linaro™

Arm Solutions at Lightspeed

Arm CCA Open Source Enablement Status

21-June-2024, Kevin Zhao, Linaro

linaro

Agenda

CCA Software Stack Introduction

CCA attestation

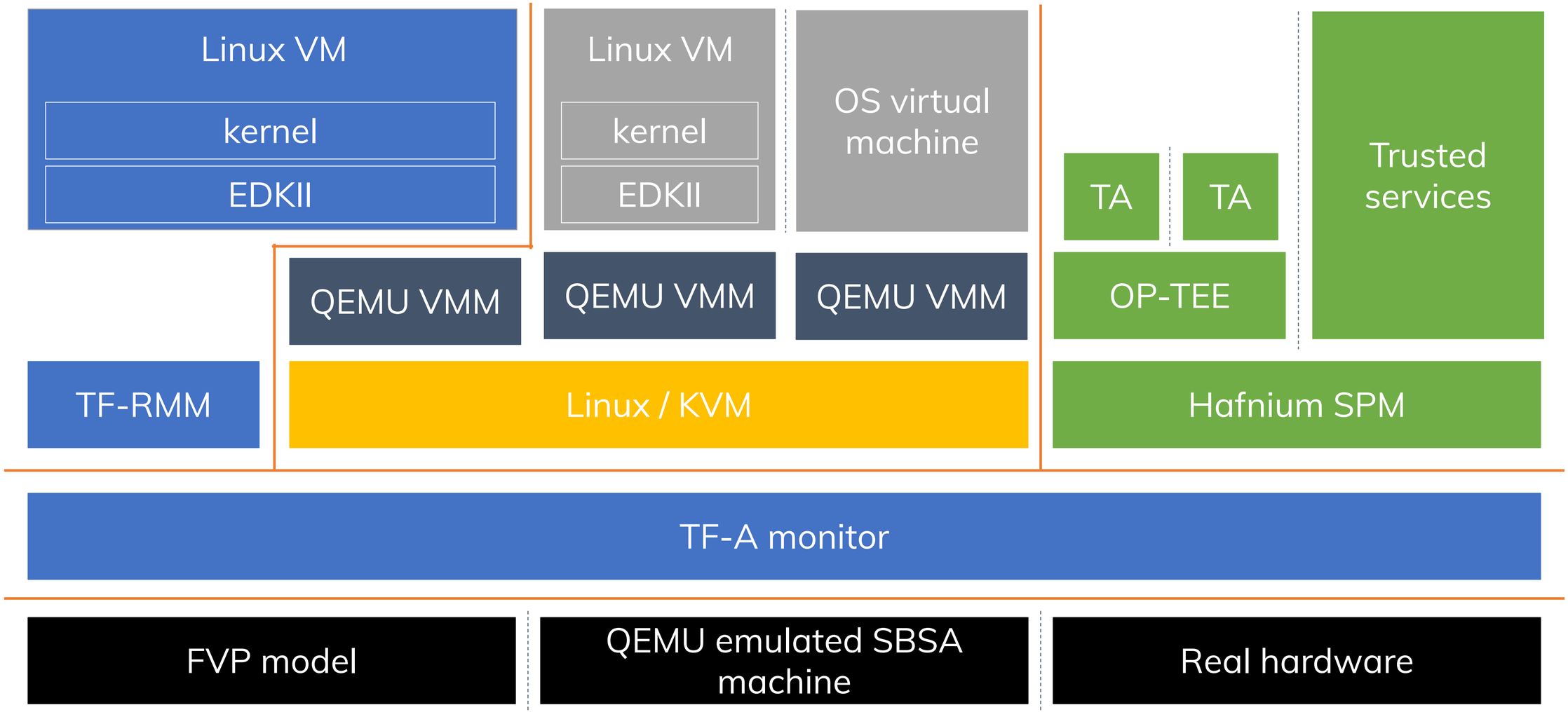
CCA 1.1 Specs

CCA Software Stack Introduction

linaro™

Arm Solutions at Lightspeed

Low level reference software stack



CCA Low level software reference stack

Arm developed a CCA stack that runs on their FVP model.

With the release of QEMU 8.1, Linaro ported that stack to QEMU:

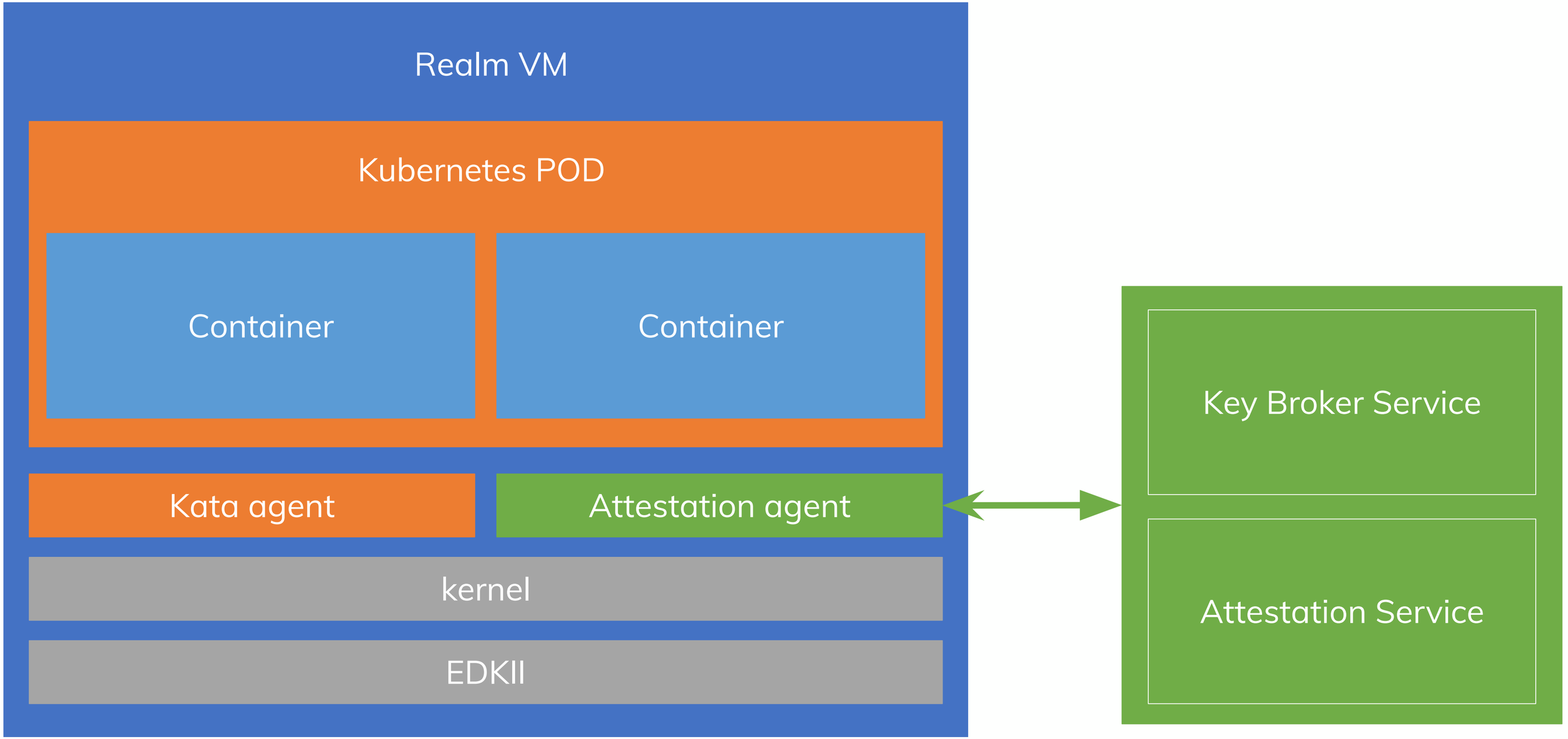
- Patches for TF-A, TF-RMM and EDK2 are available on CodeLinaro (cca/v2 branch).
- Patches for the Linux kernel and kvmtool are hosted by Arm (cca/v2 branch).
- The solution is currently for the QEMU virt machine type with buildroot.
  - QEMU as a system emulator with RME support and as a VMM launching Realms.

Work to support QEMU SBSA reference machine type is ongoing.

Support for RME in Linaro's Trusted Reference Stack (TRS) is ongoing. Plans for a CI.

Documentation is available to compile and run the stack, from base system to Realm.

High level reference software stack



CCA high level software reference stack

Kata container support: code repo.

- Current features:
  - Only supports Kata v2.
  - Only supports QEMU back end.
  - Only supports direct kernel boot with Kata. The UEFI boot disk image has been validated.
  - Only supports ACPI=off in QEMU.

Confidential Containers (CoCo):

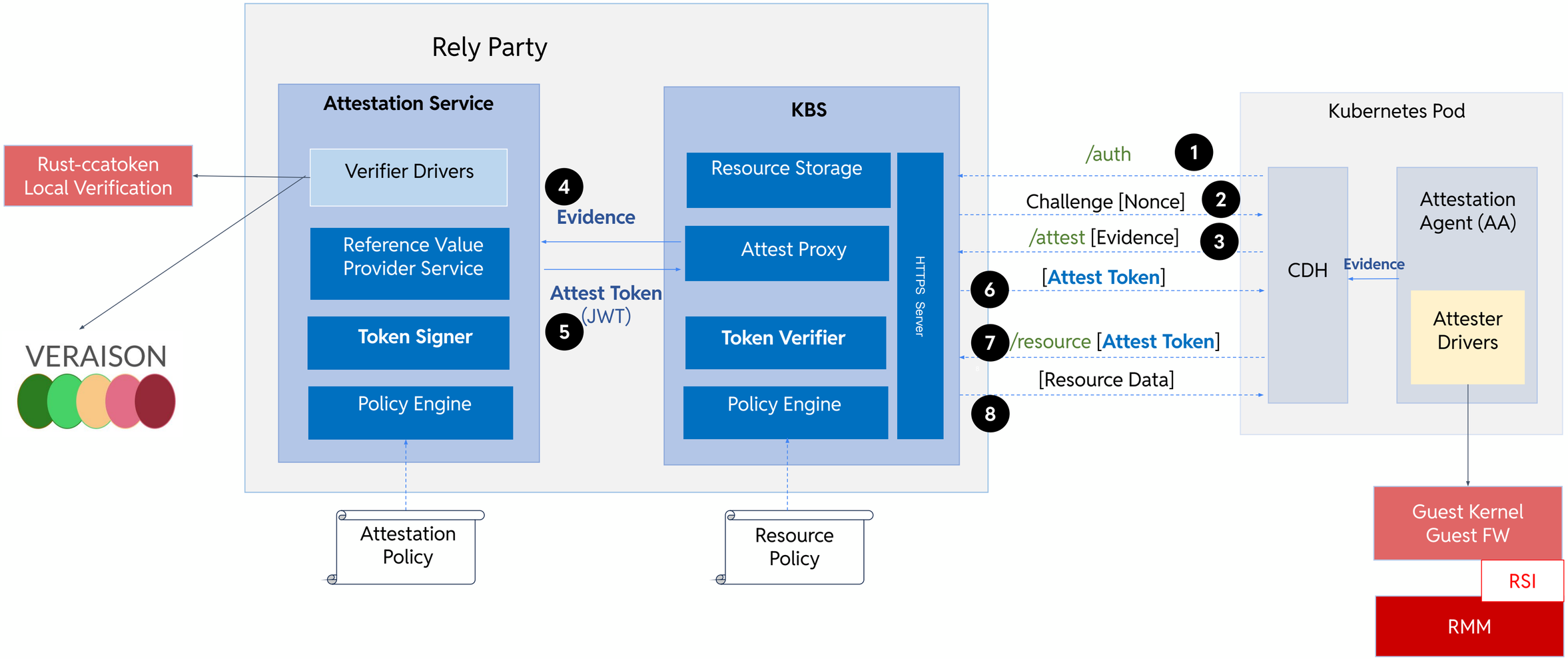
- Framework adoption
  - Kubernetes Confidential Computing operator
  - Container image service (service offload, encryption verification).
- Trustee support: code repo.

CCA Attestation

linaro™

Arm Solutions at Lightspeed

CoCo and Veraison - remote attestation



Attestation tools

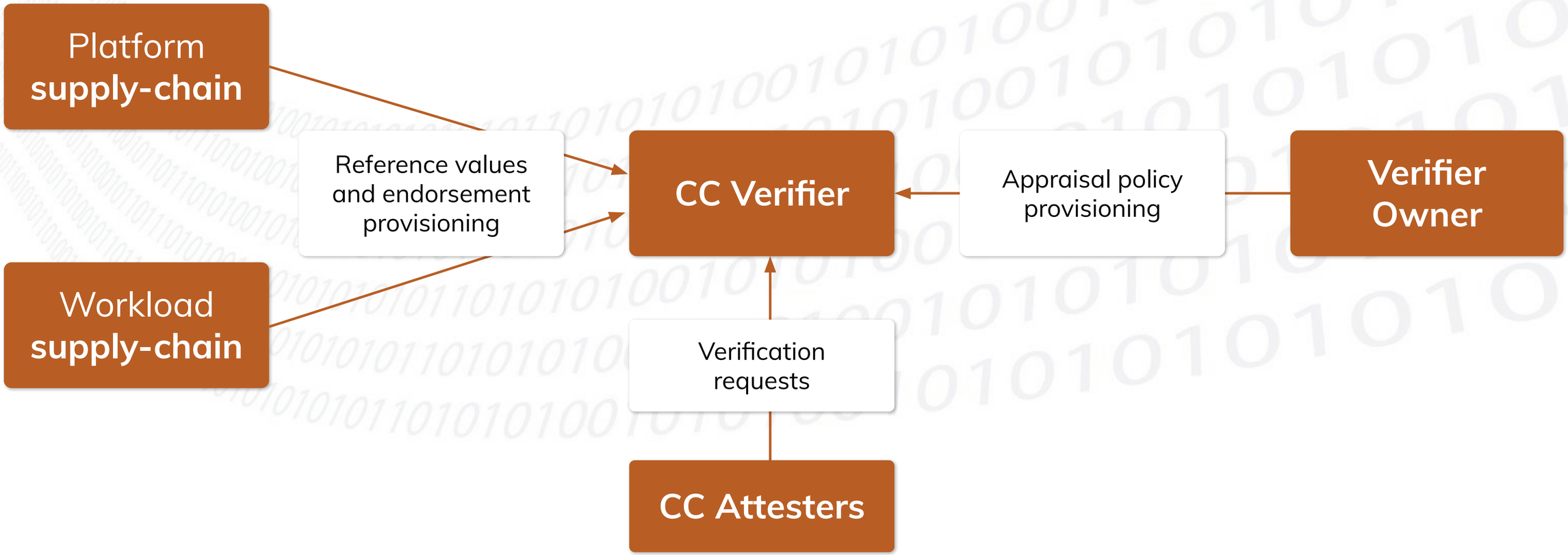
ccatoken crate

- Provides command line tools and APIs to decode and verify CCA attestation tokens.
- Published at https://crates.io/crates/ccatoken.
- Sources at https://github.com/veraison/rust-ccatoken.

realm-token crate

- Tool that calculates the Realm initial and extended measurements, needed for CCA attestation.
  - Sources at https://git.codelinaro.org/linaro/dcap/realm-token.

Remote attestation verification



Future steps

QEMU support for memory encryption.

QEMU support for SMMU. This is a requirement for device assignment.

Cloud Hypervisor support.

Lightweight firmware support for Arm CCA.

End-to-end demo for CoCo on Arm CCA with Qemu backend.

Integration of Veraison and CoCo Attestation Service (AS) to provide a holistic end to end reference solution for confidential containers on Arm platforms.

CCA 1.1 Specs

linaro™

Arm Solutions at Lightspeed

CCA 1.1 features – needed for initial deployments

Further strengthen the security guarantees provided to end users (Realm owners)

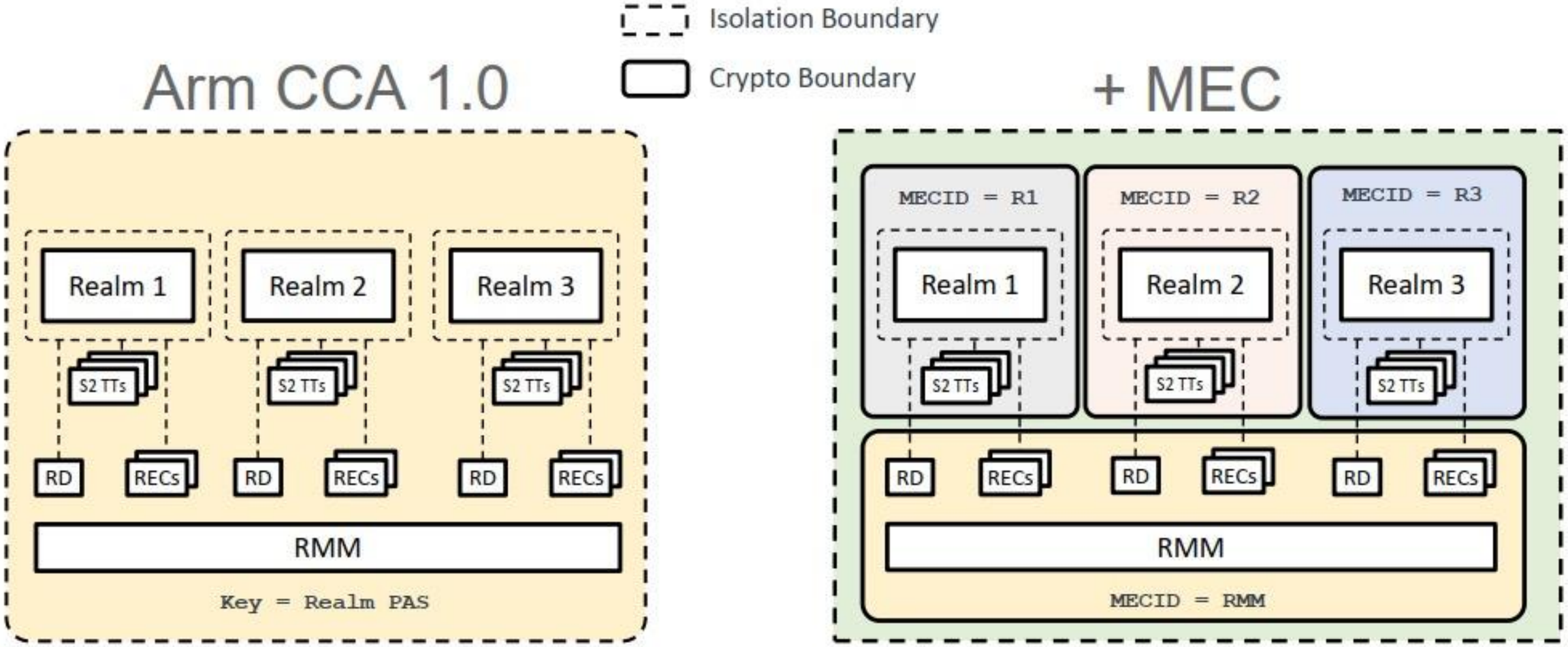
- Memory Encryption Contexts (MEC)
  - Physical memory contents of each Realm protected using a unique key or tweak
- Multiple signers
  - Require firmware image to be endorsed by multiple authorities, for example vendor plus a trusted auditor

Enable migration of workloads from non-secure VM to Realm, by providing feature parity

- Planes
  - Multiple privilege levels within a Realm, orthogonal to traditional kernel / user-space split
- Device Assignment (DA)
  - Enable trusted device functions to be admitted into a Realm's TCB, and granted DMA
- Host Debug of Realm
  - In a controlled environment, enable host to debug a realm (bypassing CCA security guarantees)

Allow platform owners additional flexibility, in deploying and updating firmware

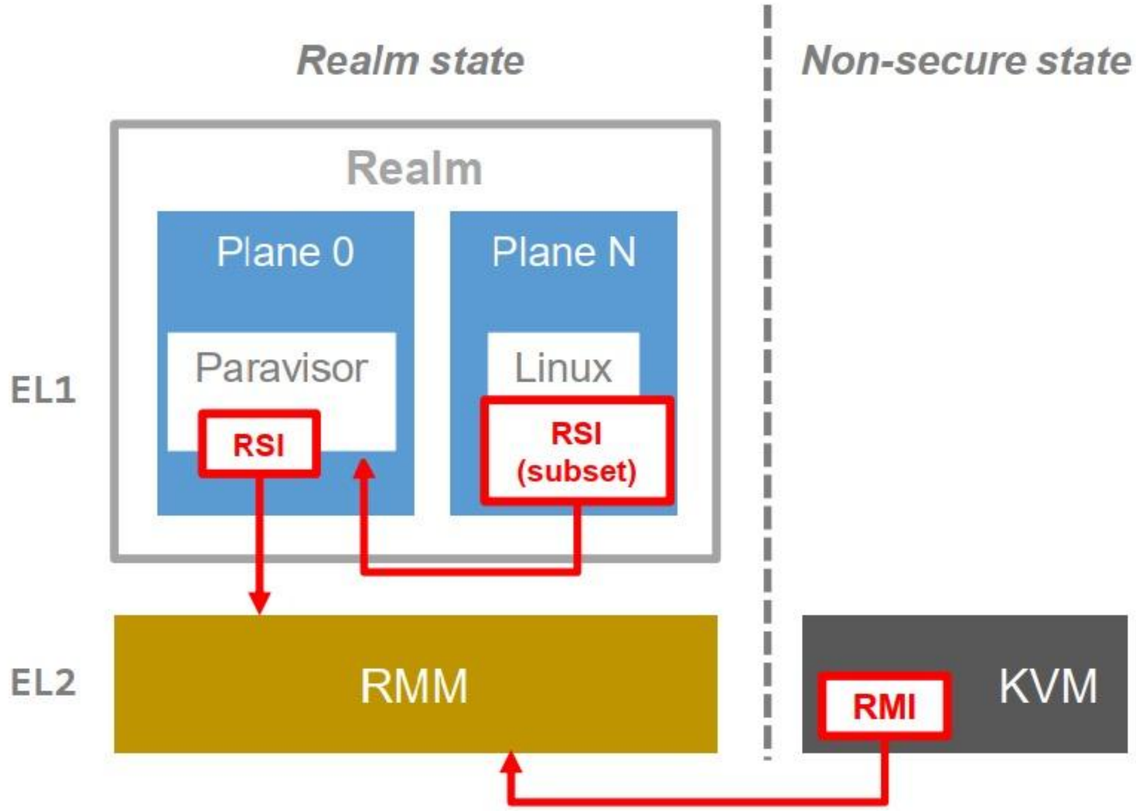
- Live firmware activation
  - Update firmware image(s) while workloads continue to run, with minimal loss of availability
  - Replace platform firmware (for example, RMM) with an image supplied by the non-secure host



- TF-RMM impact: Small
  - Enforce MEC ID uniqueness
- Realm guest Linux impact: Zero
- Host Linux/KVM impact: X-Small
  - Allocate a MECID for each Realm

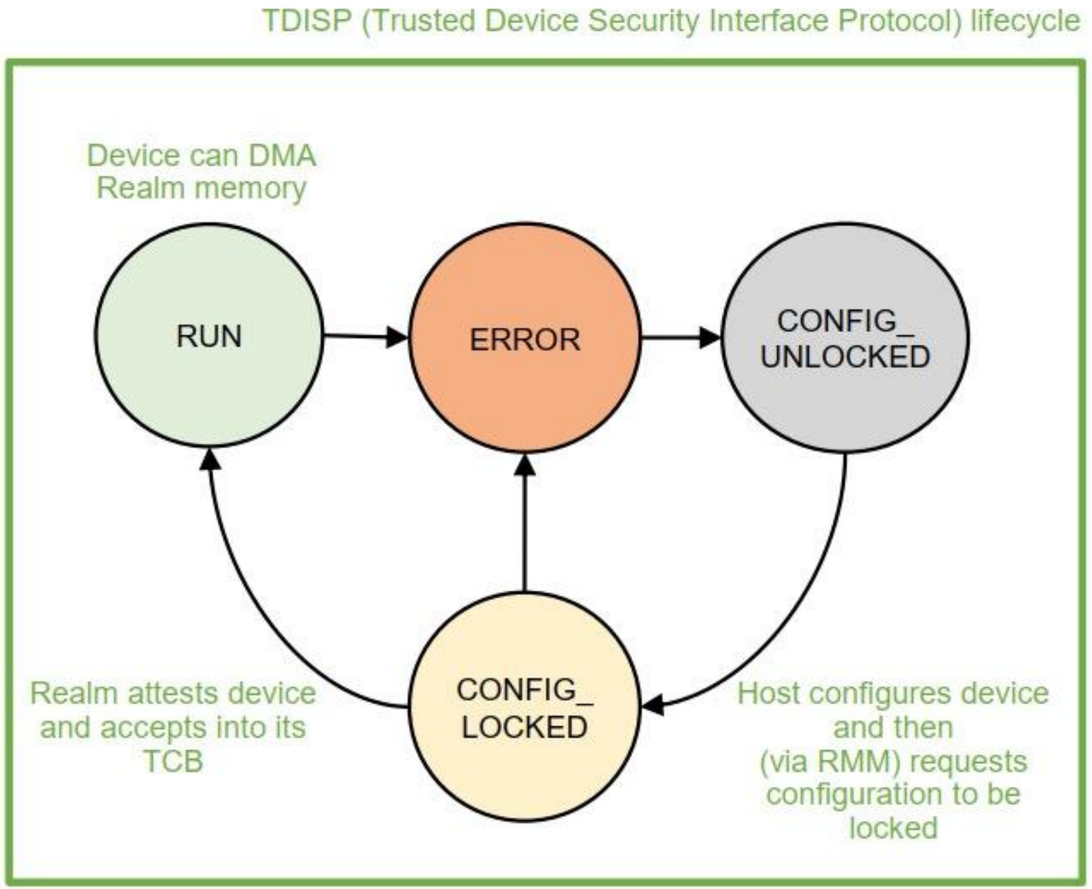
Planes

- In addition to the main guest OS and user workload, allow the contents of a Realm to include other software components (Planes)
  - For example, a security service like vTPM
- Provide isolation within a Realm, allowing privilege separation between the Planes
  - All Planes have Same IPA → PA mappings but IPA memory permissions may differ Allow the host hypervisor to continue treating
- the Realm as a single unit, for the purposes of resource allocation, scheduling and migration Within the
- Realm, privileged Plane 0 assigns resources to the other Planes



Device Assignment

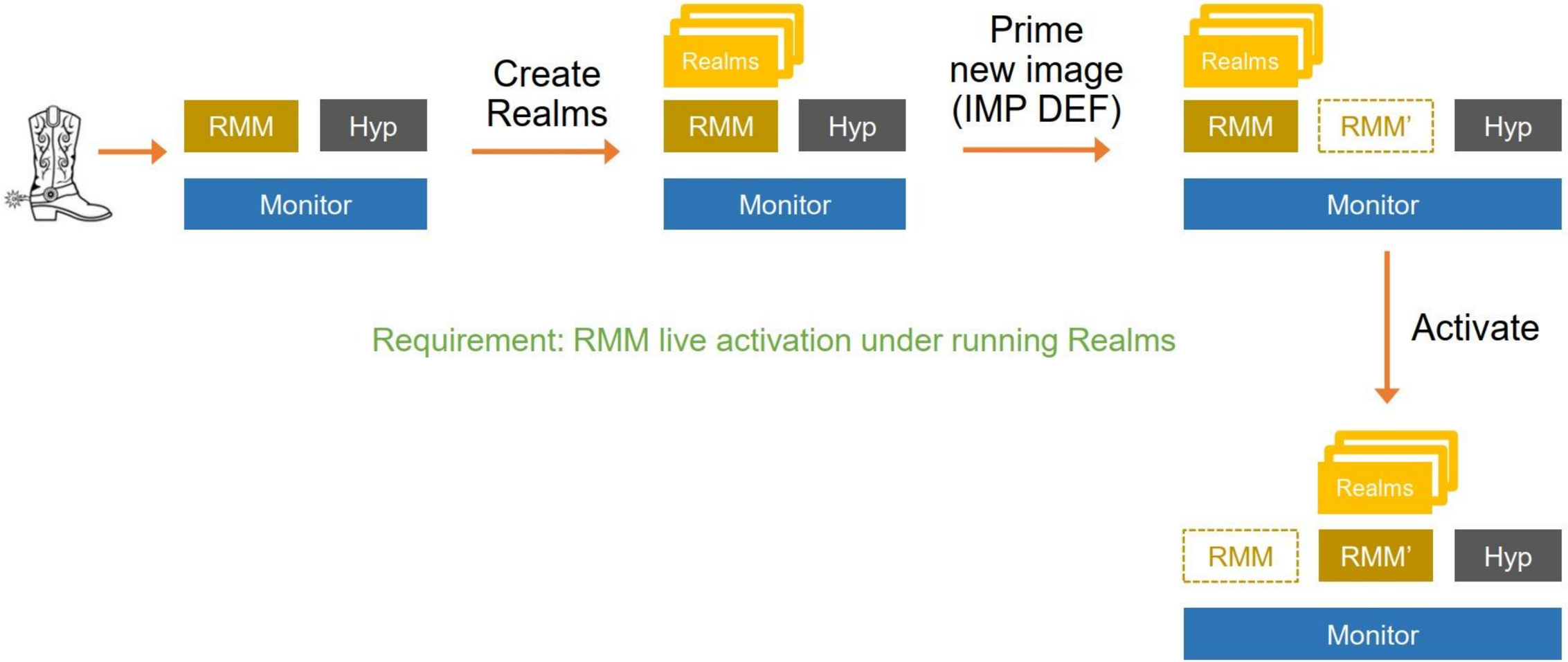
- Allow hypervisor to assign a PCIe TDISP device to a Realm
  - Also support coherently-attached devices, such as CXL instances*
  - Also support on-chip PCIe devices*
- Allow Realm to attest the identity and configuration of the device function
- Device lifecycle guarantees that
  - DMA is blocked until device has been approved by the Realm
  - Any changes in device configuration cause transition to an error state, which revokes DMA
  - Once removed from a Realm, device guarantees that it will scrub confidential state
- Management of device lifecycle must be standards-based
  - RMM must not require any device-specific knowledge
  - However, RMM will require knowledge of platform topology



Device Assignment

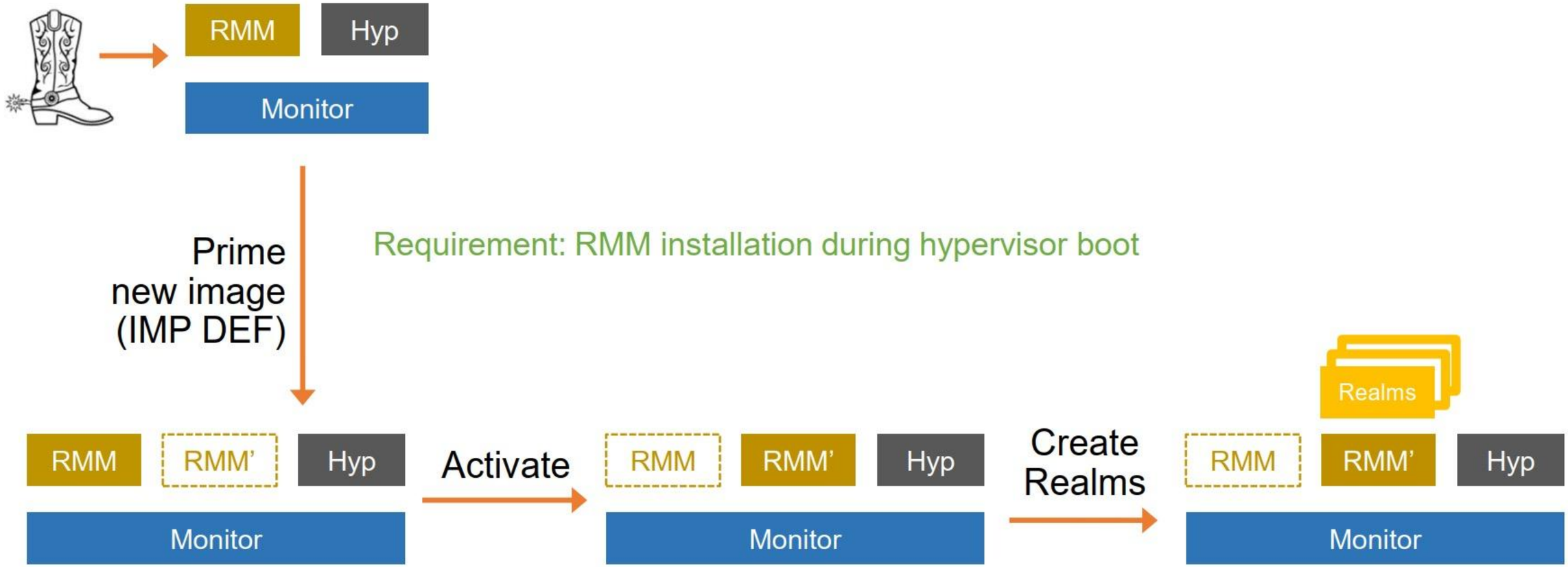
- TF-A / TF-RMM impact: Large (foundation) -> X-Large (advanced)
  - Implement DA ABIs and integrate PCIe standard reference libs to enforce device lifecycle
  - SMMU S2 driver + SMMU S1 emulation + PCIe root port programming
- Realm guest Linux impact: Large (foundation) -> X-Large (advanced)
  - Generic PCIe / driver support for trusted devices
    - Irrespective of whether running on Confidential-Compute (CC) or non-CC VMs
  - Community Drive:
    - Driven by PCI Dev maintainer
  - Arch specific backends for TDISP will plug in to CCA interface hooks
- Host Linux/KVM impact: Large (foundation) -> X-Large (advanced)
  - DA enlightenment (use new RMIs to control device lifecycle)
  - UABI for VMM to describe DA devices – shared across architectures
  - SMMU stub (for interrupt management) for RMM SMMU Driver

Live firmware activation



Requirement: RMM live activation under running Realms

Live firmware activation



Live firmware activation impacts

- TF-A / TF-RM
  - Create staging area for new firmware and transfer live state to new image
  - Live activating an arbitrary firmware version is hard – may need to restrict use-cases initially
    - For example, limit to specific code sections or require new version to be data compatible
    - Will focus on RMM and BL31 (EL3 firmware) live activate initially
    - Live activating the latter is especially hard (for example, may require CPU reset)
- Can increase use-cases over time (for example, by versioning data structures)
- Also need hooks to authenticate new firmware and to update firmware measurement log
  - Actual authentication is platform specific
- Realm guest Linux impact: None (hopefully)
- Host Linux/KVM impact: Medium
  - Use new ABIs to provide cycles to prime/activate new firmware
  - May need to quiesce activity and rendezvous CPUs during activation phase

Live firmware activation impacts

What to expect next

2024

- Continued upstreaming of CCA v1.0 Linux / KVM patches
- Monthly releases of RMM v1.1 spec
  - ALP with early DA / Planes support available now
  - Individual features will reach BET through the year, as they mature
- Collaborative development of CCA v1.1 SW (prototyping in progress)

2025

- Final RMM v1.1 spec (EAC)
- Upstreaming of non-DA-related CCA v1.1 features as they mature
  - Much quicker for TF projects than Linux / KVM
- Continued development of CCA v1.1 DA features and start upstreaming foundation support
- Quarterly Arm solution releases of integrated stack with CCA v1.1 features

Thank you!

linaro™

Arm Solutions at Lightspeed