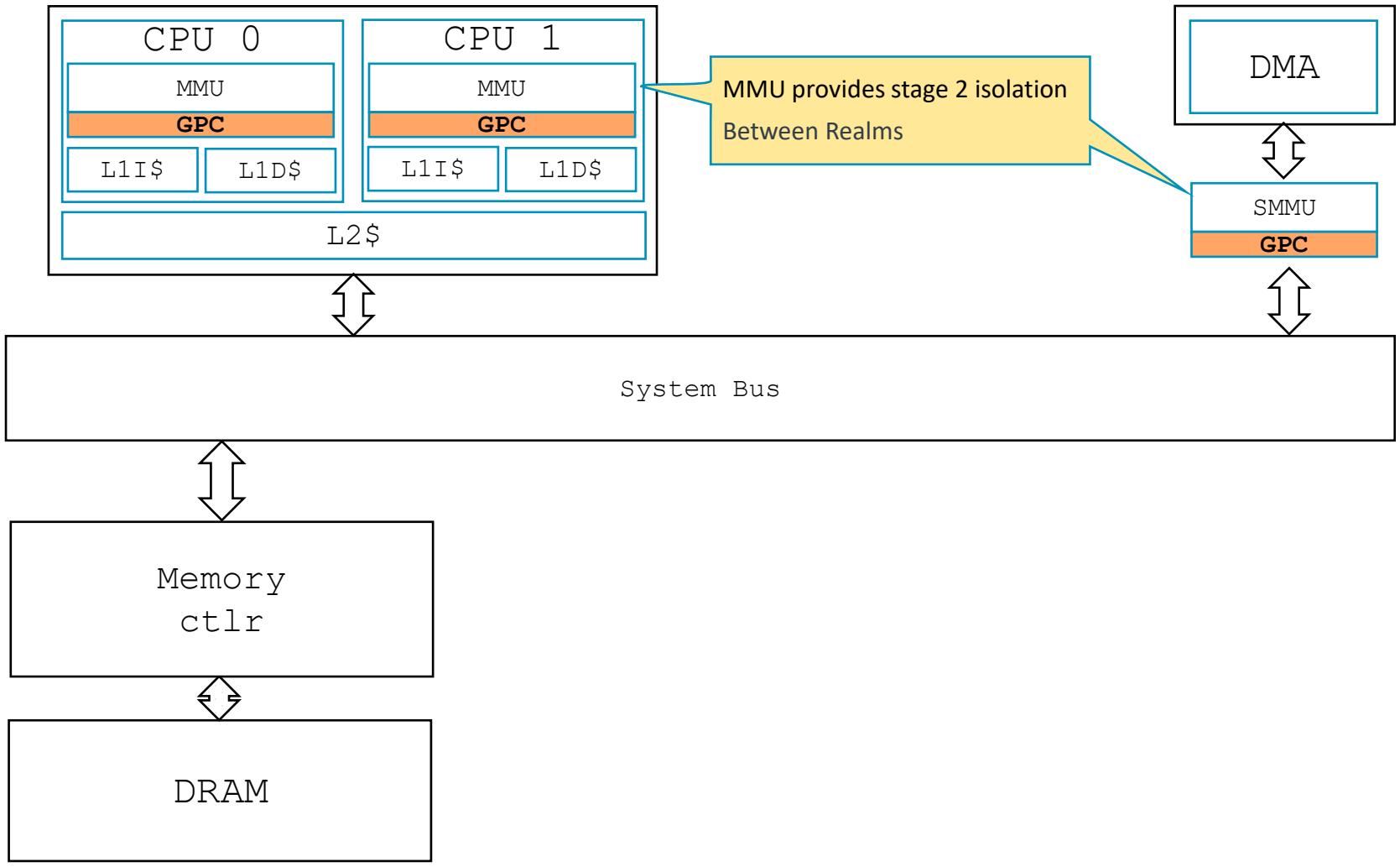
Arm CCA Hardware Architecture – What does RME add?



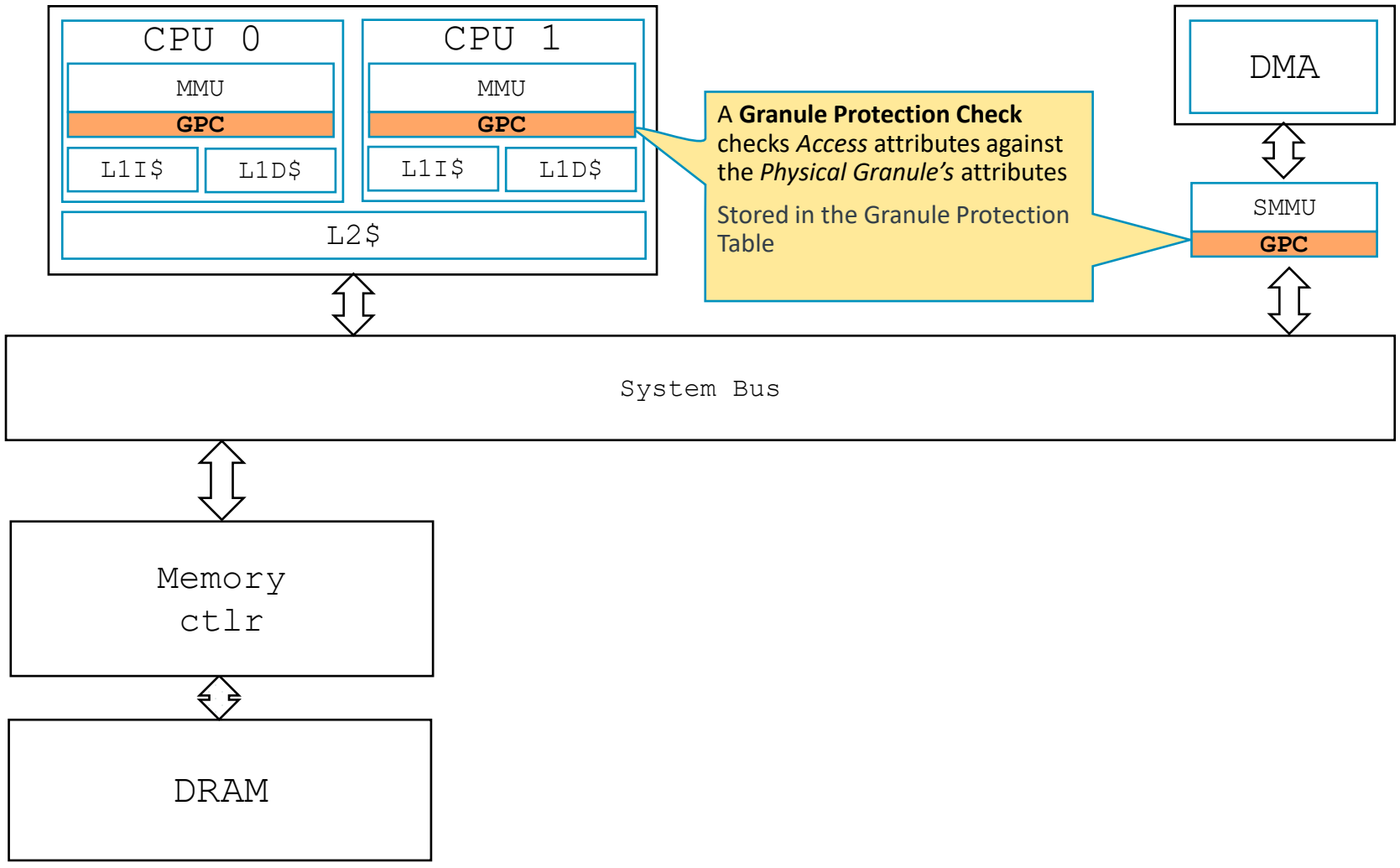
Security State/PA space	Non-Secure PA	Secure PA	Realm PA	Root PA
Non-secure	Allow	Block	Block	Block
Secure	Allow	Allow	Block	Block
Realm	Allow	Block	Allow	Block
Root	Allow	Allow	Allow	Allow

- Memory can move between physical address spaces **dynamically**
- **Granule Protection Table (GPT):** new data structure that determines the current physical address space of a page
- GPT is controlled by the monitor in EL3
- **Granule Protection Check (GPC)**: HW checks GPT on an access and faults invalid accesses
  - Faults in illegal accesses routable to EL2
  - Faults due to broken configuration go to EL3
- HW isolation between components in a security state uses page tables as before
  - Realm-to-Realm separation is based on page tables
- Root, Realm and Secure use encrypted memory
  - Boot ephemeral keys per address space and address tweak

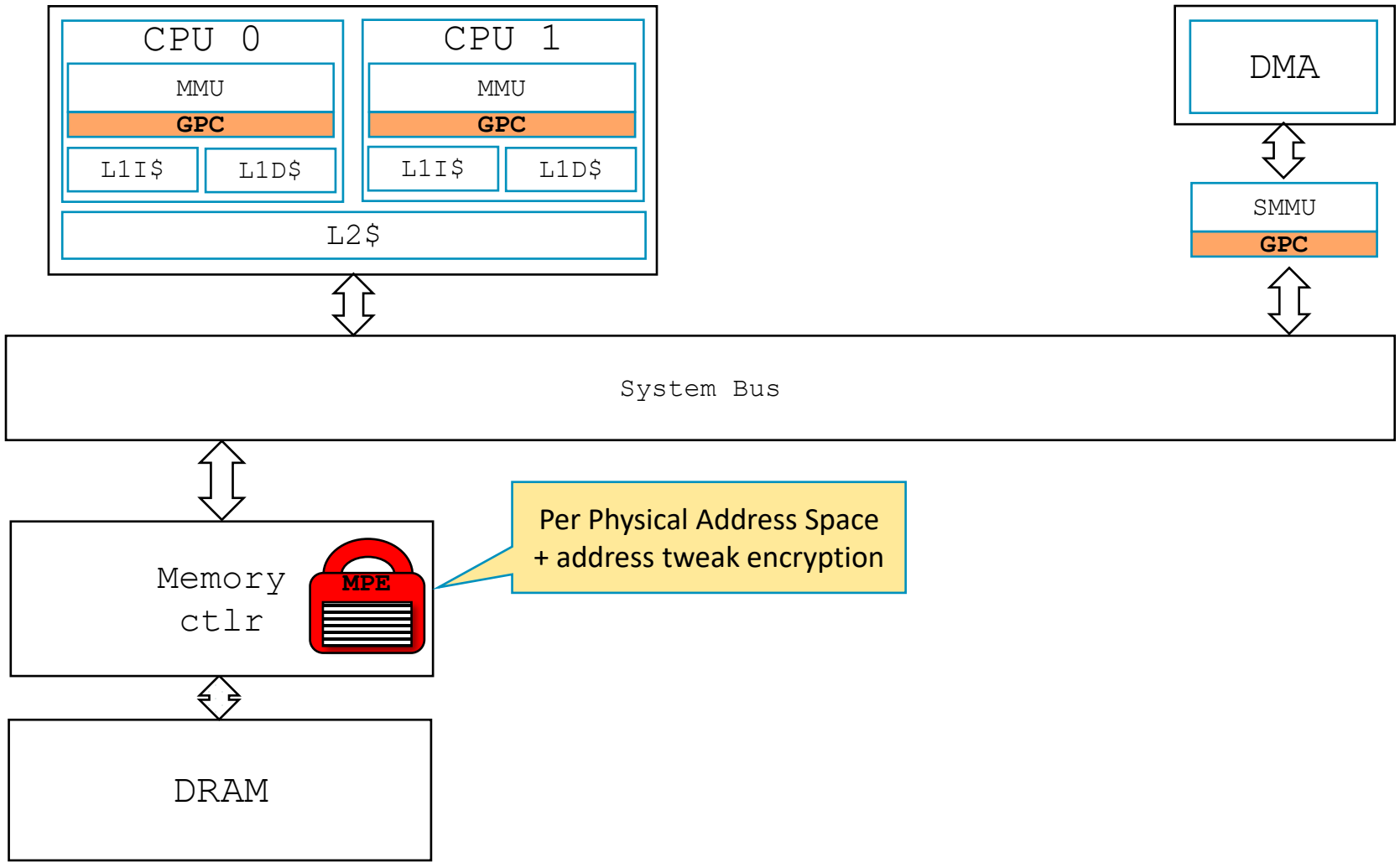
Arm CCA system concepts



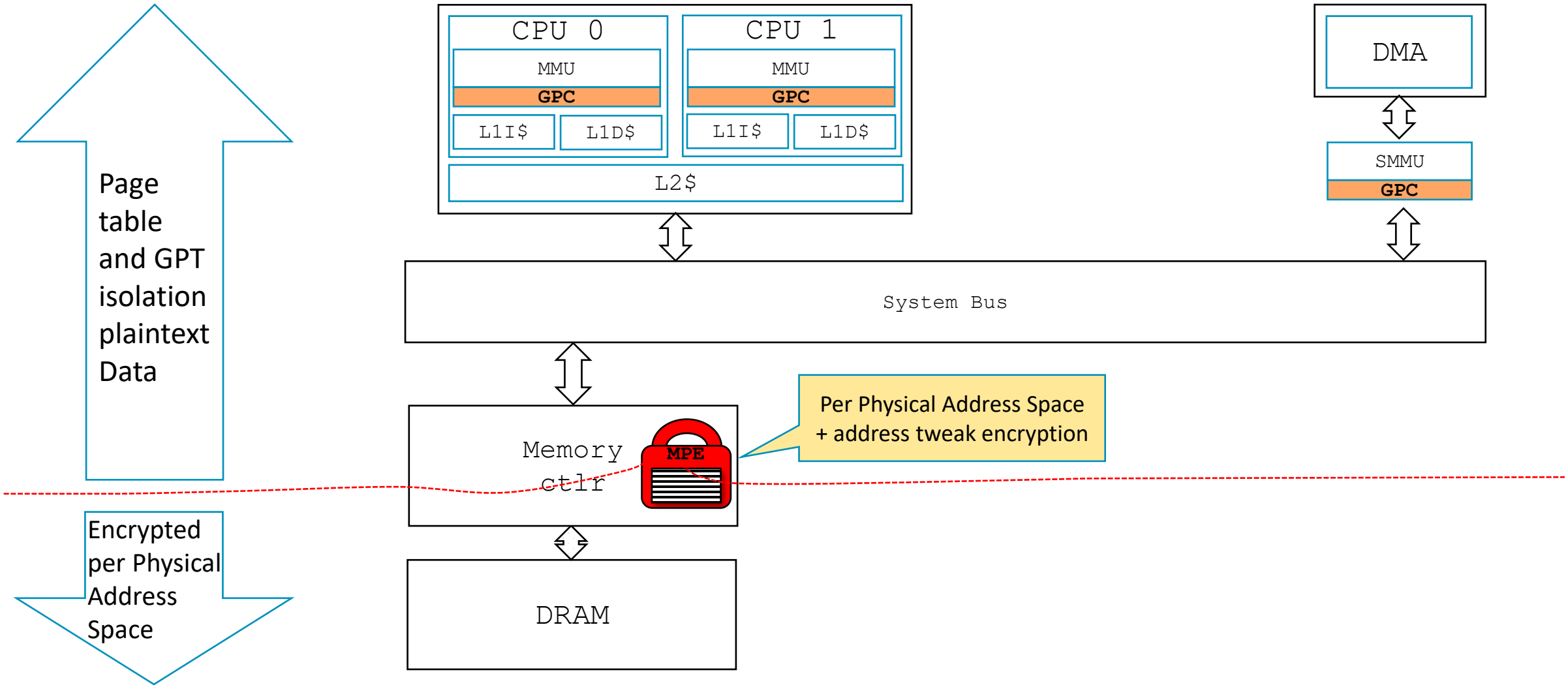
Arm CCA system concepts



Arm CCA system concepts



Arm CCA system concepts

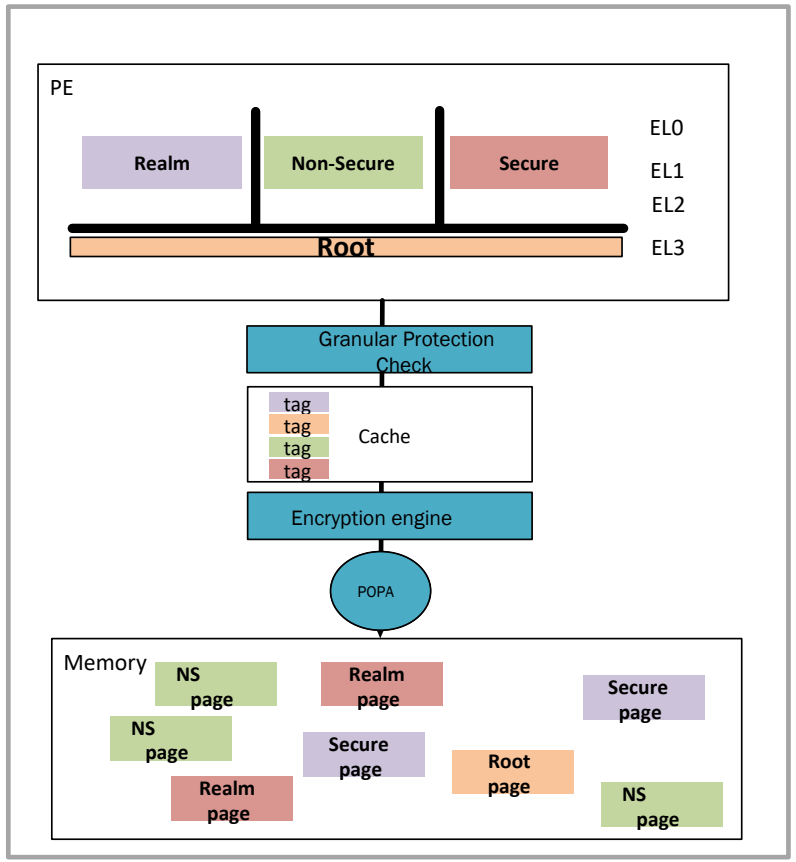


Arm CCA ISA impacts

New system instructions are introduced to be used by the monitor when it updates the GPT:

- Data Cache Clean/Invalidate to the PoPA
  - The architecture introduces a point of physical aliasing above which cache lines are tagged with the Physical Address Space they reside in
  - The instruction guarantees that no copies of a PA tagged with a Physical Address Space are above the PoPA
- TLB invalidation for GPT caching:
  - GPT association of a page with a physical address space is cacheable in TLBs – A TLB invalidation instruction is added to invalidate the association

Execution and data prediction restriction system instructions extended with new security states - CFP RCTX, CPP RCTX, DVP RCTX



Architectural state impacts

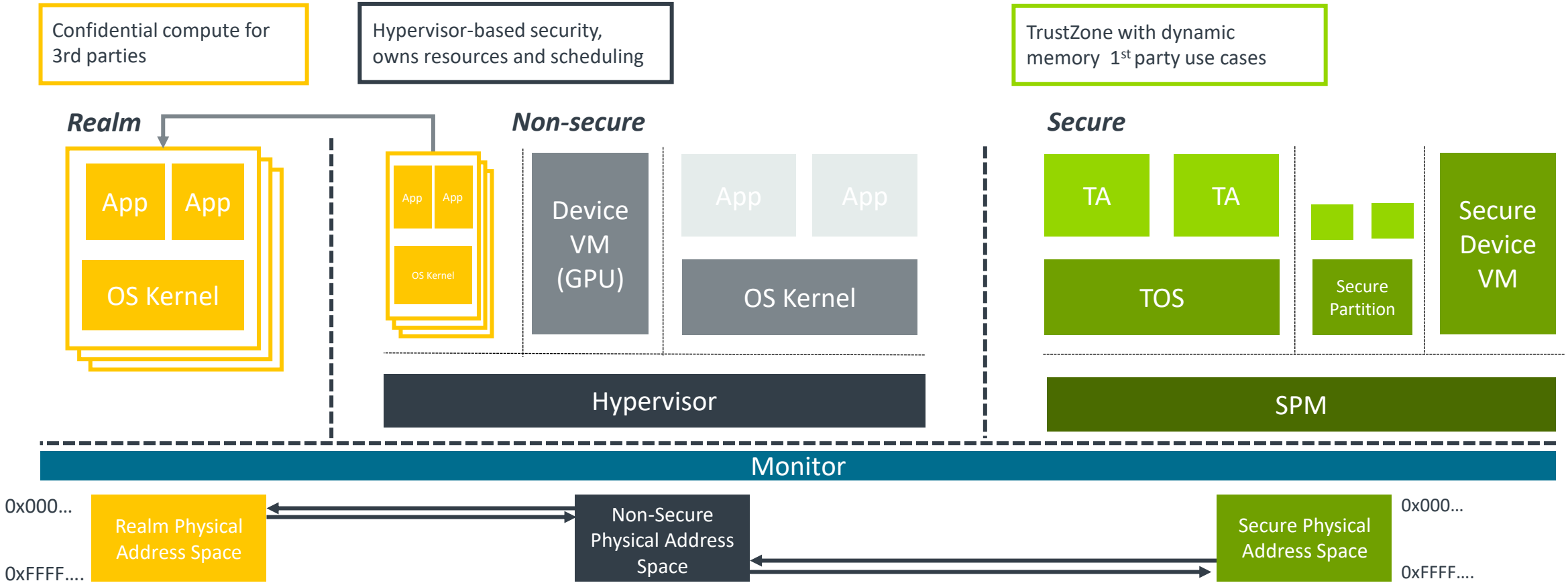
To keep HW simple most state is reused and replicated across Security States

- Configuration changes:
  - Added two Root registers for Granule Protection Table base address and configuration
- Granule Protection Check faults - treated similar existing MMU faults
  - Additional syndrome information (for loads/stores but also trace/sample profiling)
  - Routing of Granule protection faults
- Fields that represent Security state or Physical address space extended with new encodings
  - Mainly affects self hosted and external debug, trace, performance monitoring and sample profiling

MMU impacts

- The Physical Address Space of an access is derived from the output of MMU Stage 1/2 and verified by a Granule Protection Check:
  - Non-secure state: PAS is hard-wired to *Non-secure*
  - Secure state: existing NS bit in Stage 1 descriptor selects between *Secure* and *Non-secure* PAS
  - Realm state: ***new*** NS bit in Stage 2 descriptor selects between *Realm* and *Non-secure* PAS
  - EL3 stage 1: two bits (one ***new***) allow specifying 4 PAS-es
- The following Translation Regimes are supported in the Realm security state:
  - Realm EL1&0
    - e.g. Stage 1 and Stage2 of an EL1 Realm VM
  - Realm EL2
    - for Real management firmware
  - Realm EL2&0
    - allows RMM to manage the memory for EL0

Arm Confidential Compute Architecture

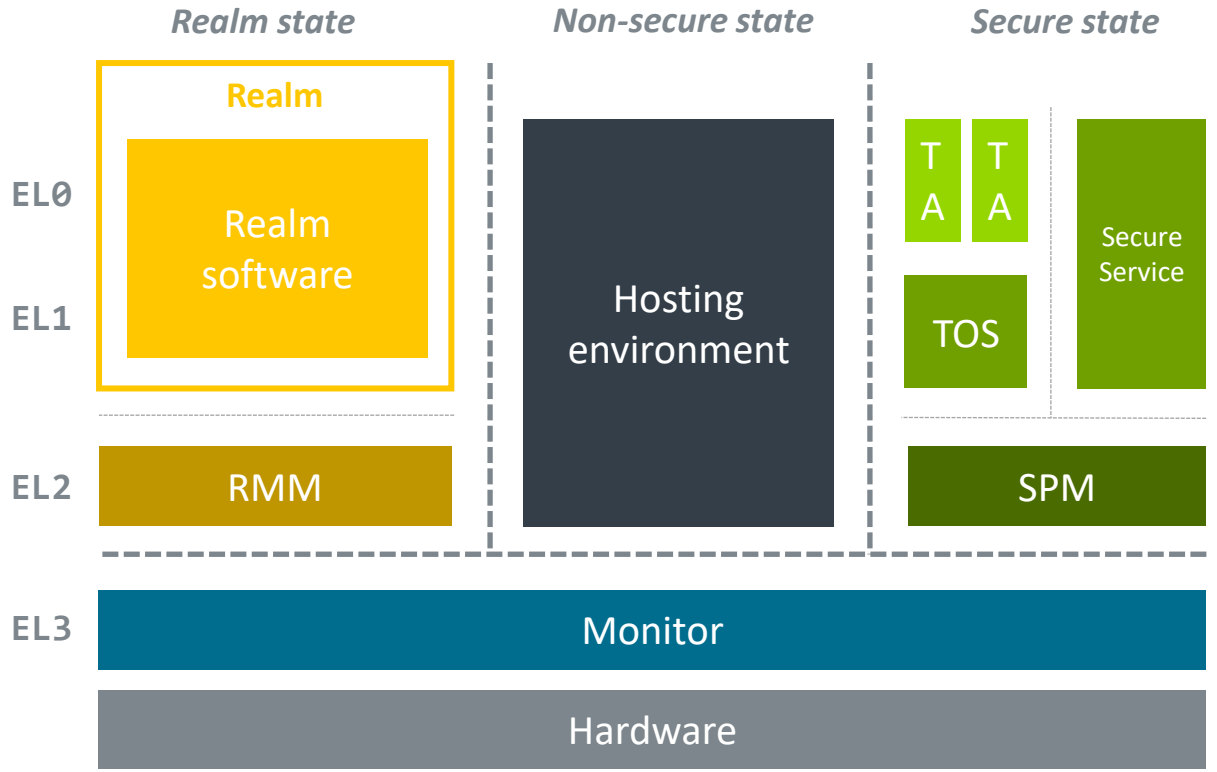


- Arm CCA adds a new environment for 3rd party confidential compute: **Realm world**
- By isolating Realms into their own private Realm World, system wide security analysis is greatly simplified
- With Arm CCA, memory can move dynamically between worlds
- Hypervisor owns management of resources

Arm CCA = RME hardware + CCA firmware

Realm Management Monitor (RMM)

- Manages Realm execution environment and inter-Realm isolation
- Allows Non-secure host to create, schedule and manage Realms



Monitor manages

- Context switching CPU execution between security states
- Memory access permissions

Hardware provides isolation primitives

- Additional processor security states
- Memory access control

arm

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה