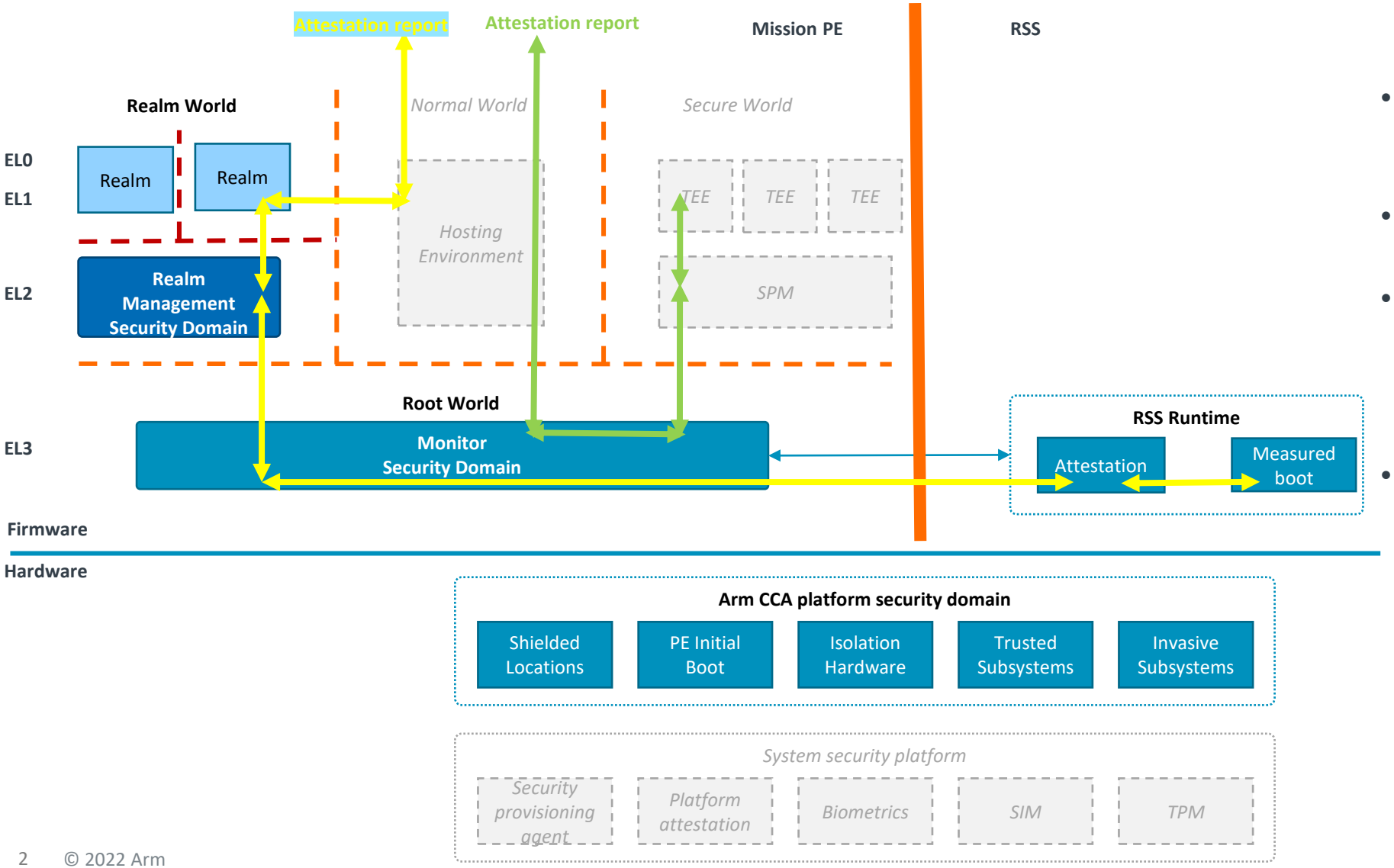
Attestation and Measured Boot

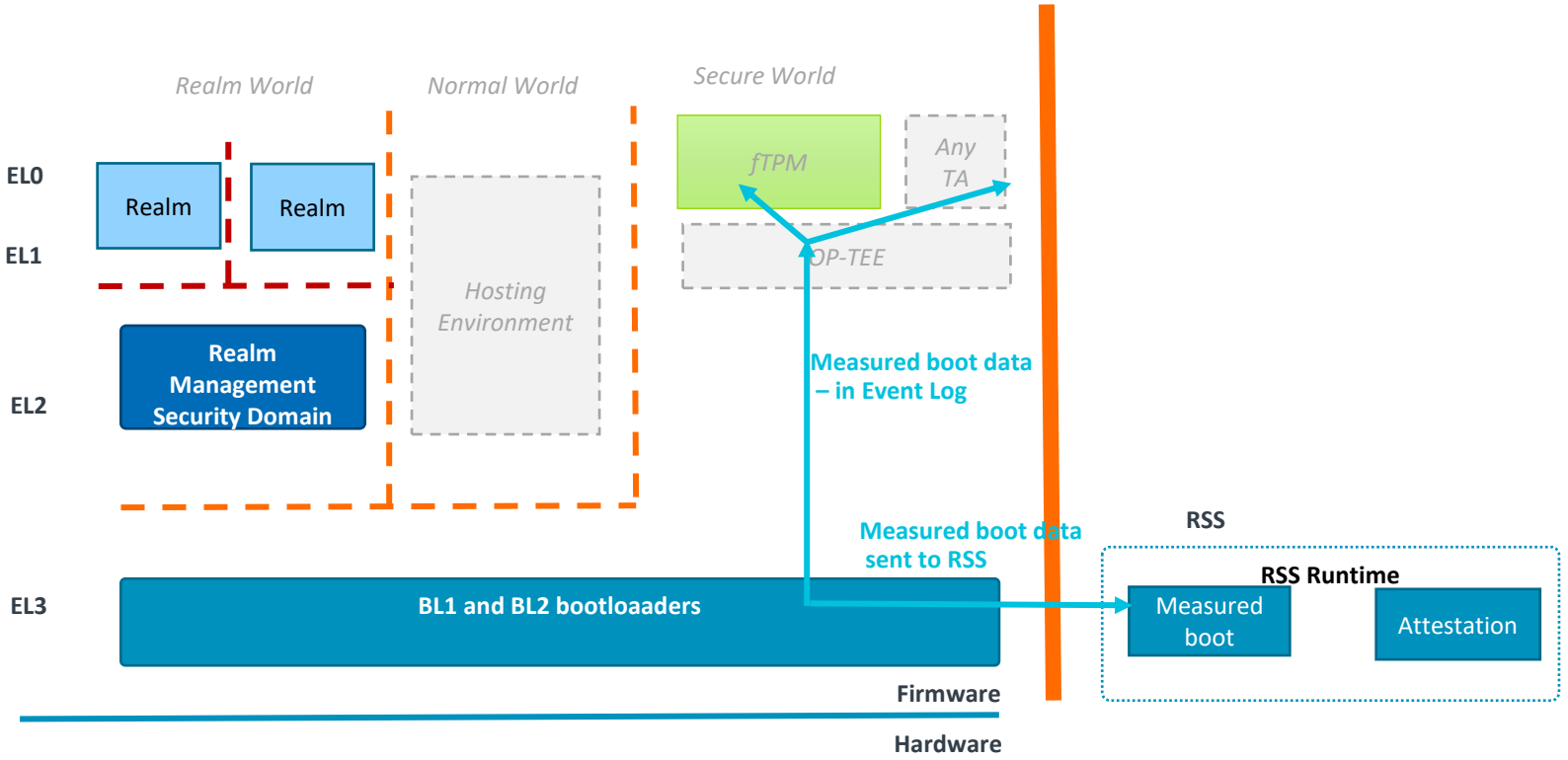
Tamas Ban

Attestation on high level



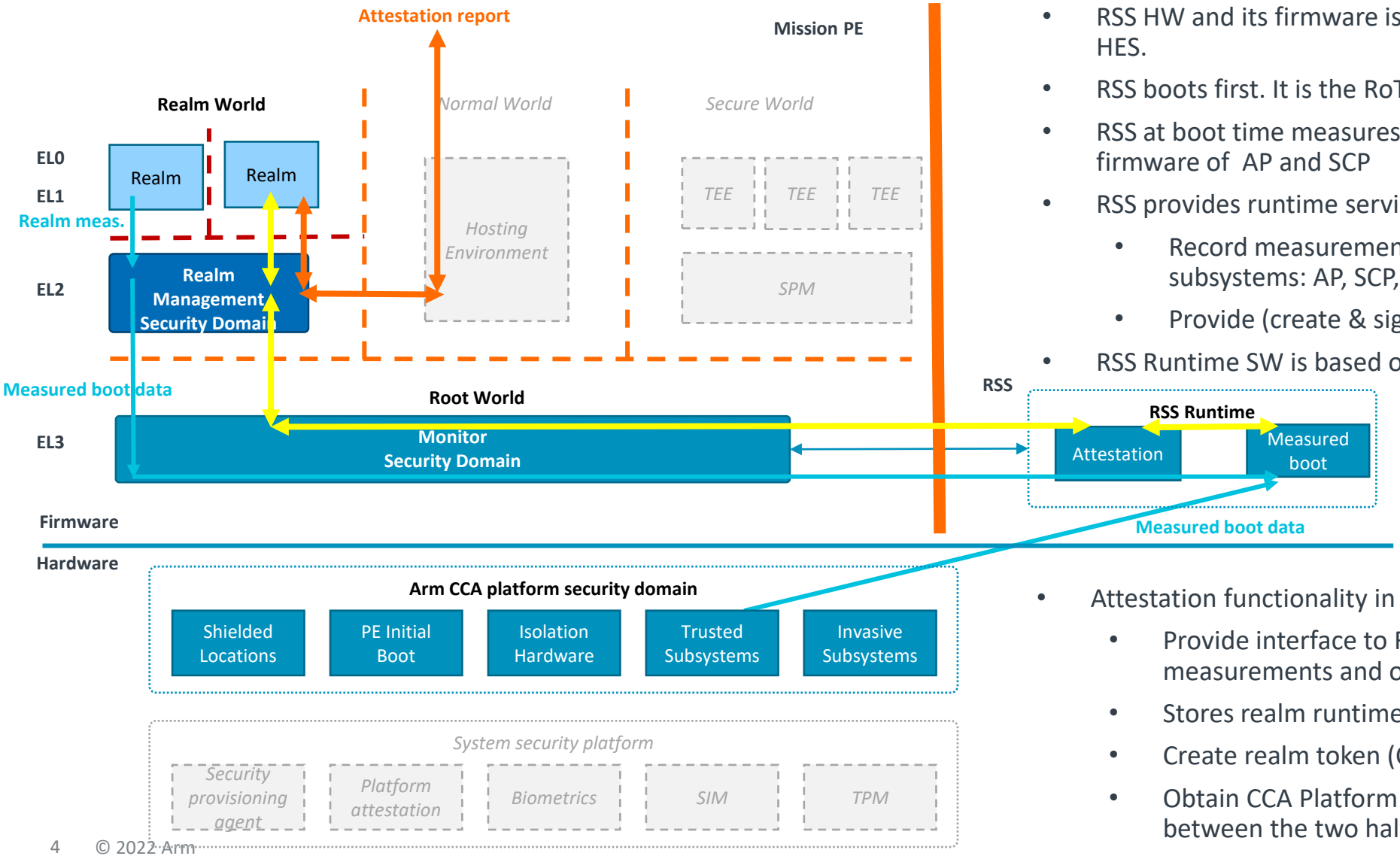
- Attestation report is a bundle of evidence, cryptographically signed by a known key.
- It is used to prove for a Realm user that Realm is running on the top of Arm CCA HW & SW.
- Report verifier can assess system's trustworthiness.
- CCA attestation token consist of two halves, with crypto binding:
  - Arm CCA Platform token
  - Realm token
- Realm attestation is specified by
  - CCA Security Model: https://developer.arm.com/documentation/DEN0096/latest.
  - RMM spec:
    It will be published later this year.

Measured boot on high level



- During boot, the loaded images and additional data (config data) is measured by bootloaders.
- Measurement means in this context to compute hash value. E.g.:
  - BL1 measures BL2: SHA256(BL2)
- Measured boot data:
  - Measurements (hash)
  - Additional data (metadata): signer-id, measurement-algo, sw-version
- So far, in TF-A Measured boot data is propagated upwards: EL3 -> EL0. NS world gets a copy too. Through shared memory.
- RSS is a secure subsystem. It has lower performance than the AP.
- With the introduction of RSS, the measured boot data can be stored by RSS as well.
- These can co-exist.

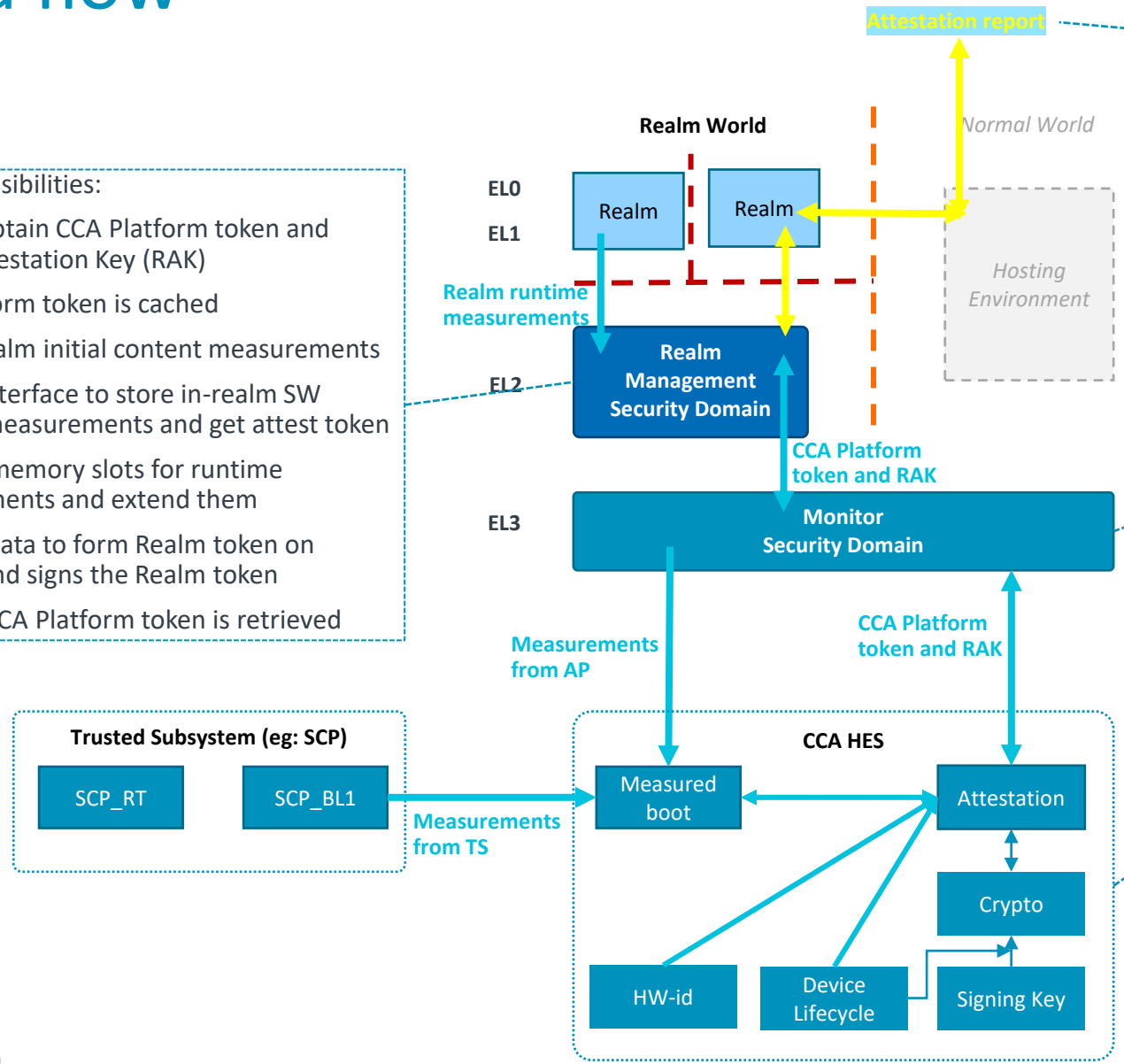
Realm world attestation



- CCA Securtiy Model strongly recommends the presence of a CCA HES component.
- CCA HES is a set of functional requirements and HW isolation.
- RSS HW and its firmware is the reference implementation of CCA HES.
- RSS boots first. It is the RoT.
- RSS at boot time measures and records RSS firmware and initial boot firmware of AP and SCP
- RSS provides runtime service:
  - Record measurements, which are computed by other subsystems: AP, SCP, etc.
  - Provide (create & sign) CCA Platform attestation report.
- RSS Runtime SW is based on TF-M.

- Attestation functionality in RMM:
  - Provide interface to Realm Runtime to record measurements and obtain attestation token.
  - Stores realm runtime + initial content measurement.
  - Create realm token (CBOR + COSE encoding).
  - Obtain CCA Platform token and create crypto binding between the two halves.

Data flow



Attestation request:

- Received through Host OS from relying party (remote entity)
- Routed to corresponding realm

RMM responsibilities:

- At boot obtain CCA Platform token and Realm Attestation Key (RAK)
- CCA Platform token is cached
- Create Realm initial content measurements
- Provide interface to store in-realm SW runtime measurements and get attest token
- Provides memory slots for runtime measurements and extend them
- Encodes data to form Realm token on request and signs the Realm token
- Realm + CCA Platform token is retrieved

Happens at system boot time

Happens at realm boot time

Monitor SD responsibilities:

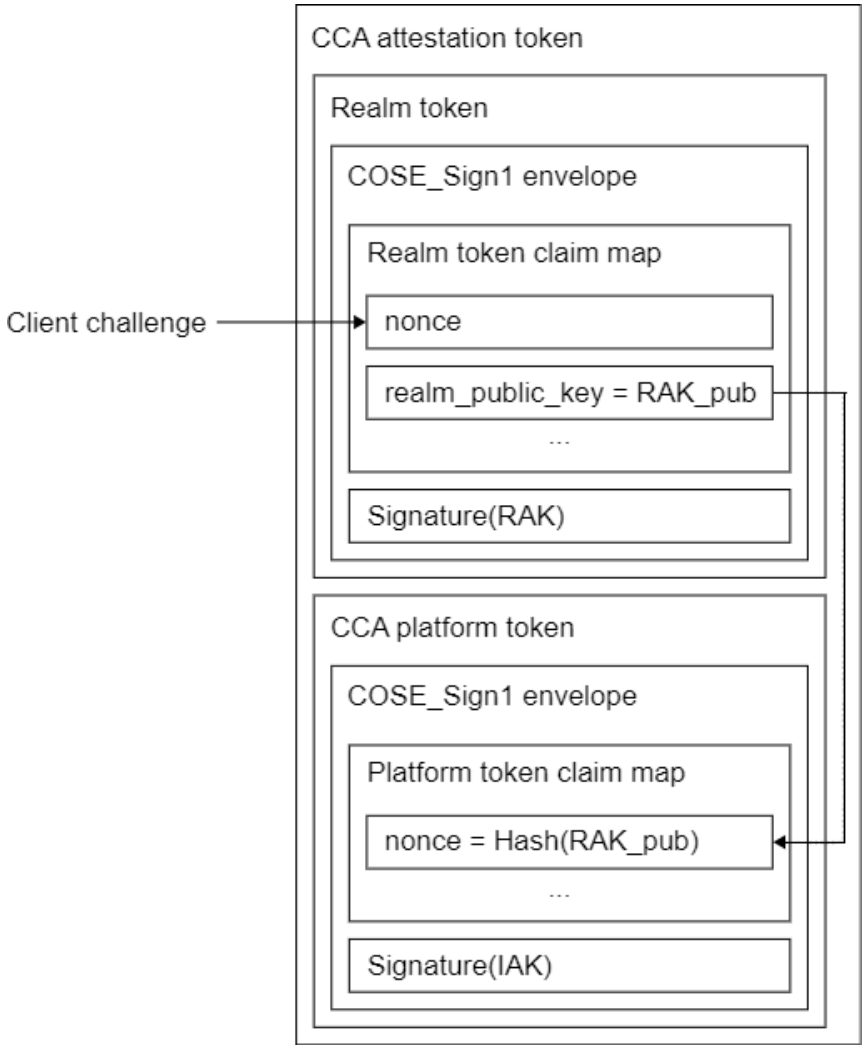
- Mediates b/w RMM ans RSS
- Boot time measurments (EL3, R-EL2) pushed to RSS

CCA HES responsibilities:

- Root of Trust
- Boot first, loads initial SCP and AP images
- Provide measured boot and attestation service
- Owns initial attestation key (IAK) in shielded location.
- Derives Realm Attestation Key (RAK)
- Has exclusive access to other device secrets: lifecycle, HW-id
- Takes input (hash of RAK) from AP to creates a hash binding b/w Realm and CCA Platform token
- Encodes data to form CCA Platform token and signs it

What is in the attestation token?

- Composed of two halves (concatenated):
  - Realm token
  - CCA Platform token
- Realm token:
  - Challenge from relying party
  - Realm measurements
- CCA Platform token:
  - HW config
  - Device Lifecycle
  - Hash(RAK_pub)
  - CCA Firmware measurements
- CCA Platform token is signed by RSS with IAK
- RAK is derived by RSS
- RMM queries RAK and CCA Platform token
- Realm token is signed by RMM with RAK



AP – RSS communication

- Only a few message exchange during boot; No messaging at runtime
- Simple MHU driver is linked to BL1, BL2, BL31
- Boot process is executed by single core. -> Concurrent calls not handled
- Calls have a blocking semantics
- Errors are handled as fatal error
- EL3 uses only a single MHU device pair (1-1 comms). -> No device identifier in API
- Generic MHU API has added:
  - `enum mhu_error_t mhu_send_data(const uint8_t *send_buffer, size_t size);`
  - `enum mhu_error_t mhu_receive_data(uint8_t *receive_buffer, size_t *size);`
- Comms layer is abstracted by PSA API.
- Protocol layer is packed C structures. Based on FF-M spec.
- Internal APIs in EL3 not meant to be exposed to S/NS world. Except to RMM to obtain CCA Platform token and RAK.

Measured boot in RSS

- An alternative measured boot backend to the Event log solution.
- Event log and RSS measured boot can be co-exist on the same platform.
- It is configurable which measurements to store by which backend.
  - BL31 -> Event Log and RSS
  - but RMM -> RSS
- At this point only Realm world TCB is planned to be stored in RSS.
- In the future, S and NS measurements may also be stored in RSS.
- Measurements are stored in RSS internal SRAM, no special HW (e.g.: TPM). In so called „measurement slots”. The number of slots is IMPDEF.
- Measurements are extended:
  - measurement_slot_new_value = SHAxxx (measurement_slot_old_value || new_measurement)
  - || stands for concatenation
- Metadata is stored alongside with measurement:
  - Mandatory: signer-id
  - Optional: SW version, hash algo identifier, SW type (e.g.: AP_BL1, BL31, etc.)

arm

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