CCC Attestation WG

Device Identity Composition Engine (DICE)

Ned Smith - Intel Corp.

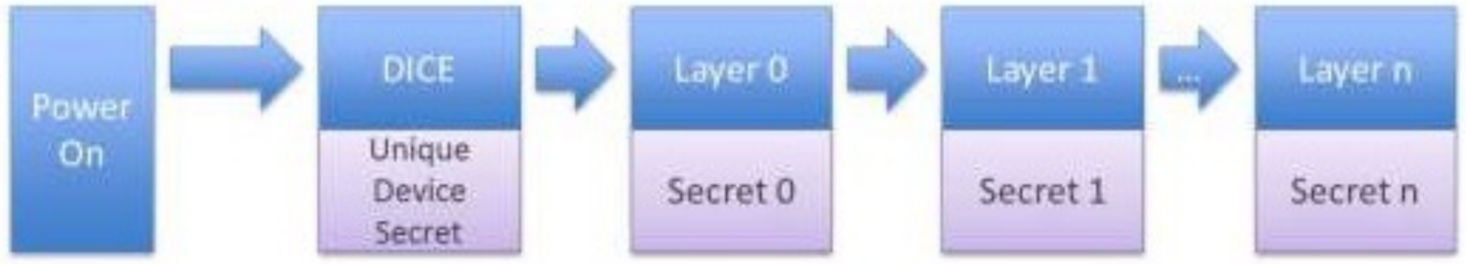
2022-11-22

DICE

- DICE: **D**evice **I**dentifier **C**omposition **E**ngine, foundation of DICE Architecture
- TCG DICE Architecture WG published the the first DICE spec in 2018
- Goals:
    - Architecture for attestation, device identity, and measured/verified boot
    - Root of Trust for Measurement building blocks
    - Targets a spectrum of environments: MCUs, SoCs, MCPs, IP Blocks and FPGAs
    - Hardening of DICE functionality
    - Alignment with broader/emerging industry attestation and device identity standards
- "DICE" may refer to both a RoT "engine" or an architecture for measured or verified boot in environments where TPM isn't possible
    - DICE Protection Environment (DPM) is an API that modularizes much of the DICE functionality
- DICE vs. TPM
    - DICE reduces RoT footprint whereas TPM is often too costly:
        - Real estate, power, functionality, storage often exceeds total budget for constrained applications
    - TPM isn't an RTM

DICE in a Nutshell

- Organizes bootstrap into layers where each layer can generate secrets used to attest the next layer.
  - A root of trust containing a Unique Device Secret (UDS) ensures subsequent layers have an entropy source



- **Typically**:
  - Power on unconditionally starts the DICE Engine
  - Each layer computes the secret for the next layer using a one-way function and a measurement of the next layer
  - Each layer is trusted to keep the secret it receives from the previous layer confidential
  - Secret derivation is such that a change to the layer TCB generates a different attestation key (implicit attestation)
    - If a patch/update is applied, a new seed is generated effectively re-keying the layer (device) key

What are DICE Benefits?

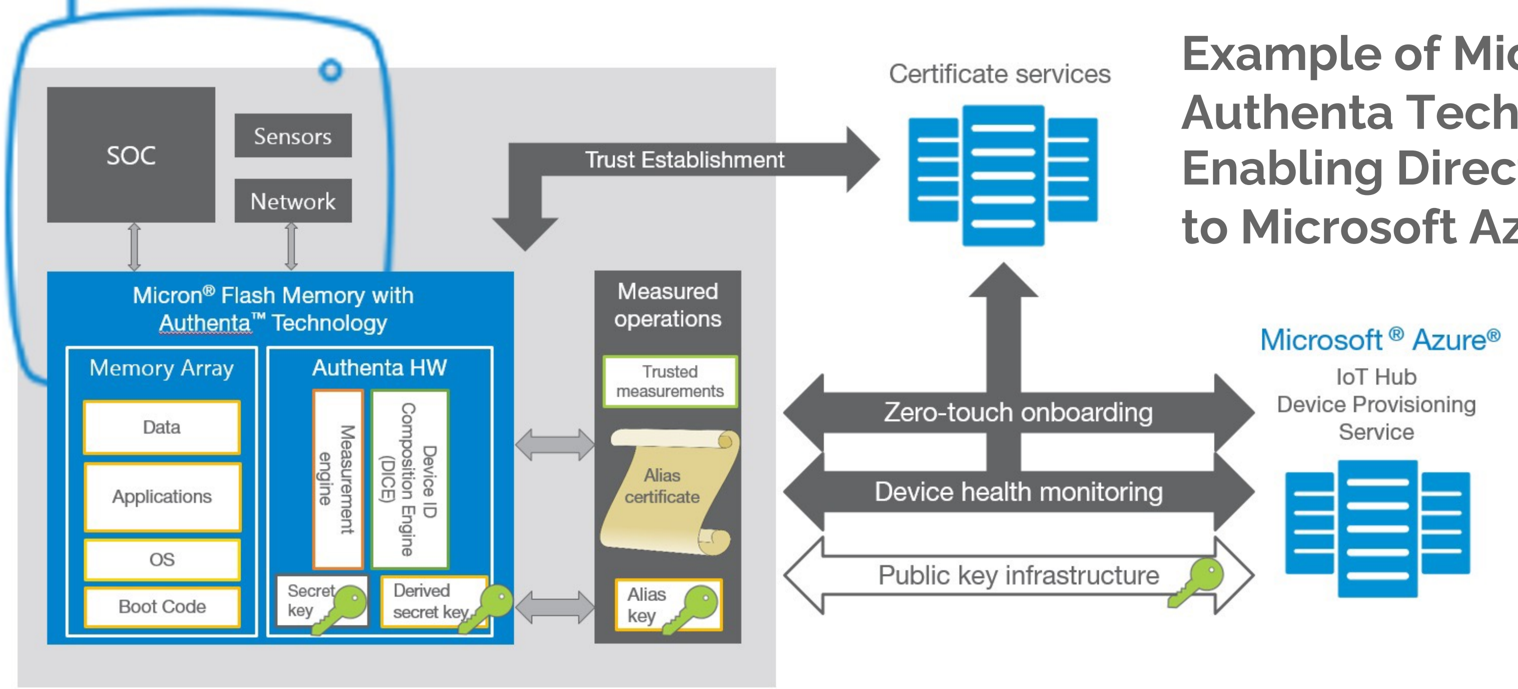
- Small RoT footprint, simple, flexible primitive
- Strong device identity
- Richly contextual measurements
  - Supports verified boot across SW update
- Integrated certificate chains
- Implicit attestation
- Self-healing
- No inherent limitations on component structure
- No requirement for durable storage of secrets or keys other that UDS
- Industry standard attestation and device identity

DICE Ecosystem

- Microchip, Micron, NXP, Microsoft Research, Intel

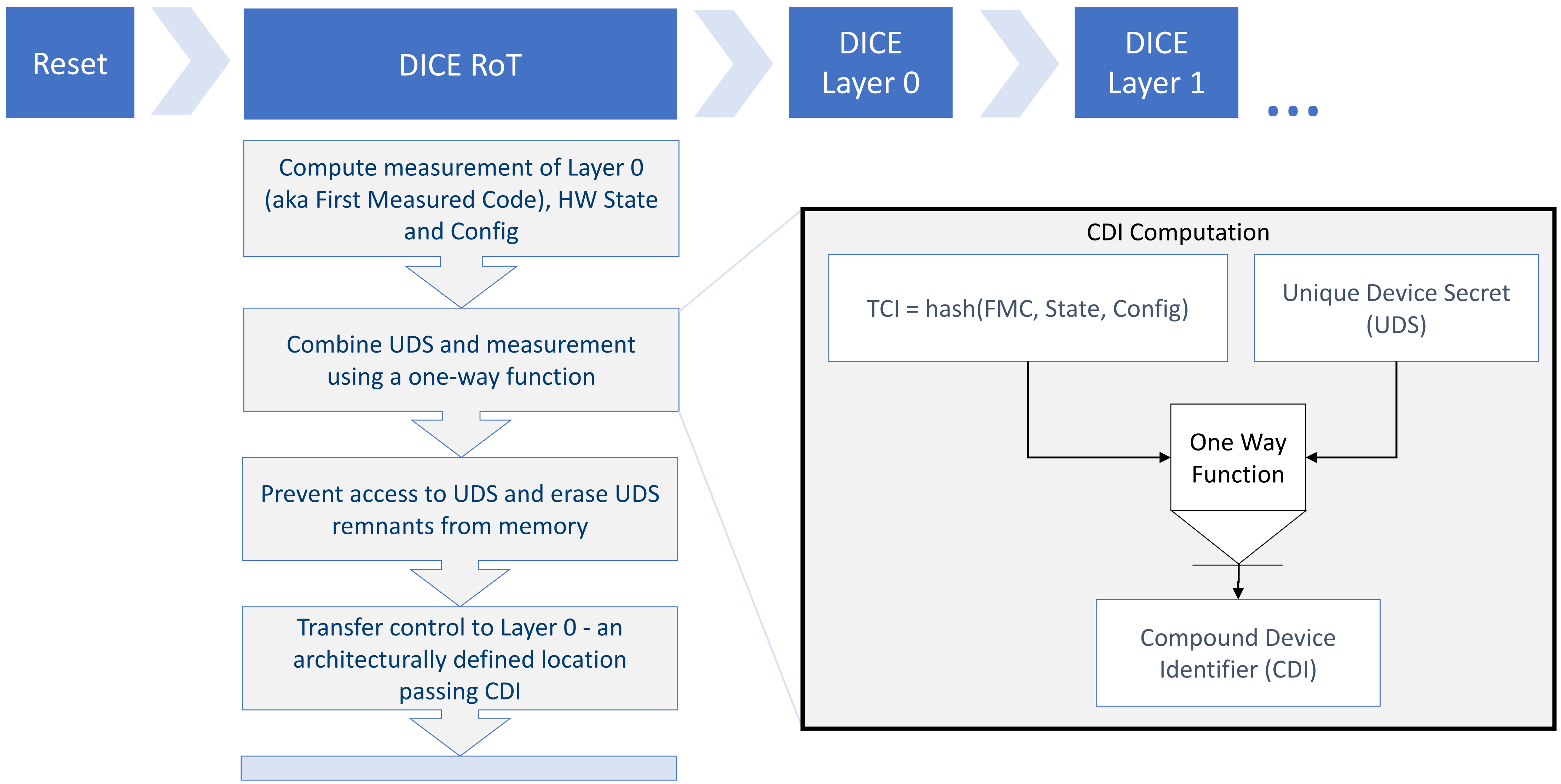
The CEC1702 is a programmable 32-bit microcontroller.

NXP MX RT600 supports DICE

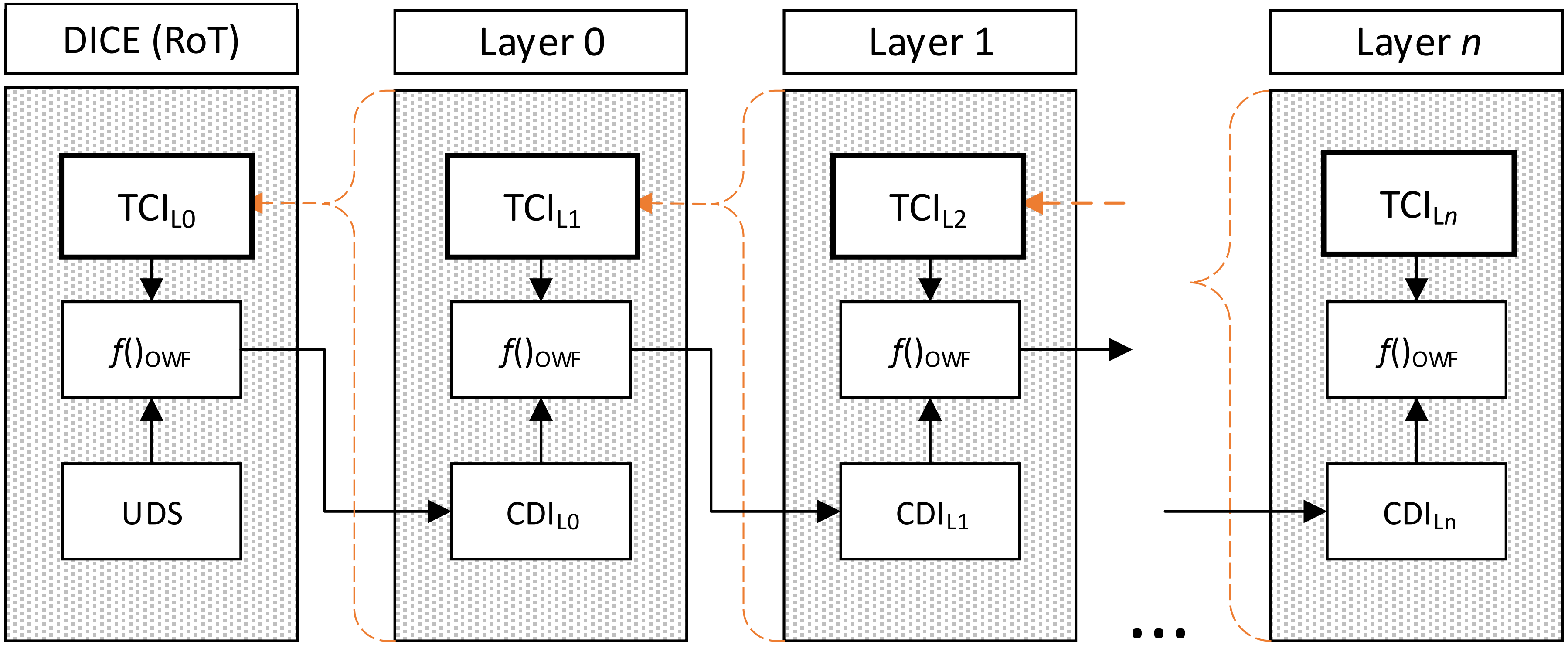


Example of Micron's Authenta Technology Enabling Direct Connection to Microsoft Azure Cloud

DICE Root-of-Trust



Layering Architecture



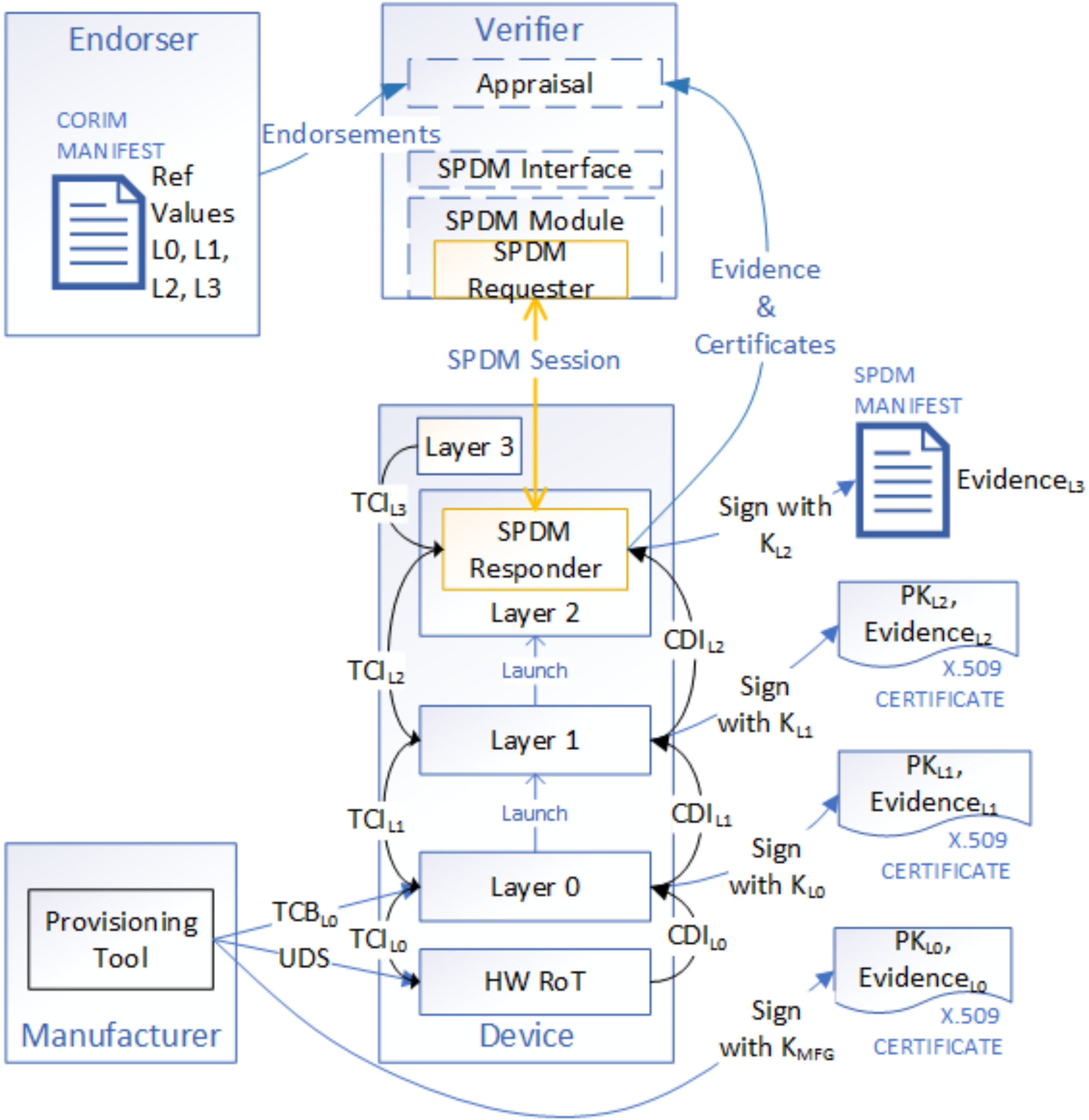
- Consecutive DICE layer CDI values depend on previous layer CDI values.
  - Trust dependencies are explicitly captured in CDI values
  - DICE layering architecture doesn't constrain the number of TCI or CDI values per layer
  - One-way functions can be FIPS140-3 compliant
  - Implementation architecture protects secrets and trusted functionality

DICE Architecture is Standards Aligned

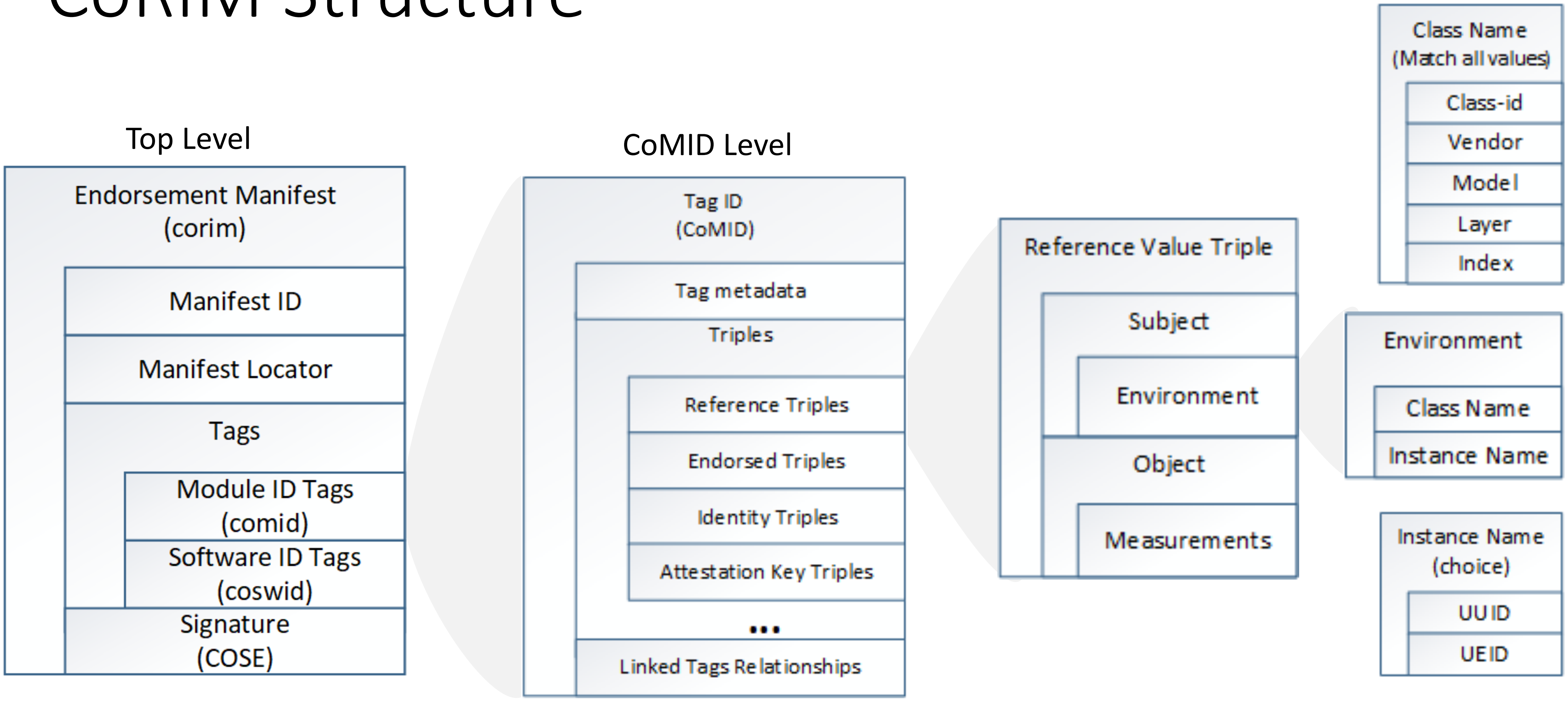
- Evidence Standards
  - TCG ‘tcbinfo’ – X.509 extension, evidence is encoded as ASN.1/BER
  - TCG ‘TaggedEvidence’ – X.509 extension, evidence is tagged CBOR
  - TCG / DMTF ‘concise-evidence’ – CDDL schema that builds on CoRIM schema
  - IETF draft-ftbs-rats-msg-wrap – Evidence is tagged using a media-type, coap-content-format or CBOR tag and serialized in the format defined by the tag.
    - Note: classic SGX attestation payloads can be CBOR tagged and encoded as a ‘bstr’
  - IETF ‘EAT’ profiles for encoding Evidence as
    - CWT (RFC8392) or
    - JWT (RFC7519)
- Endorsement Standards
  - TCG / IETF Concise Reference Integrity Manifest (CoRIM)
  - TCG ‘Manifest’ – X.509 extension, conveying signed manifests
    - SWID (XML),
    - CoSWID (CBOR, JSON)
    - CoRIM (CBOR)

DICE Integration with SPDM and CoRIM

- Device manufacturer provisions DICE RoT, Layer 0 key and issues Device Identity certificate
- Layer 0 measures Layer 1, generates L1 seed, CDI, key and issues L1 certificate
- Repeats for as many layers
- SPDM may be part of the TCB for some layer. SPDM might measure additional components.
- Both DICE layer Evidence (in certs) and SPDM Evidence are reported via SPDM connection.

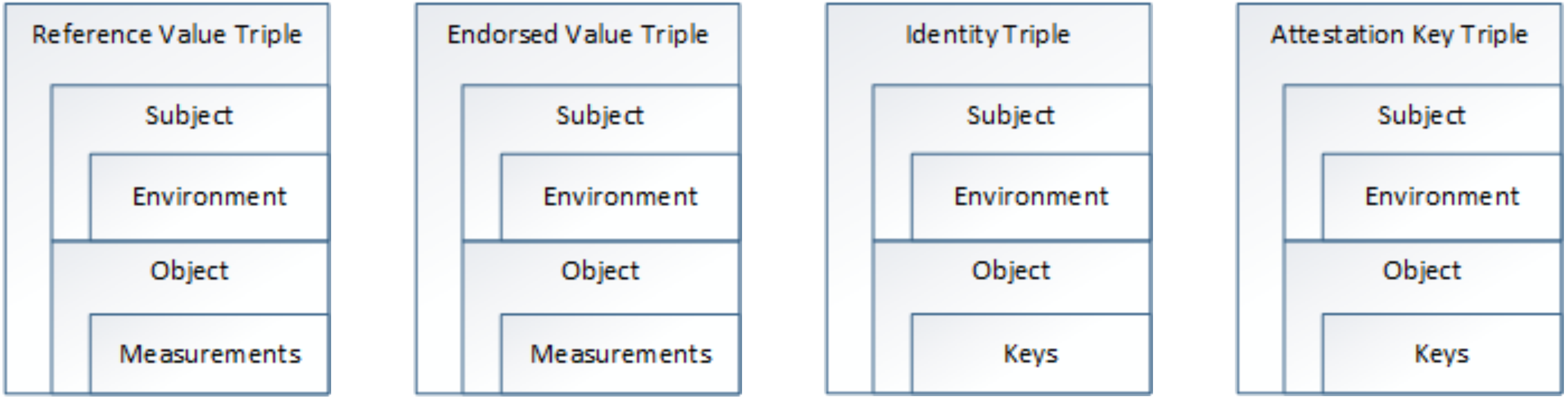


CoRIM Structure

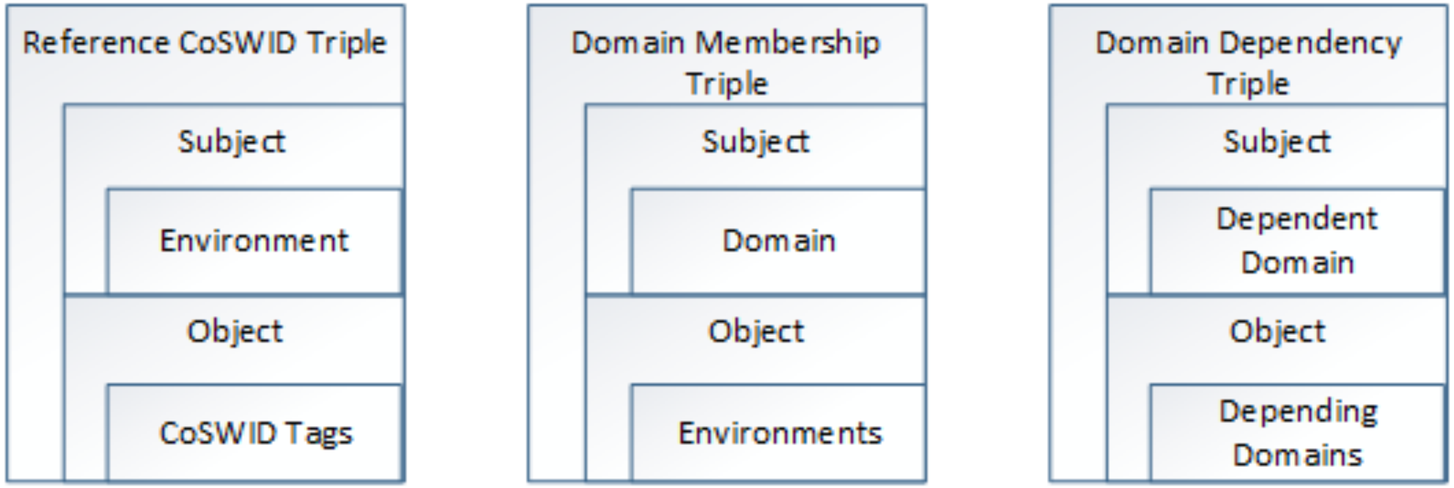


CoRIM Structure Cont. - Triples

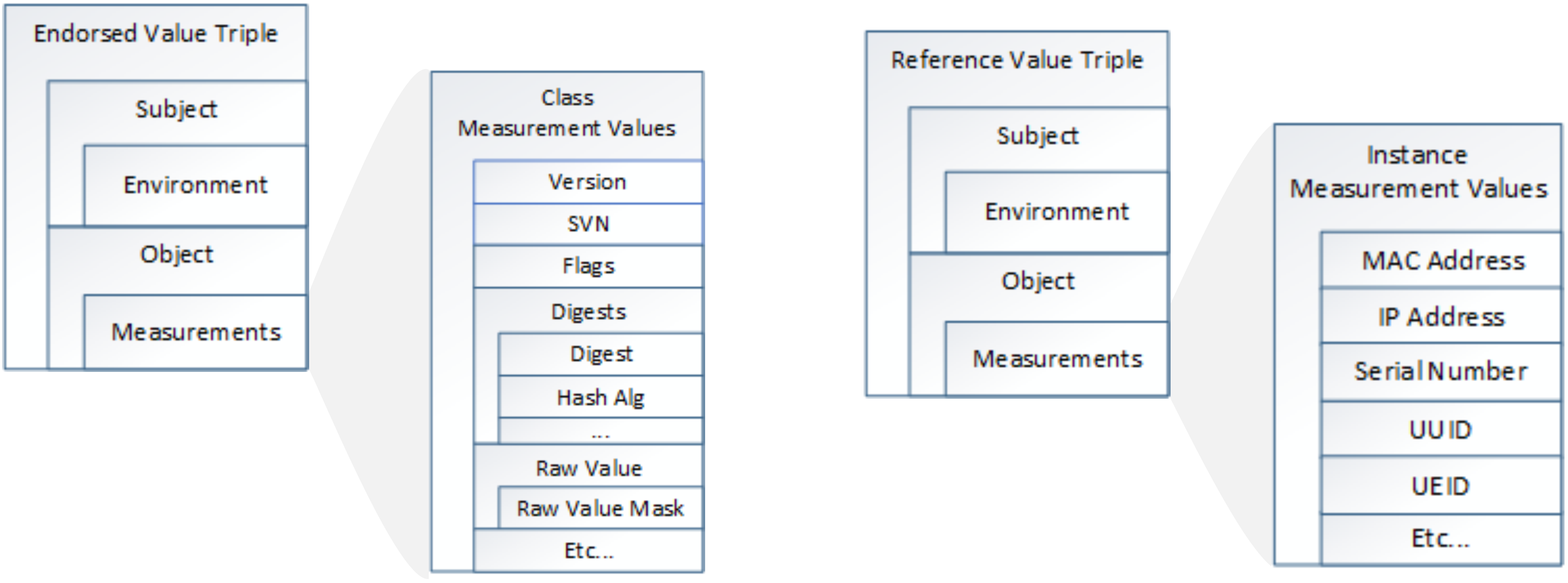
Base CoRIM Schema



Extended CoRIM Schema

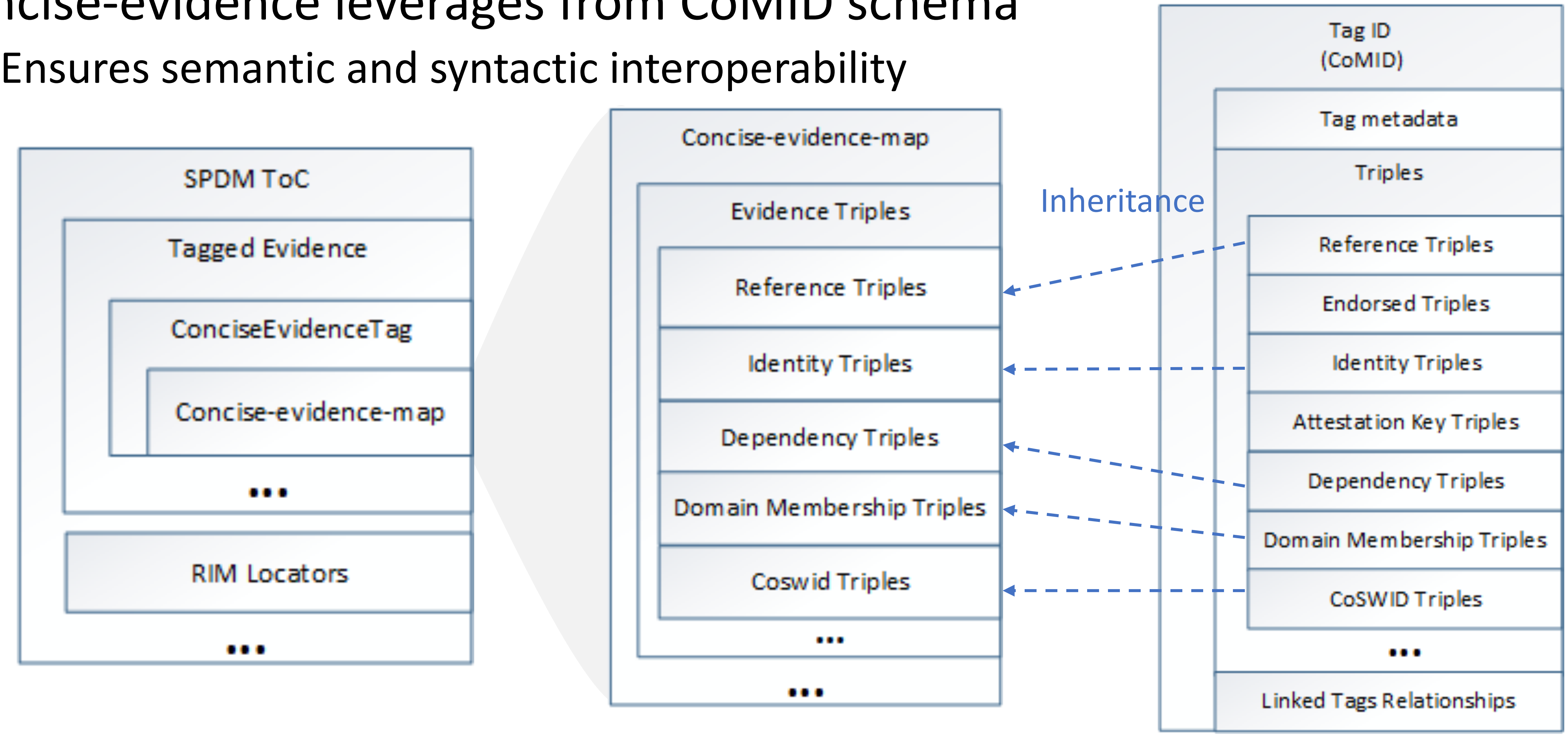


CoRIM Structure Cont. - Measurements



Concise Evidence

- Concise-evidence leverages from CoMID schema
  - Ensures semantic and syntactic interoperability

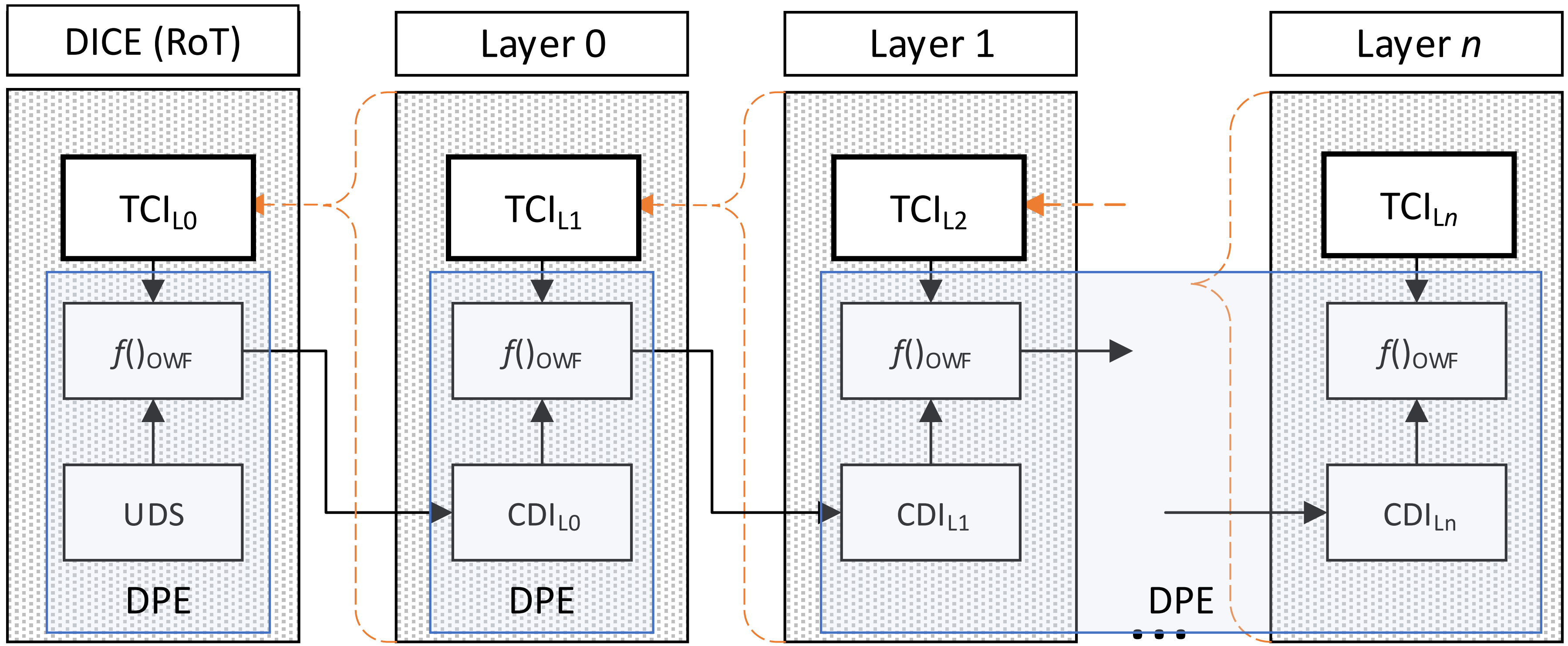


DICE Protection Environment (DPE) Primer

DICE has Implementation Challenges

- CDI handling / protection
- Performance
- Interoperability & Consistency
- Implementation Diversity
- Sealing
- Simulation

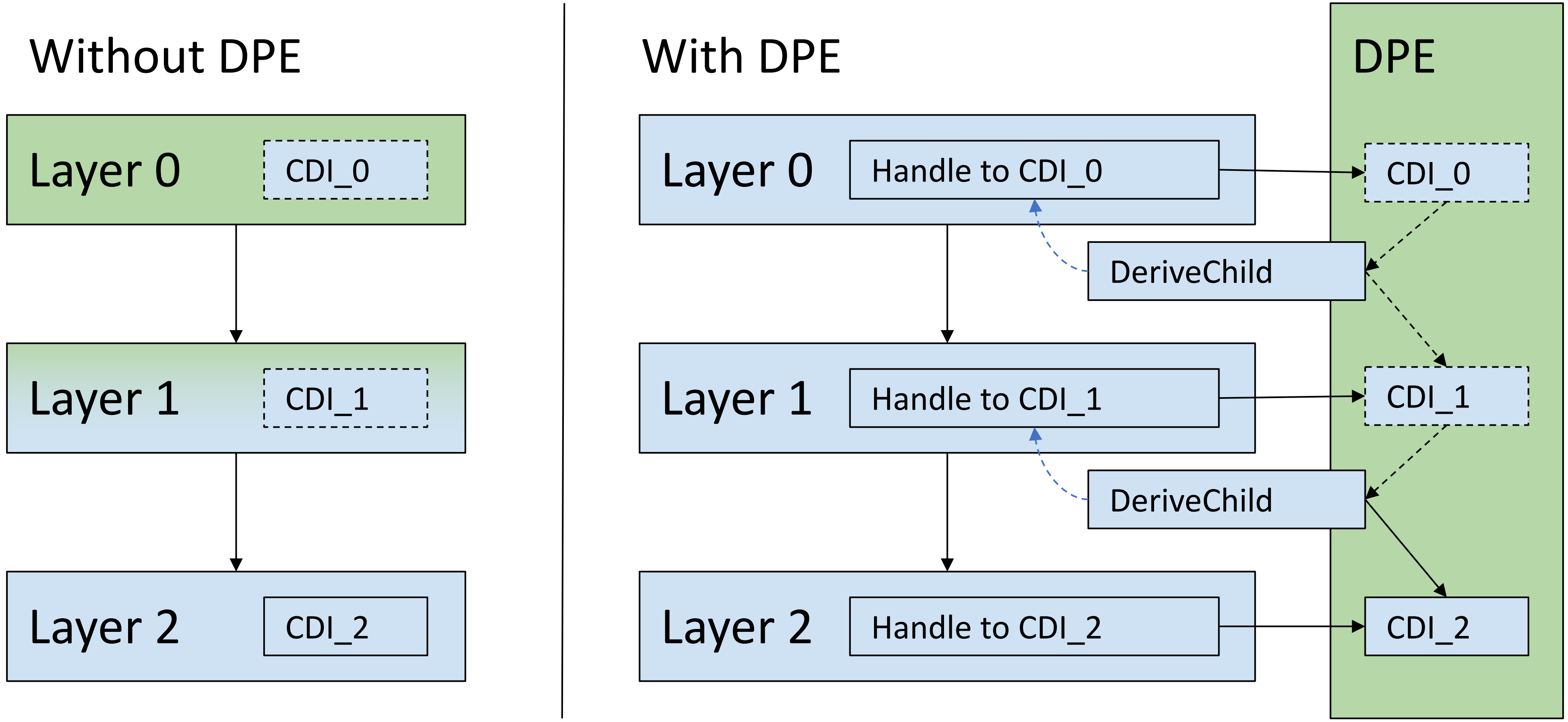
Layering Architecture with DPE



- DPE enhances and hardens DICE implementations
  - Secrets, keys, protected behind a DPE interface, hardened DPE implementations
  - Trusted functionality modularization
  - Simpler, less costly FIPS140-3 compliance evaluation
  - Lower cost implementations

DPE Primer: Basic Idea

DPE = DICE Protection Environment

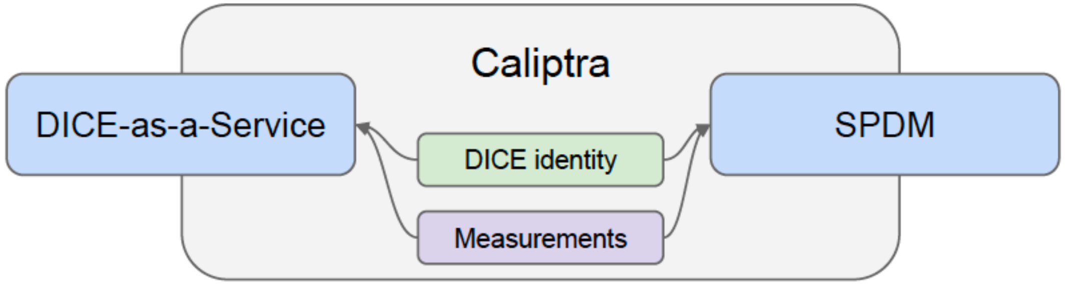


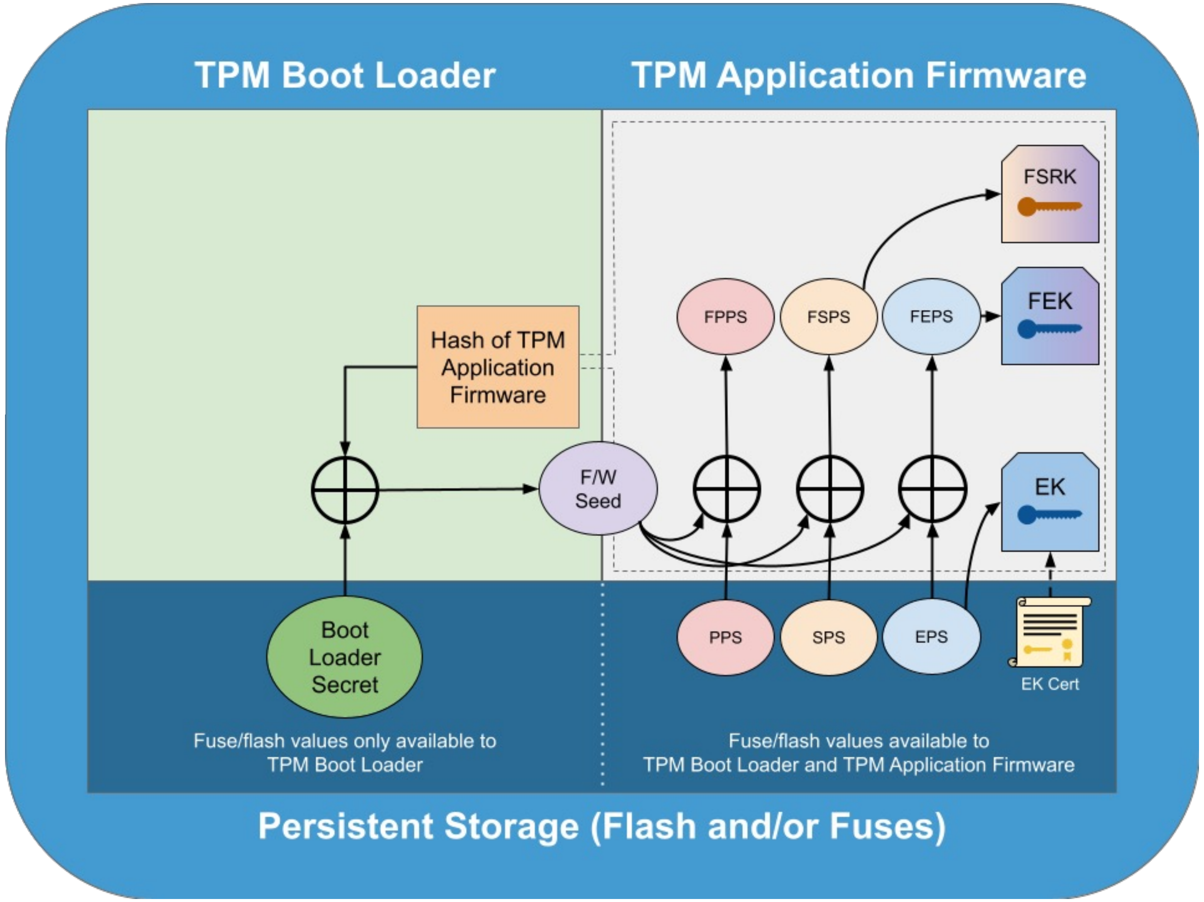
DPE Primer: Addressing Challenges

- Protects CDIs from exfiltration, leakage
- Can work as a cache, performing expensive crypto asynchronously
- Can enforce policies for sealing, simulation, etc
- Defines exactly how to do DICE, get both interoperability and flexibility
- Provides a single implementation for multiple client components / layers
- Can enforce good practices, hygiene

Other Groups Looking at DICE

- Open Compute Project
  - Cerberus,
  - Caliptra
- TCG TPM
  - Attestable TPM Firmware





Backup

DICE Challenge #1: CDI handling / protection

- CDIs are sensitive secrets
- CDIs are handled by the components themselves
- Different components may handle CDIs inconsistently
- CDIs may be exfiltrated by exploiting vulnerable components
- CDIs may leak as they are passed from one component to another
- CDIs may leak due to system memory management (e.g. swap)

DICE Challenge #2: Performance

- Working with asymmetric keys is expensive (memory, time, power)
  - PQ-safe algorithms exacerbate this
- Delays are often unacceptable in a system’s critical boot path
- Certificates are expensive (memory)
- Basic hashing may be expensive (not accelerated, contention, etc)

DICE Challenge #3: Interoperability & Consistency

- Flexibility and inclusivity => ambiguity
- There are many, many ways to implement DICE that meet the requirements of the specifications
- Makes interoperability hard across components and systems

DICE Challenge #4: Implementation Diversity

- Every component needs to implement DICE, carefully
- Quickly becomes unwieldy
- Risk of bugs, missing fixes, etc
- Challenge for both security and quality in general

DICE Nice-to-have: Sealing

- Brittle measurements (e.g. different across update) don’t work well for sealing
- Stable measurements (e.g. same across update) work for sealing but may fail to capture important system details
- DICE has no mechanism for complex policy evaluation or enforcement

DICE Nice-to-have: Simulation

- It is not possible, by definition, for a component in a DICE system to simulate what would happen if itself and/or its parent components had different measurements
- This is because with DICE, to simulate is to impersonate
- The same root issue as sealing: there is no policy enforcement