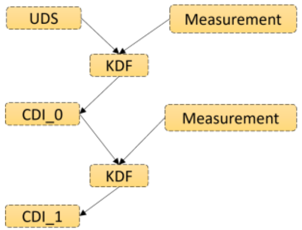
Linaro Connect
MADRID 2024 | MAY 14-17 2024

Enabling mobile trust thanks to DICE/DPE in Android

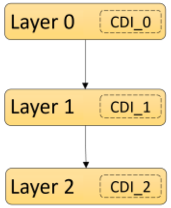
What is DICE and DPE?

- DICE and DICE Layering are concepts that help create attestation schemes and other system features:
  - Defined by Trusted Computing Group (TCG): Device Identifier Composition Engine
  - The device is provisioned with a Unique Device Secret (UDS).
  - At boot time all loaded components are measured and measurements recorded.
  - The combination of UDS and the first measurement derives a Compound Device Identifier (CDI) value.
  - Later with the combination of CDI and measurements further CDIs are derived.
  - SW components can be grouped. A group of components are contributing to single CDI value.
  - From the CDI further keys (sym. and asym.) can be derived. These can be used for attestation and sealing.
  - A group is represented by a certificate. It is signed by a key derived from its CDI.
- What is DPE:
  - Another TCG spec (driven by Google) : DICE Protection Environment
  - DPE is a specification for an isolated enclave used to store and manage DICE secrets, perform DICE derivations and sign attestation certificates.
  - It defines the HW and the SW requirements to make DICE computation in a secure, isolated environment.
  - Server-client architecture, where all bootloader components are a client of the entity that executes the DPE service.

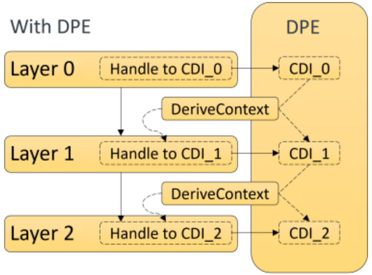
DICE computation



Without DPE



With DPE



Why DICE and DPE are important?

- Google is interested in a DICE based attestation scheme in Android:
  - The sequence of certificate creates the DICE Certificate Chain . It could represent all the SW components of a system from immutable bootloader up to user space.
  - Data can be bound to a given version of a SW component with seal and unseal operations. Same or greater version of the SW component can unseal (decrypt) the previously sealed (encrypted) data.
- A SW only implementation already exists:
  - Generic and Android specific DICE library: Open-dice repo
  - The chain is starting (ideally) from the immutable bootloader.
  - All bootloader stages do DICE computation and create a certificate. DICE data is handed off to the next stage which will extend the data coming from the previous stage and so on.
  - Available on Pixel 6 and presumably onwards.
  - No end-to-end open-source reference (BL1 - pVM user space) implementation is available.
- The goal is to extend the Android DICE certificate chain to pVMs for new attestation use cases, e.g.: provisioning secrets to a pVM.
- DPE provides extra security guarantees compared to the SW only DICE implementation.

Where to run the DPE service?

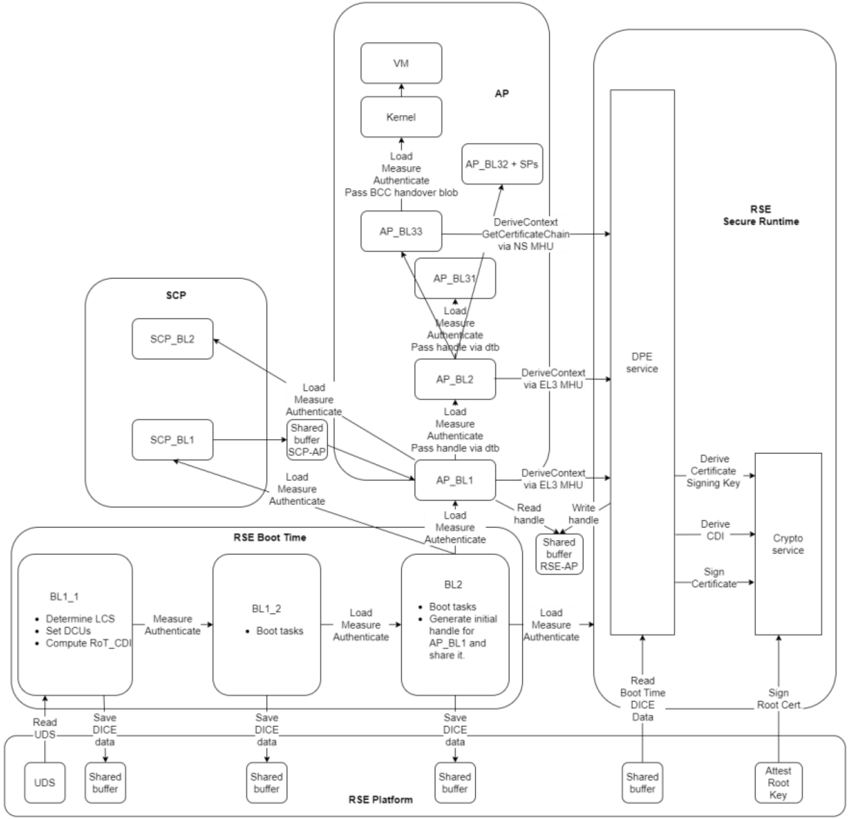
- The DPE spec does not have hard requirements on the exact implementation details of the enclave or secure environment that comprises a DPE.
- The Runtime Security Engine (RSE) is an isolated execution environment which can provide security guarantees for DICE secrets.
- RSE capabilities:
  - On-chip secure enclave
  - Act as the Root of Trust: Secure boot, loading components
  - Provide runtime services
  - M-class based
  - MHU interface towards the rest of the system
  - Crypto acceleration;
  - Side channel and fault injection protection
  - Exclusive SRAM; OTP; ROM code
  - Access all system memory

DPE support in RSE

- The DPE implementation is based on the combination of DPE spec (revision 0.9) and Open DICE profile.
- Open DICE is followed (like a DPE profile) where the DPE spec leaves details unspecified.
- The DPE service is kept simple:
  - Reduced command support: DeriveContex(), GetCertificateChain(), CertifyKey(), DestroyContext()
  - No dynamic memory allocation
  - A single, plain session is supported.
- DPE commands are CBOR encoded.
- SW components can be grouped to layers with a custom argument (cert-id) of DeriveContext.
- A certificate chain, rooted in the ROM code of RSE, is produced which represents all the components recorded by DeriveContext command.
- Focused so far on attestation use case, sealing is planned.
- Developed and tested on the TC2 platform, which includes RSE (part of TF-M project).
- DPE implementation is available here.

DPE integration with boot time components

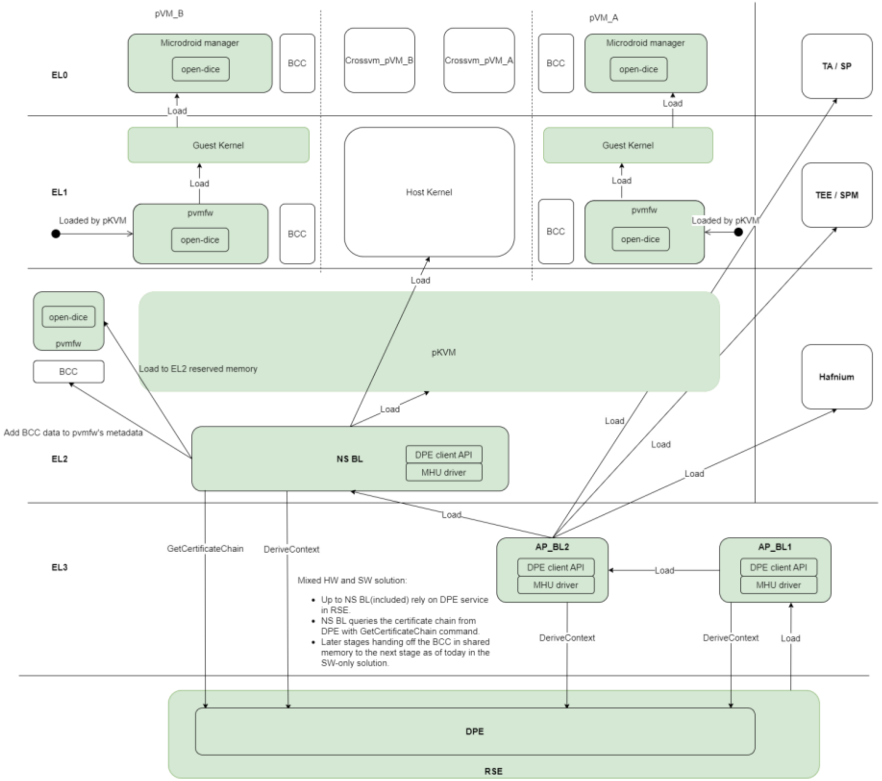
- RSE executes the runtime DPE service.
- Early bootloader measurements of RSE are kept in SRAM and processed later by DPE at service init.
- All bootloader stages (up to NS BL) have their own DPE client library and MHU driver. They send CBOR encoded commands.
- All SW components (config data could be as well: dtb) are measured and recorded to RSE.
- RSE does the DICE computation and certificate creation.
- Context handles are handed off through shared memory between boot stages.
- The boot flow is single threaded with blocking calls.
- Plan to showcase this in the TC23 release in July.
- Integration with TF-A available here.



Hybrid solution

- A full solution (DPE integration up to pVM’s user space) is a lot of work. Responsibility of affected components is spread among many project. A massive collaboration is required in the ecosystem to pull it off.
- The hybrid solution is meant to mixes the HW-backed and the SW-only solution to produce a single DICE certificate chain.
- RSE FW, TF-A and NS bootloader rely on DPE service in RSE (HW backed part).
- NS bootloader queries the certificate chain and the last CDI values.
- NS bootloader creates an Android DICE handover blob and adds it to the pvmfw’s configuration data.
- This enables the Android SW stack, which will rely on SW-only crypto as of today, to extend the exported certificate chain to produce a single certificate chain. It covers the TCB from the immutable bootloader up to the user space.
- Benefits:
  - UDS is provisioned to an OTP memory accessible only by RSE. AP has no access to the root secret.
  - Avoids the implementation complexity and risks around Linux/pKVM/pvmfw/user space integration.
  - Extends the Android DICE certificate chain to the entire SW stack.
  - The SW-only solution can be gradually upgraded to rely on the DPE service in RSE instead.
- Drawbacks:
  - Not an ideal solution from a security point of view.
- **The goal is to showcase DPE and enable partners to build upon it.**
- However, Linaro is working on adding DICE support to TF-A on platforms without a DPE-capable execution environment.

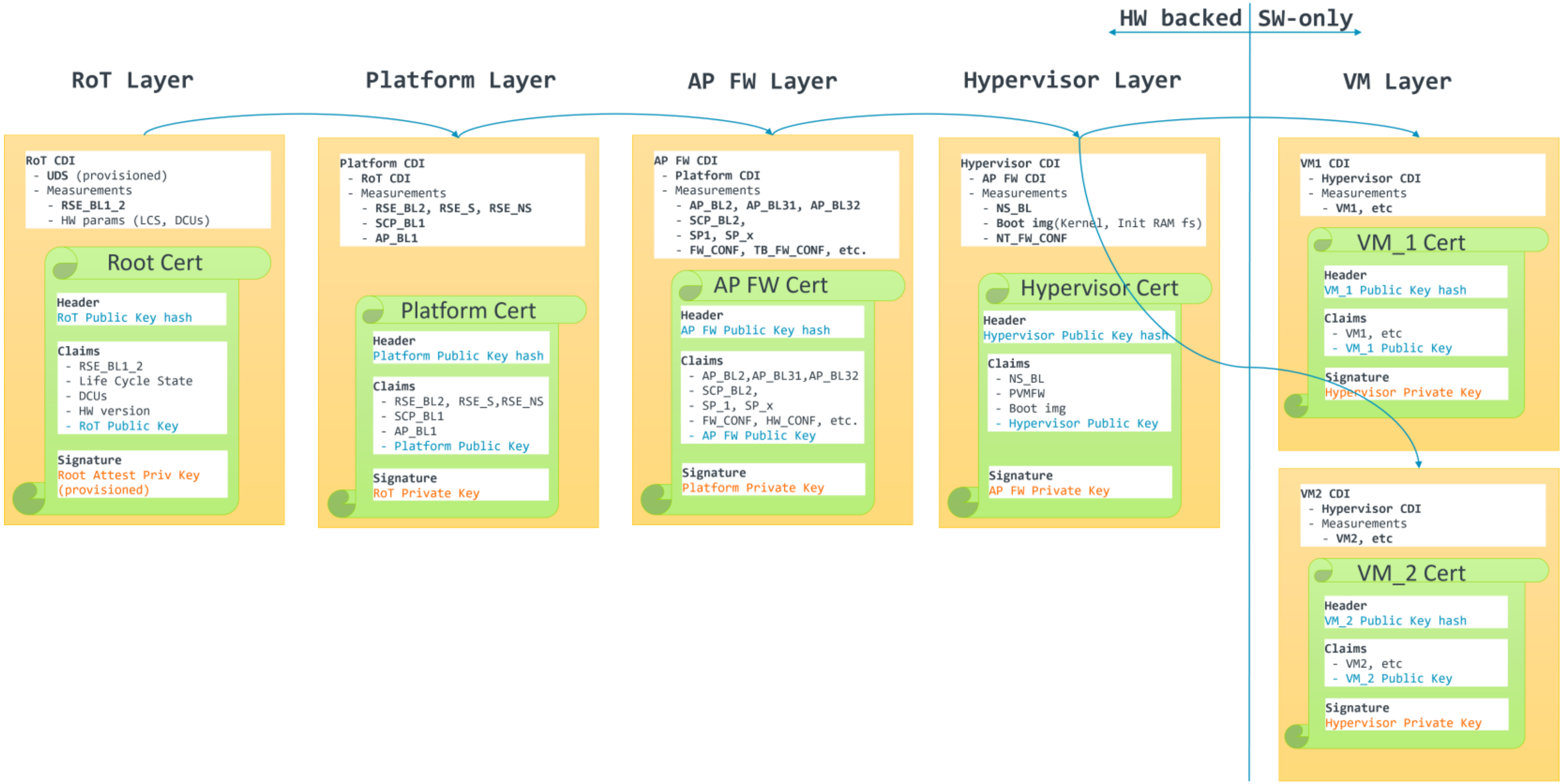
Hybrid solution



Mixed HW and SW solution:

- Up to NS BL(included) rely on DPE service in RSE.
- NS BL queries the certificate chain from DPE with GetCertificateChain command.
- Later stages handing off the BCC in shared memory to the next stage as of today in the SW-only solution.

DICE Certificate Chain on TC platform



HW backed | SW-only

RoT Layer

RoT CDI
- **UDS** (provisioned)
- Measurements
  - **RSE_BL1_2**
  - HW params (LCS, DCUs)

Root Cert

Header
RoT Public Key hash

Claims
- RSE_BL1_2
- Life Cycle State
- DCUs
- HW version
- RoT Public Key

Signature
Root Attest Priv Key (provisioned)

Platform Layer

Platform CDI
- **RoT CDI**
- Measurements
  - **RSE_BL2, RSE_S, RSE_NS**
  - **SCP_BL1**
  - **AP_BL1**

Platform Cert

Header
Platform Public Key hash

Claims
- RSE_BL2, RSE_S,RSE_NS
- SCP_BL1
- AP_BL1
- Platform Public Key

Signature
RoT Private Key

AP FW Layer

AP FW CDI
- **Platform CDI**
- Measurements
  - **AP_BL2, AP_BL31, AP_BL32**
  - **SCP_BL2,**
  - **SP1, SP_x**
  - **FW_CONF, TB_FW_CONF, etc.**

AP FW Cert

Header
AP FW Public Key hash

Claims
- AP_BL2,AP_BL31,AP_BL32
- SCP_BL2,
- SP_1, SP_x
- FW_CONF, HW_CONF, etc.
- AP FW Public Key

Signature
Platform Private Key

Hypervisor Layer

Hypervisor CDI
- **AP FW CDI**
- Measurements
  - **NS_BL**
  - **Boot img**(Kernel, Init RAM fs)
  - **NT_FW_CONF**

Hypervisor Cert

Header
Hypervisor Public Key hash

Claims
- NS_BL
- PVMFW
- Boot img
- Hypervisor Public Key

Signature
AP FW Private Key

VM Layer

VM1 CDI
- **Hypervisor CDI**
- Measurements
  - **VM1, etc**

VM_1 Cert

Header
VM_1 Public Key hash

Claims
- VM1, etc
- VM_1 Public Key

Signature
Hypervisor Private Key

VM2 CDI
- **Hypervisor CDI**
- Measurements
  - **VM2, etc**

VM_2 Cert

Header
VM_2 Public Key hash

Claims
- VM2, etc
- VM_2 Public Key

Signature
Hypervisor Private Key

Linaro Connect
MADRID 2024 | MAY 12-17 2024

Thank you