arm CCA

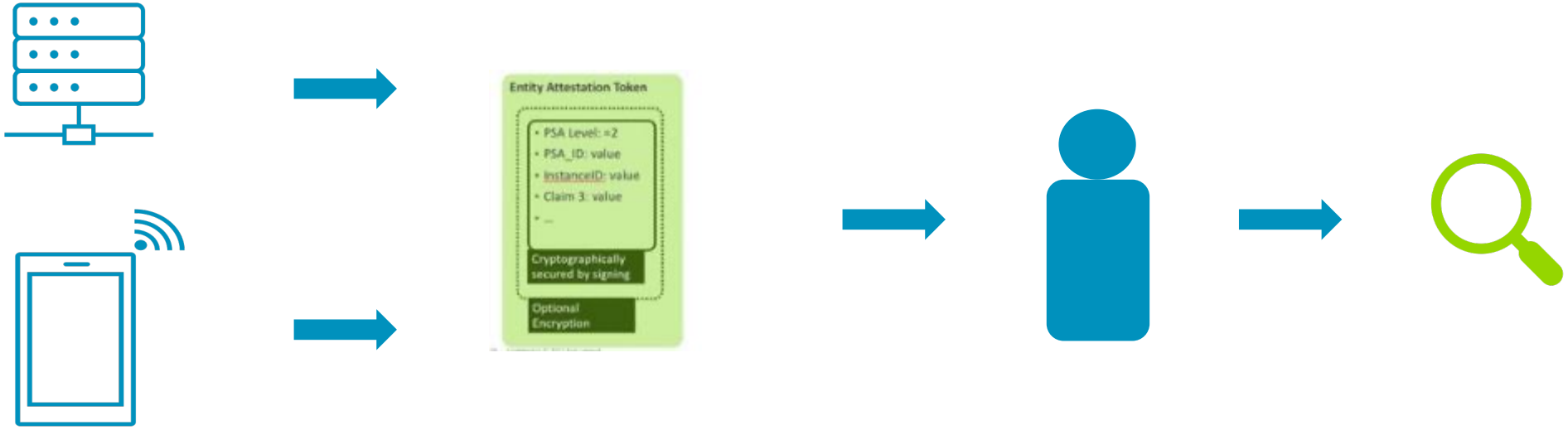
Arm CCA

Attestation

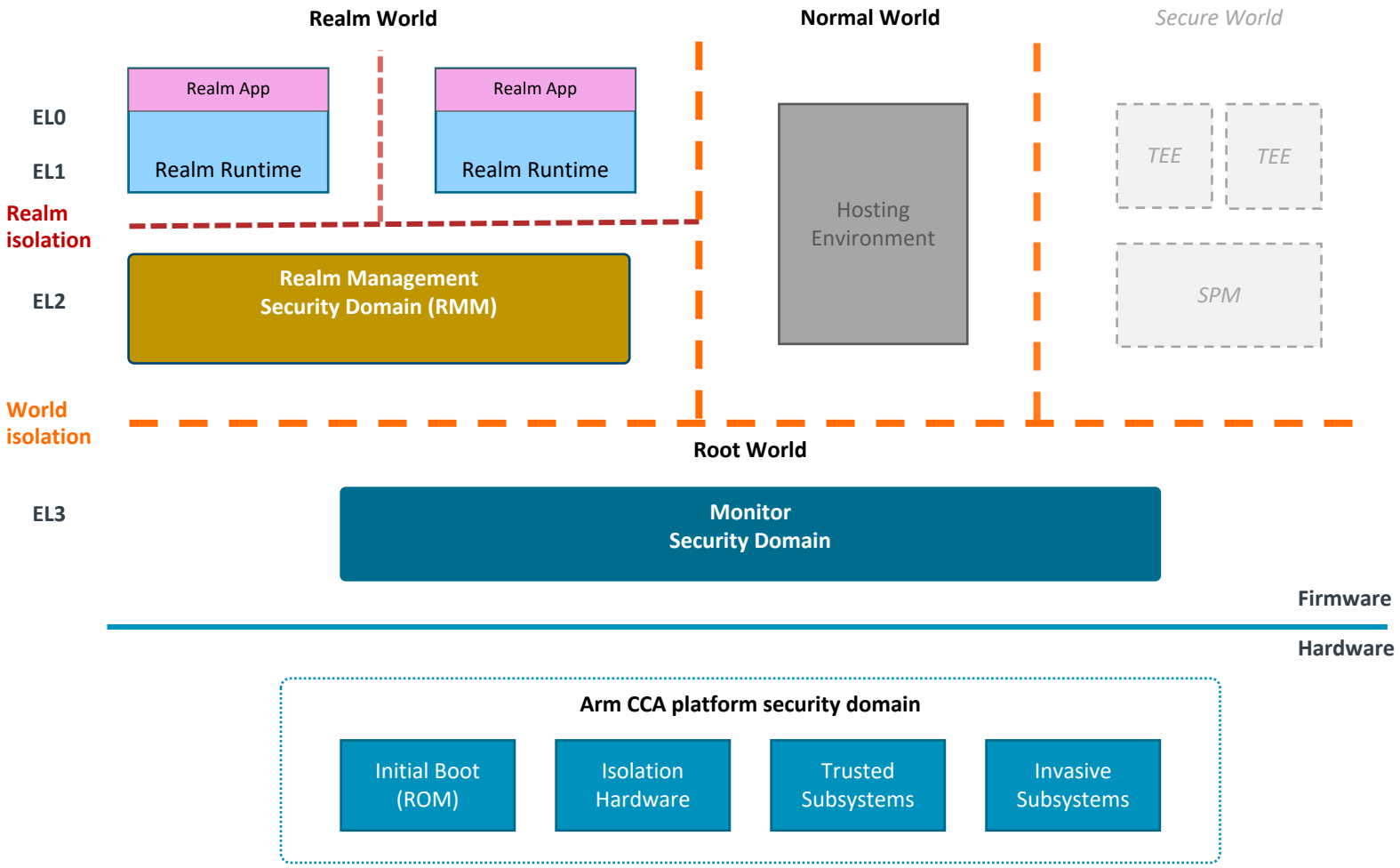
Simon Frost

Attestation

- How does a Realm user confirm that they are using Confidential Compute Architecture?
- They use Attestation - a system where something produces evidence about itself that another party can evaluate the to assess trustworthiness

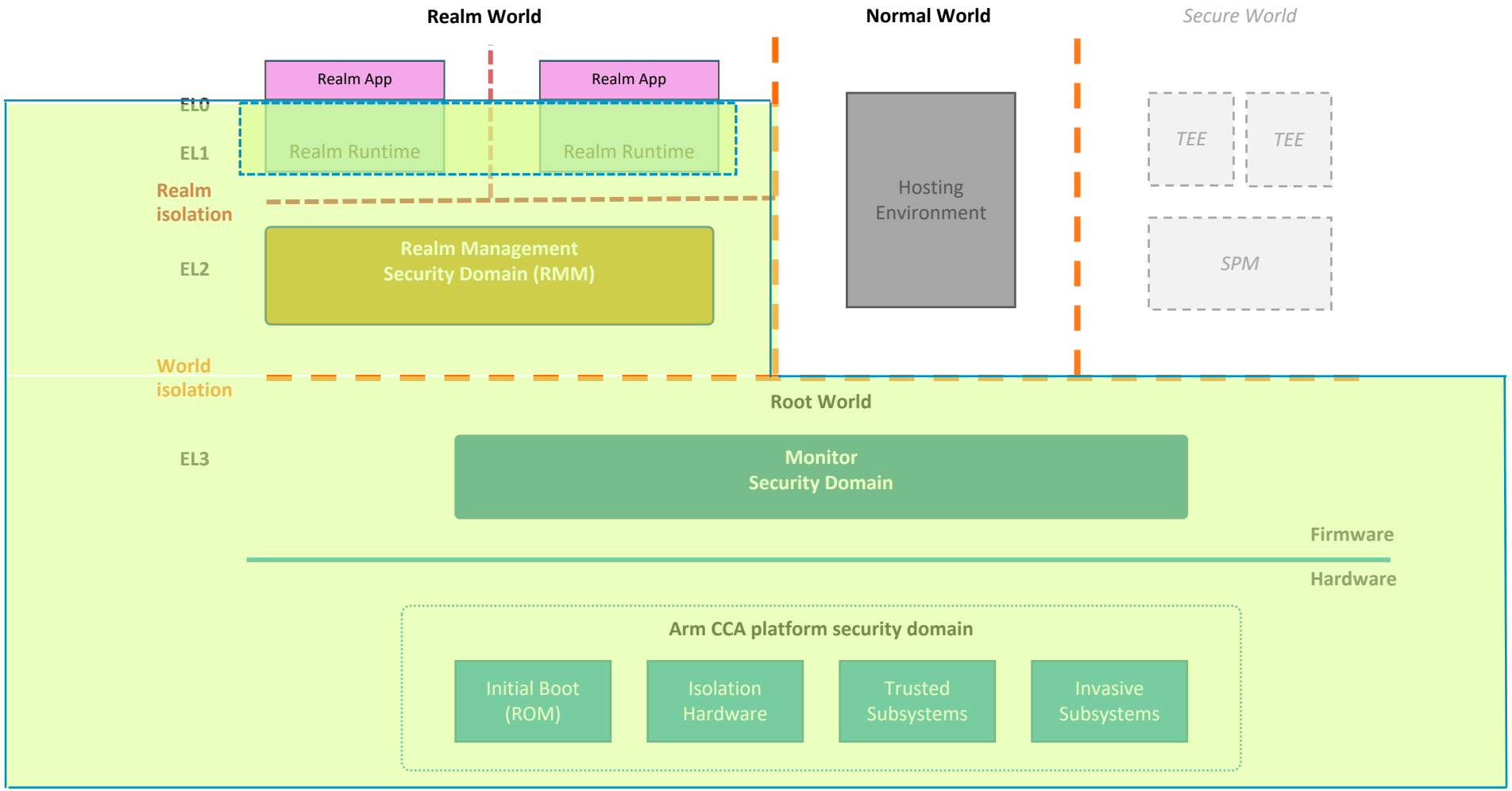


Arm CCA Components



Arm CCA Components

What components do I need to trust?



Attestation Report

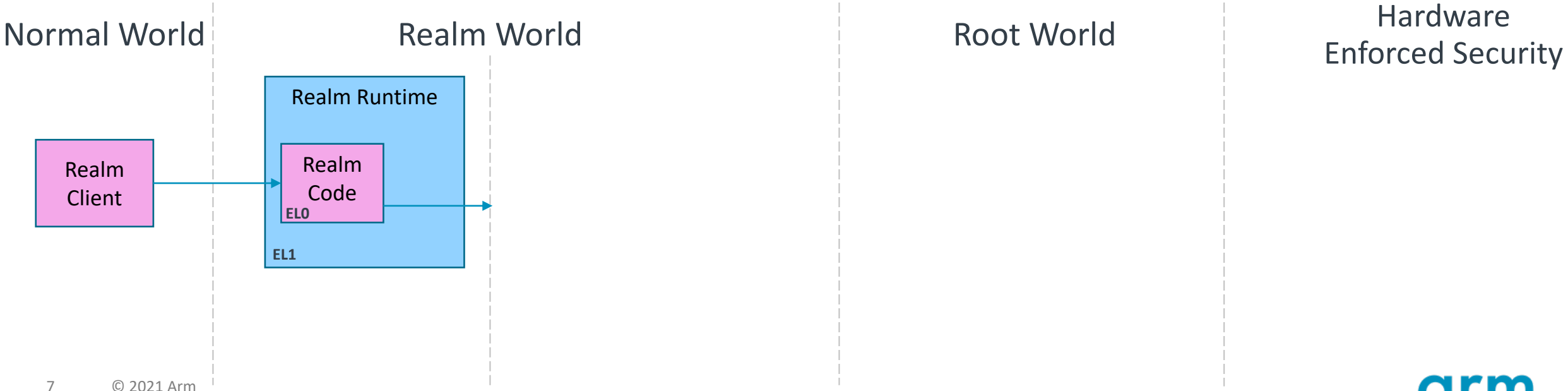
- A Realm can request an Attestation Report for itself at any time during its execution
- The Report is a bundle of Evidence, cryptographically signed by a known key
- The Evidence can be used to prove:
  - This is a genuine processor with CCA (and not an emulator)
  - The firmware for the system and Realm Management comes from the expected supplier
  - That the system is not in a debug state
  - That the software running in the Realm is that expected to be there

What's in the Attestation Report?

- The Report is in two logical sections
- The sections are cryptographically bound together
- The Arm CCA Platform Report
  - Hardware Identity
  - Firmware measurements & metadata
  - Security Lifecycle State
- The Realm Report
  - Measurements of Realm Configuration + Initial Content
  - A set of 'Realm Extensible Measurement' items
    - measurements generated by Realm runtime SW
    - e.g. during UEFI boot
  - A 'challenge', from the caller, used to show freshness
- The Report is encoded in a standard format
  - Arm contributes to the IETF Entity Attestation Token working group

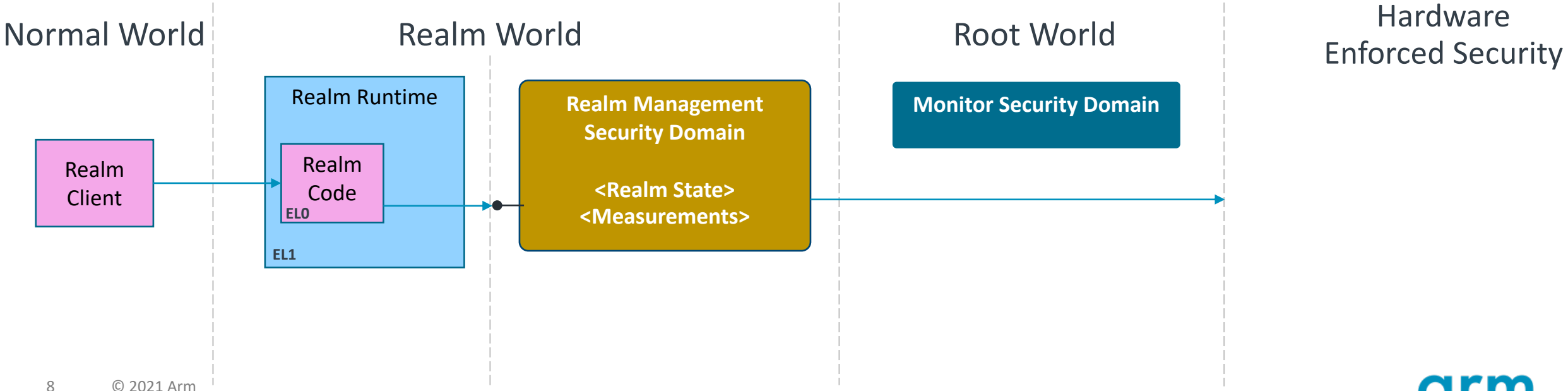
Attestation Report Flow

- Before a client of the Realm software releases any secrets to the Realm, it needs to establish trust
- The client and the Realm software collaborate on the context of a request
  - This will normally include the negotiation of some state that can be used to identify the request
- The Realm software makes an ABI call to the Realm Management Monitor (RMM)
  - Call includes the state negotiated above as a report 'challenge'



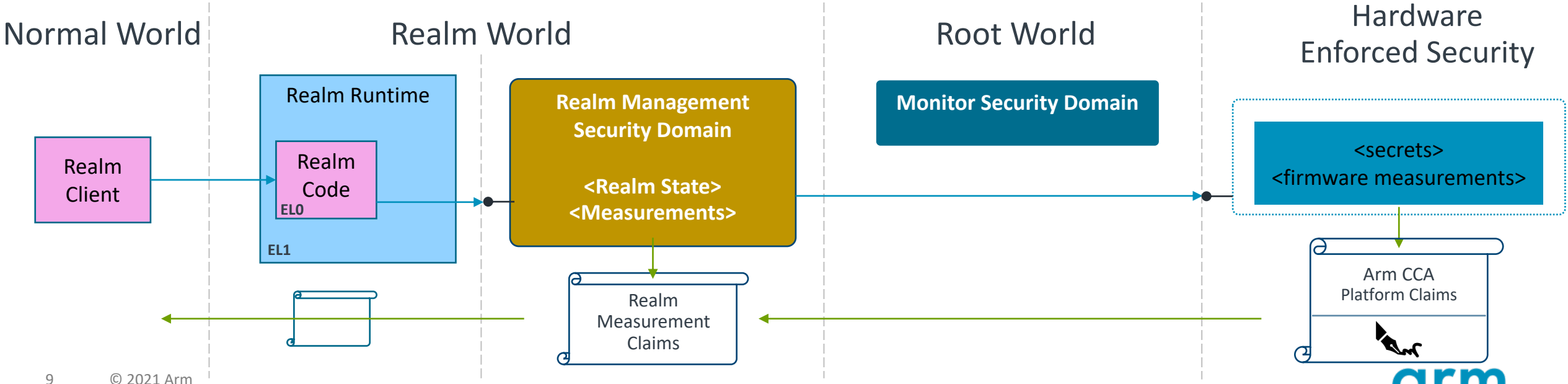
Attestation Report Flow (continued)

- The RMM makes a call into Root Monitor world to request a CCA Platform Report
- This request is routed via the Monitor…
- …to call an ABI exposed by Hardware Enforced Security (HES)
- CCA strongly recommends that Attestation uses Hardware Enforced Security
  - more details on HES in a moment…



Attestation Report Flow (continued)

- Monitor obtains the CCA Platform Report from HES and return it to the RMM
  - HES uses measurements obtained during host boot to populate report
- RMM appends measurements for the Realm to the report and returns it to the Realm
  - Cryptographic binding exists between the Realm and Platform measurements
  - Realm returns the Report to the client
- The evidence values are checked for correctness



Hardware Enforced Security

- Arm CCA strongly recommends the use of Hardware Enforced Security (HES)
- HES is a term covering hardware trusted controllers that can store and process secrets outside of the main AP
  - e.g. secure elements
  - Separation aids the threat model in protecting secrets
- Main AP measured boot evaluates firmware component measurements & metadata
  - sends those values to HES to hold as boot state
- Attestation request routed to HES results in CCA Platform Attestation Report response
- HES signs report using Attestation key derived from a provisioned secret

Evaluating Trustworthiness

- What should the Realm client do with the report?
- Evaluation requires:
  - Checking cryptographic integrity of the report
  - Comparing the values of the evidence in the report against known Reference Values
    - Hardware identities / Firmware measurements from supply chain
  - Checking that the Challenge in the report matches that negotiated with the Realm code
- Sounds tricky?
- Normally the client will send the Report to a trusted Verifier to perform this task
- The Verifier acts on behalf of the SoC Supply Chain to obtain and compare the Reference Values
- Verifiers can be an independent service or pieces of code with provisioned values depending on use case

Building a Verifier Service

- Verifiers sound complicated?!
- Arm is contributing to Project Veraison to help with this problem
- Project Veraison (***VER**ific**A**t**I**on of atte**S**tati**ON**)* is an Open Source project to build the software components required for an Attestation Service
  - Flexible, extensible Verification pipeline with Policy driven appraisal of Attestation evidence
  - Provisioning of Resource Values to link supply chain to verification operations
  - Designed to address industry standards (PSA, EAT, DICE) – not just CCA
  - Project has Open Governance – collaboration is very welcome
  - https://github.com/veraison

VERAISON

Firmware provenance

- Attestation systems can identify that firmware components originate from the supply chain
- Signing checks at boot time can ensure only expected components are loaded
- Can we also check whether the components
  - Have known vulnerabilities
  - Have been based on given source revisions
  - Were actually released by the expected process & not due to an insider attack
  - Were built with up to date tools
- Answering these type of questions needs either a dynamic view of a component state or much deeper understanding of its provenance
- Arm is investigating or collaborating in Open Source projects such as Trillian Firmware Transparency / SigStore / In-Toto to understand how provenance can be improved

arm

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