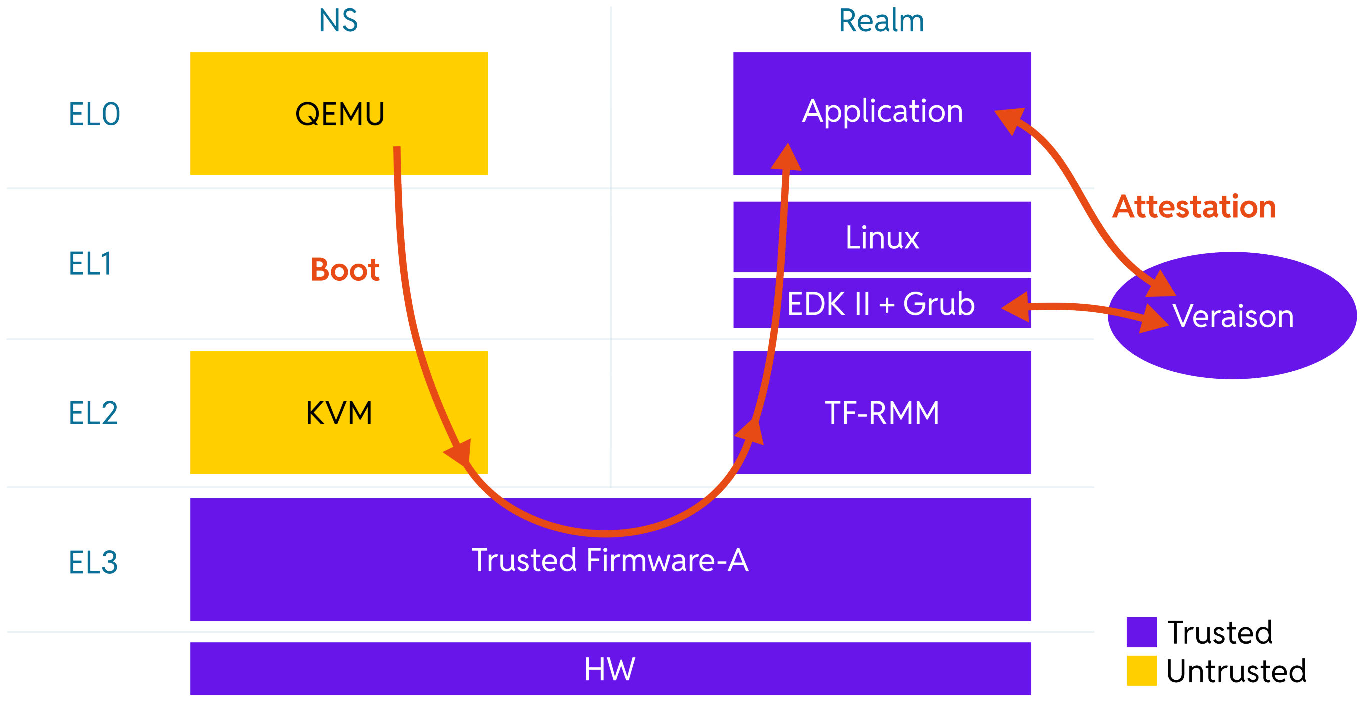
Virtual Machine Attestation on Arm CCA

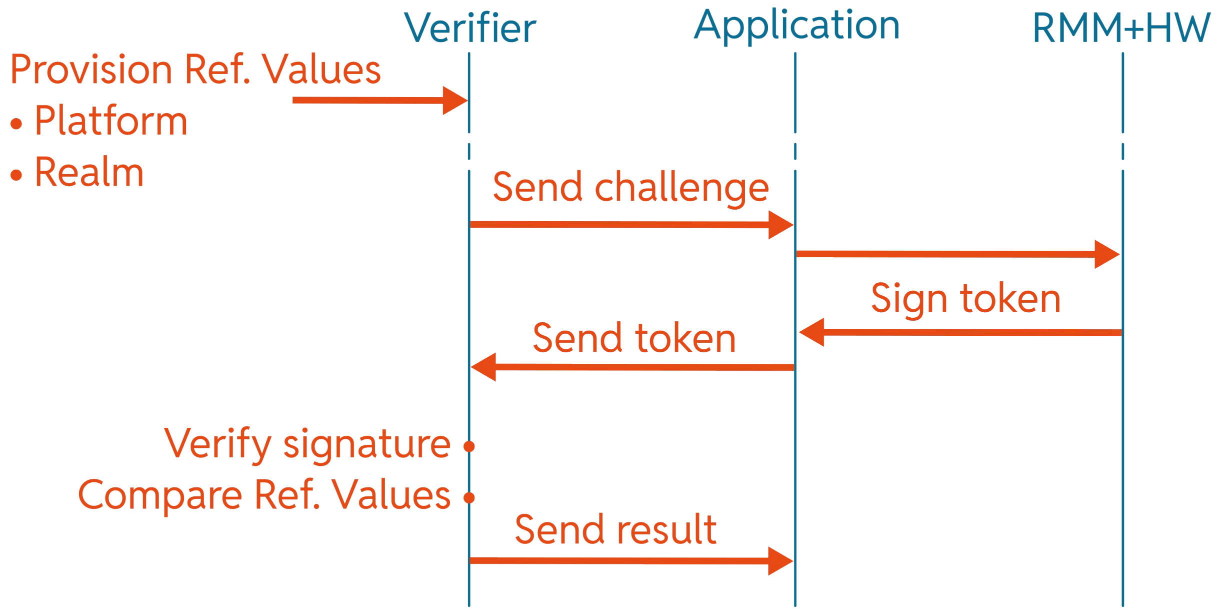
FOSDEM 2025

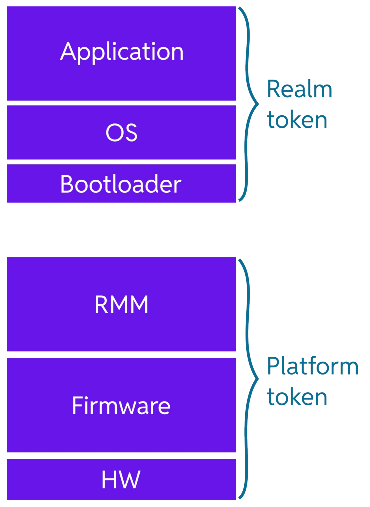
Realm boot and attestation on Arm CCA



🟪 Trusted
🟨 Untrusted

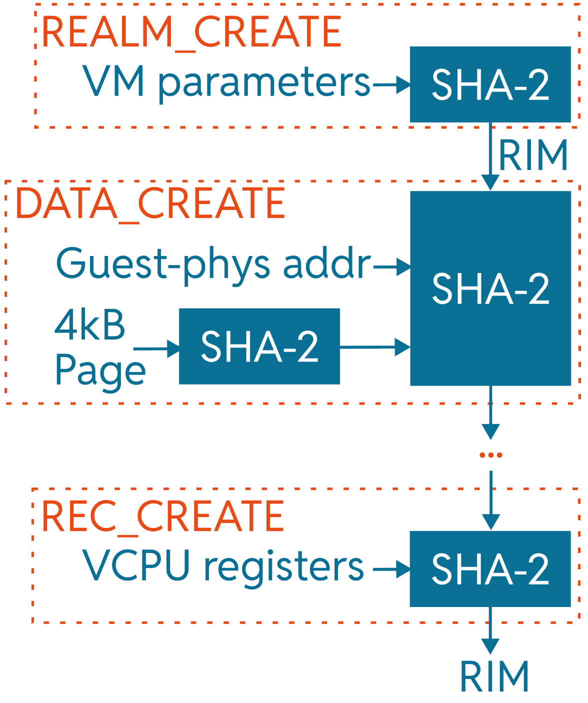
Attestation on Arm CCA





Computing the Realm Token

- **RIM**: *Realm Initial Measurement*, a hash of the state of the VM at reset
- **REM**: *Realm Extensible Measurements*, four hashes for runtime measurements



Computing the RIM

As a Reference Value provider, how do I compute the RIM?

➔ Easy: run it once and write down the RIM.

➔ Don't own the machine? Do it offline.

Use ✨**cca-realm-measurements**✨ command-line tool + rust library

```
$ cca-realm-measurements <host-config> <images> qemu <arguments>
RIM: 62072e353a762a55…
```

Problem: there is no standard Arm VM

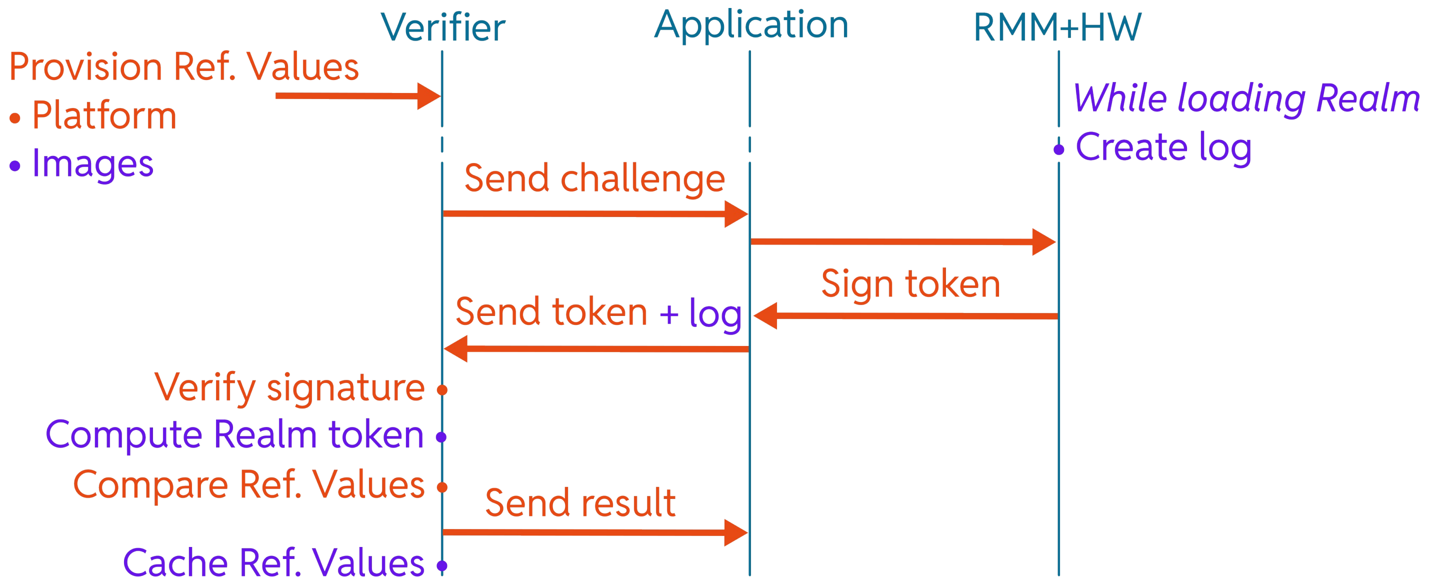
➔ Define canonical initialization order

➔ Specify each virtual platform, generate the firmware tables

Computing the RIM dynamically

- Problem: poor scalability $N_{RIMs} = N_{VMM\ versions} \times N_{vCPUs} \times N_{RAM\_sizes} \times N_{images} \times N_{opt\ A} \times N_{opt\ B} \times \ldots$

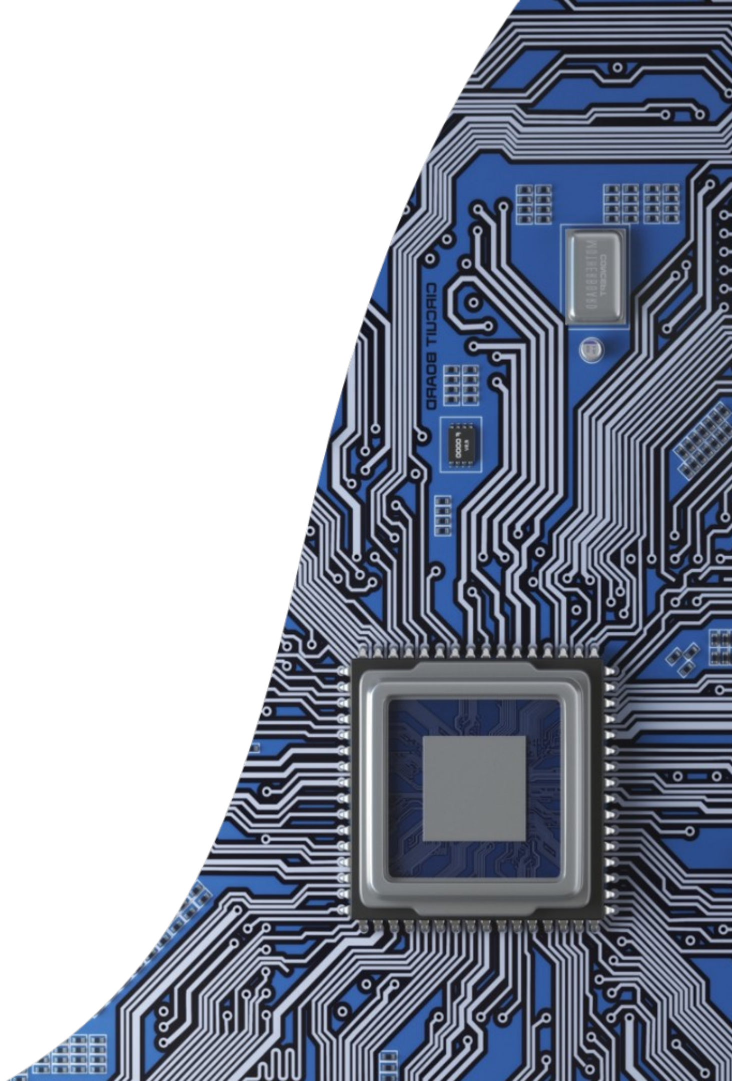
➔ Compute the Reference Values dynamically



Conclusion

- At least three options to compute a Realm token
- PoC implementation for offline and event_log: ✨cca-realm-measurements✨
- Needs input from users. What to improve:
  - More VMMs
  - Standardize:
    - VM formats?
    - Attestation protocol for sending token + log
    - Event log format (new events types for TCG TPM2)
  - Interoperability with other projects (eg. IGVM)

Extras



Links and references

- https://github.com/veraison/cca-realm-measurements
- Learn the architecture - Arm Confidential Compute Architecture software stack
- Build and run the CCA stack on QEMU
- TCG PC Client Specific Platform Firmware Profile Specification Event log specification
- QEMU PATCH v3: Run Arm CCA VMs with KVM RIM event log proof of concept
- IGVM describes load order to the VMM

Pre-calculating the RIM

Requirements:

- Host hardware capabilities
- Hypervisor (implementation choices eg. page table allocation order)
- VMM capabilities and enabled features
- Firmware/kernel/initrd images, where and in which order are they loaded
- Initial vCPU registers (entry point, device tree address)
- Firmware tables (= machine description) loaded into the VM

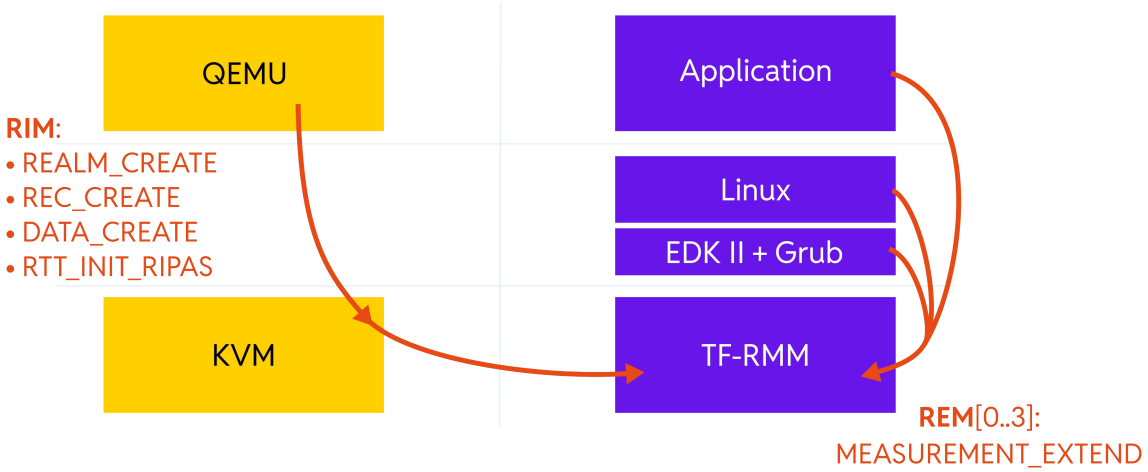
Measuring the firmware tables

Do we need to measure the firmware tables? Not necessarily, but

- Untrusted host provides the DTB/ACPI tables and could for example:
  - Add extra nodes to exploit vulnerable drivers
  - Add pointers to MMIO regions under host control, fake initrd
  - Change kernel parameters to disable hardening
  - Introduce out of bounds value to confuse a lenient parser
- To validate the FW tables at runtime:
  - All components (FW, OS) that parse the tables must now have a strict validator
  - Upheaval of the threat model has a significant maintenance and review cost. Each DT and ACPI change must now anticipate this new threat.

Realm Extensible Measurements

- Four registers
- Realm software extends them (REM + hash → REM)
- Need a log as well


QEMU
RIM:
• REALM_CREATE
• REC_CREATE
• DATA_CREATE
• RTT_INIT_RIPAS
KVM
Application
Linux
EDK II + Grub
TF-RMM
REM[0..3]:
MEASUREMENT_EXTEND