TRUSTED® COMPUTING GROUP

SPECIFICATION

DICE Layering Architecture

Version 1.0
Revision 0.19
July 23, 2020

Contact: admin@trustedcomputinggroup.org

PUBLISHED

Contents

1 SCOPE

This specification describes an architecture for device identity and attestation, which includes code integrity and other claims. This specification formalizes the process of attestation at each layer in a device's boot or startup sequence. This specification details transitions between layers and the creation of seed values that are tied to the identity of each layer. Seed values are used for key derivation operations that enable implicit attestation or other use cases involving a Device Identifier Composition Engine hardware Root of Trust [1]. This specification formalizes mechanisms for establishing trust in derived keys that can be used in a protocol for attestation. In addition to a strong device identity rooted in hardware, layered attestation is an extension to typical attestation schemes in that it also relies, implicitly, on a device's statistically unique, cryptographically strong, identity**.**

1.1 Key Words

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this document normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

1.2 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statements.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind informative comment ...

This is the second paragraph of text of the kind informative comment ...

This is the nth paragraph of text of the kind informative comment ...

To understand the TCG specification the user must read the specification. (This use of MUST requires no action).

End of informative comment

2 REFERENCES

[1] Trusted Computing Group, "Hardware Requirements for a Device Identifier Composition Engine," 2018. [Online]. Available: https://www.trustedcomputinggroup.org.

[2] Trusted Computing Group, "TCG Glossary," 2017. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-glossary/.

[3] NIST, "NIST Compouter Security Resource Center Glossary," [Online]. Available: https://csrc.nist.gov/glossary.

[4] IEEE, "802.1AR: Secure Device Identity," 2018. [Online]. Available: https://www.ieee.org/.

[5] Trusted Computing Group, "Implicit Identity Based Device Attestation," 2018. [Online]. Available: https://trustedcomputinggroup.org/.

[6] NIST, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes," 2018. [Online]. Available: https://www.nist.gov.

[7] Internet Engineering Task Force, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," 2015. [Online]. Available: https://datatracker.ietf.org/doc/rfc5280/.

[8] Internet Engineering Task Force, "CBOR Web Token (CWT)," 2019. [Online]. Available: https://datatracker.ietf.org/doc/rfc8392/.

[9] Internet Engineering Task Force, "The Kerberos Network Authentication Service (V5)," 2015. [Online]. Available: https://datatracker.ietf.org/doc/rfc4120/.

[10] Internet Engineering Task Force, "A Voucher Artifact for Bootstrapping Protocols," 2020. [Online]. Available: https://datatracker.ietf.org/doc/rfc8366/.

[11] IEEE, "802.1X-2004 - IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control," 2004. [Online]. Available: https://standards.ieee.org/standard/802_1X-2004.html.

[12] Trusted Computing Group, "Symmetric Identity Based Device Attestation," 2019. [Online]. Available: https://trustedcomputinggroup.org.

3 TERMS AND DEFINITIONS

For the purposes of this document, the following terms and definitions apply.

3.1 Trusted Computing Terms

This section contains terminology commonly understood by security, cryptology, and trusted computing practitioners. The reader may be interested in the following terminology references:

- Trusted Computing Group Glossary [2],
- NIST Computer Security Resource Center Glossary [3].

3.2 Glossary

TERM	DEFINITION
Digest	The result of a cryptographic hash operation.
Device	A highly integrated platform containing a programmable component with other optional programmable components and peripherals.
DevID, IDevID, LDevID	These terms are defined by the IEEE 802.1AR [4] standard as information that an entity (a person or device) possesses that allow it to make a verifiable claim of identity, i.e., to be authenticated.
Measurement	A digest of code and/or configuration data. It is implementation-specific, and out of scope for this document, whether a measurement is over a region of memory, a firmware or software image, or some combination thereof.

3.3 DICE Architecture Terminology Conventions

This section describes conventions for use of the DICE acronym in connection with various concepts found in the architecture where the literal expansion of the DICE acronym may result in confusing or awkward syntax.

TERM	DEFINITION
Compound Device Identifier	The Compound Device Identifier (CDI) is a secret value resulting from the application of a cryptographic one-way function to a combination of a DICE Layer's secret value and the measurement of the subsequent DICE Layer.
DICE	Device Identifier Composition Engine, a hardware Root of Trust (RoT)
DICE Architecture	This usage refers to the set of concepts that make up the trusted computing architecture with a Device Identity Composition Engine as its central feature.
DICE Engine	See DICE. The redundancy in terms is noted and it confers no further meaning.
DICE Layer	This is a shorthand term used to describe an element of a DICE Architecture. See Section 5.3.
DeviceID	An asymmetric key that authenticates a combination of device and firmware. This term refers specifically to the key derived from the Compound Device Identifier value that is produced by the DICE.
Layered Identity, DICE Layered Identity	An identity that cannot exist without a precise chain of TCB components because it is derived from a Compound Device Identifier.

3.4 Abbreviations

For the purposes of this specification, the following abbreviations apply.

ABBREVIATION	DESCRIPTION
CDI	Compound Device Identifier
DICE	Device Identifier Composition Engine
ECA	Embedded Certificate Authority
FMC	First Mutable Code
FSD	Firmware Security Descriptor
HRoT	Hardware Root-of-Trust
KDF	Key Derivation Function
MFG	Manufacturer
OID	Object Identifier
OWF	One Way Function
PCR	Platform Configuration Register
PRF	Pseudo Random Function
PSK	Pre-Shared Key
RoT	Root-of-Trust
RTM	Root-of-Trust-for-Measurement
SVN	Security Version Number
TCB	Trusted Computing Base
TCI	TCB Component Identifier
TEE	Trusted Execution Environment
TLS	Transport Layer Security
UDS	Unique Device Secret

4 INTRODUCTION

This document assumes the reader is familiar with the TCG specification Hardware Requirements for a Device Identifier Composition Engine [1], i.e., DICE. The DICE specification describes the composition of an identifier that represents the combination of hardware and software that begins a device boot sequence. This document is a normative specification describing how this concept is extended beyond this first layer to any number of layered components. This is valuable because transitions between device components across layers are often security relevant and result in increasing code size and functional complexity.

The DICE specification describes the construction of the Compound Device Identifier (CDI). In this specification that construction also applies to transitions between components of a device's Trusted Computing Base (TCB). A device's TCB consists of all security relevant components that have been loaded at a given point in the boot sequence. A TCB component is comprised of hardware, firmware, software and/or configuration. A measurement of a TCB component is a TCB Component Identifier (TCI). An example of a TCI value is a digest of component firmware. In some cases where a component does not consist of measurable firmware or software, a hardware product identifier (or equivalent) may be used.

This architecture takes a layering approach to model multi-component systems. DICE hardware is used as the hardware Root of Trust (RoT) that anchors every layered component. The DICE hardware specification [4] describes the hardware layer combining the DICE hardware secret (called a UDS) with a measurement of the next firmware/software component, and providing a CDI secret to that next layer. Each layered TCB component performs an analogous process, combining the current TCB component's CDI secret with a measurement (TCI) of the next TCB component, and providing the next TCB component with its own CDI.

TCB components also use their DICE Compound Device Identifier (CDI) as the input to a key generation function. For example, this may be an asymmetric key generation function producing a key pair that may be enrolled as an 802.11AR device identity credential, known as IDevID or LDevID [4]. Keys derived from CDI values may be enrolled with an application specific certificate authority and used to perform attestation, authentication, and certification. This specification doesn't specify key hardening or protection requirements; it assumes the layer that generates the private key is sufficiently protected.

This specification requires hardware that complies with the DICE hardware specification [1]. How the Unique Device Secret (UDS) is provisioned within a device is not in scope; only that it has been provisioned. This architecture presents a solution that does not require manufacturers or vendors to maintain databases of UDS values.

While DICE was originally designed for resource constrained devices, this does not imply any reduced benefit to implementing DICE in complex systems. DICE provides building blocks that systems can use to address multi-tenancy, scalability, resiliency, and recoverability goals.

5 DICE CORE CONCEPTS

This section contains the core concepts of the DICE Root of Trust and layering. Additional details on the DICE Root of Trust can be found in [1]. The Device Identifier Composition Engine (DICE) is a hardware and firmware capability that establishes a verifiable cryptographic identity that, assuming the Compound Device Identifier is protected as indicated in the DICE specification, implicitly attests that an expected TCB is instantiated during the boot process.

DICE performs measurements of code (and, optionally, configuration data) and generates a cryptographically unique value, called the Compound Device Identifier (CDI). The CDI entropy derives from a Unique Device Secret (UDS) and is cryptographically bound to an execution environment by measuring the execution environment's runtime code. This code is sometimes referred to as First Mutable Code (FMC); this specification uses the term layer 0. The layer 0 measurement is combined with the UDS to produce the CDI.

5.1 Purpose

The DICE layering architecture provides a way to know which components are operational and which mutable code is currently loaded in these components in a specific device. A cryptographically strong device identity is essential for linking a device's DICE hardware Root of Trust to the DICE layer performing attestation. A DICE layering architecture provides the context in which secrets and keys are protected and provides the foundation for proving attestation assertions.

5.2 Unique Device Secret (UDS)

The Unique Device Secret is a statistically unique, device-specific, secret value. The UDS may be generated externally and installed during manufacture and/or generated internally during device provisioning. The UDS must be stored in non-volatile memory on the device, e.g., eFuses, or any other suitably protected area to which the DICE can restrict access. See the DICE hardware specification [1] for details.

5.3 Components and Layers

The device is presumed to be composed of multiple components that are layered or organized in a tree structure. A layer is comprised of one or more components. Layers are numbered relative to the point of interaction with a DICE implementation. By convention, the layer that receives the CDI value from a DICE implementation is layer 0.

5.4 TCB Components

A device's Trusted Computing Base (TCB) consists of multiple components. All security relevant components are part of the device's TCB. This specification refers to these as TCB components.

5.5 TCB Component Identifier (TCI)

Components are individually identifiable. The measurement of a TCB component is used to construct a component identifier. For example, a layer component identifier may consist of a digest of one or more of the following: firmware, configuration, vendor name, product information, version, Security Version Number (SVN), and/or an instance identifier. Multiple component instances may have the same TCI. If a TCB component contains measurable firmware or software, then the TCI for that component includes a measurement of that firmware or software.

5.6 Compound Device Identifier (CDI)

The CDI is the result of a one-way function that accepts as input (1) the TCI of the next component and (2) the CDI value received from the previous component.

The initial CDI in a DICE system is computed by a DICE HRoT that measures its layer 0 environment (firmware, configuration data, identifying attributes) plus the Unique Device Secret (UDS).

5.7 Hardware Roots of Trust

Hardware Roots of Trust are presumed to be immutable, though in practice many manufacturers implement mechanisms for correcting vulnerabilities. While the DICE hardware requirements [1] permit this, the DICE architecture philosophy regarding hardware Roots of Trust includes the possibility that implementations of protected capabilities and shielded location technologies have provable correctness, so it becomes feasible to avoid needing post-production update of a HRoT.

5.8 Summary

The DICE engine, and a meaningful composition of layered components, make up a device. An identity for the device is derived from this composition. Any device that implements the Device Identifier Composition Engine can support device attestation [5]. Therefore, a platform comprised of multiple devices may contain multiple attestable DICE identities (See Section 7.2).

6 DICE LAYERING ARCHITECTURE

DICE defines a hardware Root of Trust (RoT). This architecture uses DICE as the basis of a multi-layered Trusted Computing Base (TCB). The RoT must be trusted because its misbehavior cannot be detected. A layered TCB architecture allows a lower TCB layer to supply trusted functionality to a higher TCB layer. Successive layers may supply increasingly sophisticated trusted functionality, but a minimal set of functions is needed to support TCB layering.

Start of informative comment

The phrase "lower TCB layer" refers to a TCB component that is below (i.e., executes before) the current TCB layer. Conversely, the phrase "higher TCB layer" refers to a TCB component that is above (i.e., executes after) the current TCB layer. Lower TCB layers are typically less complex, of smaller code size, and implement more security-sensitive functionality.

End of informative comment

6.1 System Layering

The layering architecture based on DICE considers the execution states that are entered progressing from a base hardware layer (i.e., the DICE hardware Root of Trust (HRoT)). The base hardware layer is assumed to be in a trustworthy state before transitioning to layer 0. Layer 0 is assumed to be in a trustworthy state before transitioning to layer 1 and so on. Those familiar with the concept of a Root of Trust for Measurement (RTM) [2] will recognize the base hardware layer as an RTM. Transitioning between layers of a DICE layering architecture involves creating a CDI value and securely passing it to the next layer. DICE layering is similar to a trusted boot process as defined by the Trusted Computing Group [2] with the exception that integrity measurements of a layer *n* are not necessarily extended into a PCR as the CDI contains measured values that are securely passed to the next layer.

Figure 1 highlights several examples of layered systems to show a relationship to DICE layering. It is not a goal of this specification to prescribe specific sequencing of layered objects. Instead, DICE layering architecture design goals focus on the construction of attestable capabilities at each layer as well as resilient attack and/or corruption detection and recovery mechanisms. Recovery from attack depends on each layer being able to recreate a layer specific key as part of the transition to the next layer. Since each layer is measured prior to becoming active, if a layer is compromised then all subsequent layers likewise are unable to make similar trustworthiness assertions because all their CDI dependent keys are changed.

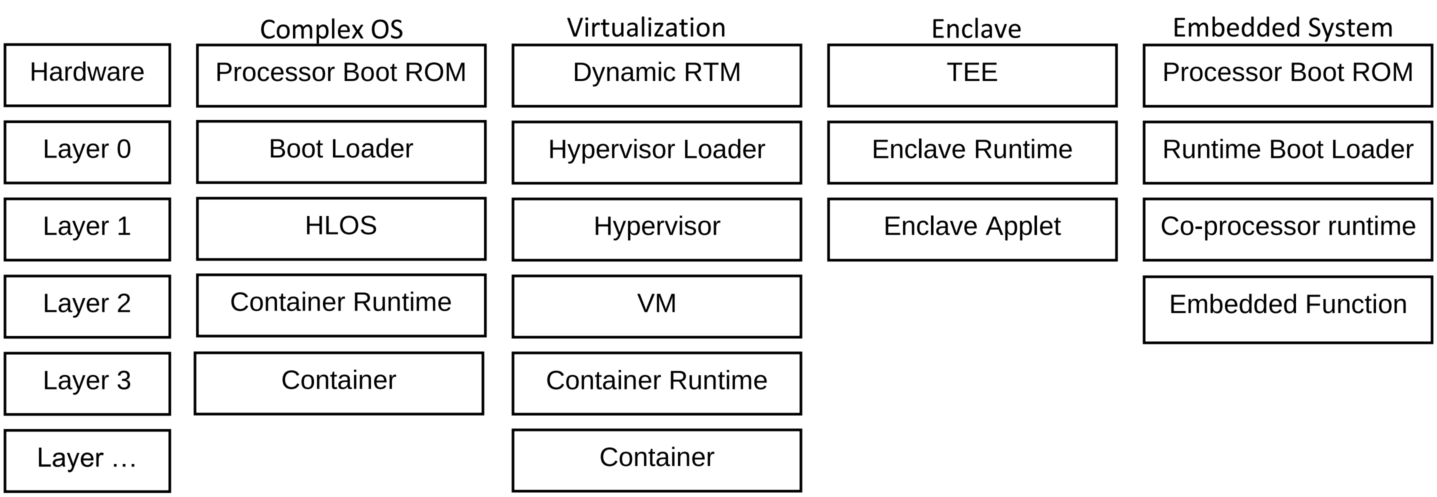

Figure 1: Examples of system layering

6.2 TCB Layering

A layered TCB architecture uses a constrained set of TCB capabilities to construct the next TCB layer. TCB capabilities are assumed to be protected within a hardened execution environment. The transition from one TCB layer to the next is assumed to be protected using interaction capabilities trusted by both TCB layers.

6.2.1 TCB Capabilities

Every TCB layer (see Figure 2) MUST have trusted access to TCB capabilities that can produce the following:

1. **TCB Component Identifier** (TCI) – See Section 5.5. The TCI is a component-specific identifier and describes a TCB component. Examples of TCI values include (i) a hash computed over runtime code that executes in shielded locations, (ii) a code measurement combined with either a product identifier (e.g., vendor-model-version or a vendor-model-SVN[1]), or (iii) a hash of an FPGA bitstream that can be loaded into programmable hardware. The TCI value for any TCB component that includes firmware or software MUST include measurement of said firmware or software. Any change to a TCB component MUST result in a different TCI value for that TCB component. A given layer *n* MUST use a trustworthy mechanism for computing the TCI value corresponding to layer *n+1*.
2. **Compound Device Identifier** (CDI) – See Section 5.6. The CDI value received by a layer *n* MUST be based on a minimum of two input values: (i) the previous CDI value (CDI_{n-1}) and (ii) the TCI of the target TCB component (TCI_n).[2] The input values are combined using a one-way function (OWF). Additional values may be included in a CDI computation for a given layer. The Unique Device Secret (UDS) supplies a statistically unique value to the DICE HRoT layer since no previous context exists. At layers above the DICE HRoT layer, the CDI value received from the previous TCB component supplies a statistically unique value to the current TCB component. A component MUST use a trustworthy mechanism for producing the CDI value of a subsequent component and a component MUST use a trustworthy mechanism for providing that CDI value to a subsequent component.
3. **One-way Function** (OWF) – A cryptographic pseudo-random function (PRF) that complies with NIST SP800-56c recommendations [6]. The PRF accepts seed (s) and data (x) values. The seed and data values for subsequent components MUST be the CDI value received from the previous component and the TCB Component Identity (TCI) of the next component, respectively.

Start of informative comment

Note that Figure 2 is not meant to imply that the measurement of a given layer includes the secret CDI value passed to that layer nor the TCI value that layer computes for the subsequent layer. This would not be possible in practice, despite the diagram's implication otherwise. For example, the measurement of layer 1 is taken by layer 0 before layer 1 is ever in possession of CDI_{L1}, since CDI_{L1} is dependent on this measurement. Similarly, layer 1 would not be in possession of TCI_{L2} until it is computed by layer 1, which can only occur after the transition from layer 0.

End of informative comment

---

[1] The SVN is a monotonic value that the vendor uses to notify security relevant changes to a TCB component. An SVN differs from a traditional version number in that non-security relevant changes to the component may not increment the SVN.

[2] There are scenarios in which a given layer may consist of multiple components each with an individual identity. For example, consider trusted applications that have equivalent security properties, but are isolated from one another within a trusted execution environment (TEE), each having unique CDI values.

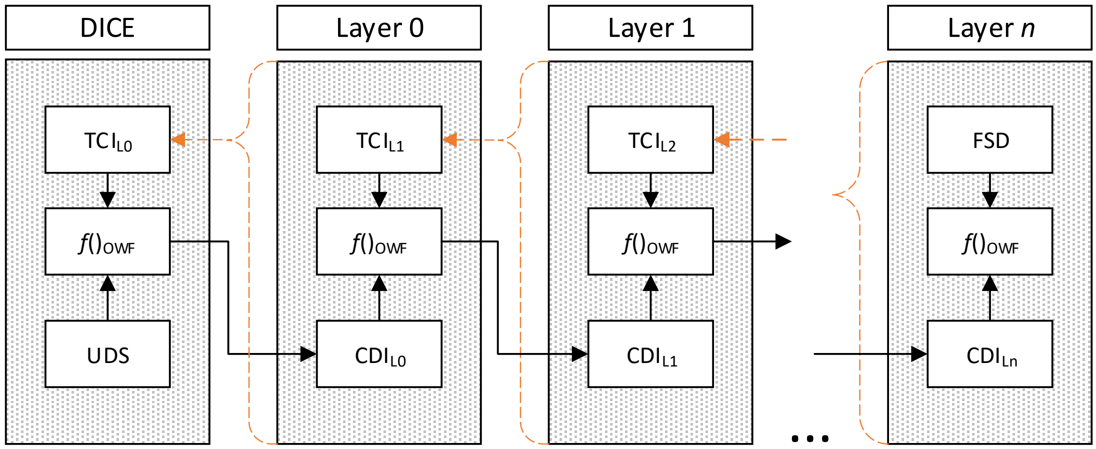

Figure 2: TCB layering architecture

6.2.2 Other TCB Protections

Start of informative comment

TCB capabilities exist either in hardened execution environments or during early boot stages in which they are in complete control of system execution. Definition of the environment and hardening criteria is out of scope for this specification. However, a manufacturer can describe TCB protection properties using manufacturer issued attestation certificates and manifests. Inter-TCB communication may require implementation specific hardening to avoid possible man-in-the-middle attacks or to detect compromise in a subsequent layer prior to a layer transition.

End of informative comment

6.2.3 DICE Layered Identity

A DICE Layered Identity cannot exist without a precise chain of TCB components because it is derived from a CDI value. This is in contrast to randomly generated identities that may also be held by a device or component. The CDI value is an accumulation of the measurements of all preceding TCB components. A Layered Identity, therefore, describes not only the identity of device TCB components but also their order.

6.2.4 Layered Trust Considerations

6.2.4.1 DICE HRoT Trust Considerations

This section describes trust requirements pertaining to the base hardware layer.

A DICE HRoT has the following requirements (see also [1]):

1) The UDS MUST be of sufficient security strength for its usages.
2) The UDS and measurement of layer 0 MUST contribute to the layer 0 CDI computation.
3) The DICE SHOULD NOT access other layer secrets besides the CDI value.

Start of informative comment

Since the protection of layer secrets is critical for maintaining the secrecy of private keys, it is important for the DICE to access and/or use layer secrets in a way that avoids or prevents other entities accessing or intercepting these values. Similarly, interference with the transition of execution from DICE to layer 0 should be infeasible.

End of informative comment

4) The DICE manufacturer or vendor SHOULD assert the DICE HRoT's trustworthiness properties (e.g., via the DeviceID certificate, see Section 9.2).
5) Secret values, if present, MUST be provisioned securely within the DICE and not externally visible.

6) If the DICE generates keys, then a secure entropy source MUST be part of the DICE HRoT.

6.2.4.2 Layer 0 Trust Considerations

This section describes trust requirements pertaining to layer 0.

1) All common trust considerations (see Section 6.2.4.3) apply to layer 0.
2) The DeviceID key MUST be created by a manufacturer-controlled process.
3) The DeviceID key derivation process MUST depend on the CDI value describing Layer 0
4) The manufacturer SHOULD issue a DeviceID certificate (e.g., IEEE IDevID).
5) If layer 0 is not modifiable outside of a manufacturer-controlled process, then both the DeviceID key and certificate MUST NOT be modifiable outside of said process.

Start of informative comment

If the DeviceID key certificate is expected to be irrevocable and not expire, then is it expected that layer 0 is not modifiable outside a manufacturer-controlled process.

End of Informative Comment

6) Any change to the layer 0 TCB MUST result in a different DeviceID key.

6.2.4.3 Common Layer Requirements

This section describes layer trust requirements pertaining to layers 0 thru n.

Trust requirements common to all layers are as follows:

1) A DICE layer MUST be constructed from shielded locations and protected capabilities [2].
2) A DICE layer MAY be constructed using shielded locations and protected capabilities used by previous DICE layers. This consideration SHALL NOT apply to layer 0.

Start of informative comment

The definition of a shielded location is a place (memory, register, etc.) where it is safe to operate on sensitive data. Since a DICE layer operates on sensitive data (e.g., its CDI value) to which it must have exclusive access, the location of this data constitutes a shielded location as long as said DICE layer has exclusive access.

The set of commands with exclusive permission to access shielded locations are considered protected capabilities. This definition includes code that constitutes a given DICE layer since it has exclusive permission to access shielded locations.

Considerations 1 and 2 are intended to protect layer secrets while they are in use.

End of informative comment

3) A DICE layer MUST protect its secret CDI value and private keys derived therefrom.
4) A DICE layer MUST NOT implicitly trust a layer that executes after it.

Start of informative comment

A DICE layer assumes that all layers below are trustworthy, including the DICE HRoT layer. Trusting a layer implies one also trusts previous layers and their hardware roots.

End of informative comment

5) A DICE layer MUST create its own secrets, private keys, and trust anchors, or obtain them from a previous layer.
6) DICE layer n keys MUST be generated by layer n or by a previous layer.
7) The CDI value provided to layer n MUST contribute to the layer n+1 CDI computation.
8) If a DICE layer implements a CA, the CA MUST be an Embedded Certificate Authority (ECA) that certifies layer specific keys following ECA defined procedures (see Section 9.2).

9) If a device provides attestation of layer n, that attestation MUST be provided by layer n or by layers below.

7 CERTIFICATION, ATTESTATION, AND AUTHENTICATION

This section describes important TCB capabilities that build on a DICE layering architecture.

7.1 Certification and Token Issuance

Certification of a DICE layered component refers to the issuance of a certificate or token that links the next layer TCB to the current layer TCB. Public keys for a next layer can be generated and certified in the current layer before passing control to the next layer. Certification related capability may exist in a given layer TCB when it is appropriate for the layer to certify keys.

Certification may be applied to both asymmetric and symmetric key types. Asymmetric keys are certified using a certificate (e.g. X.509v3 [7]). Symmetric keys are issued in tokens (See RFC8392 (CWT) [8] and RFC4120 (Kerberos) [9]).

Certification using certificates has two forms (i) the TCB layer generates the key pair and signs the certificate for the next layer TCB (See Section 9.2.2.1) or (ii) the next layer TCB generates the key pair and obtains a certificate from the previous layer TCB (See Section 9.2.2.2).

Certification using tokens involves the certifying TCB constructing a token containing the TCI of the next layer TCB, the next layer CDI, and a symmetric key derived from the CDI.

The method by which the symmetric key is shared or exchanged with a verifier is out of scope for this document.

7.2 Attestation

Attestation of a DICE layered component refers to the use of a symmetric or asymmetric key that is approved by the layer Embedded Certificate Authority for attestation (See Section 9.2). Attestation asserts trustworthiness properties about a TCB layer or component. Trustworthiness properties may be explicit – meaning properties are enumerated and an encoding of them is available for inspection by an attestation verifier. Trustworthiness properties may be implicit – meaning properties are inferred by a verifier based on a condition or state that wouldn't otherwise be possible.

A DICE TCB layer that supports attestation inspects a subsequent layer TCB or component to collect its trustworthiness properties. A TCI computation collects trustworthiness properties inherent in code and settings used to execute a subsequent layer TCB. The inspecting TCB layer also creates attestation evidence about a subsequent layer that is used by an attestation verifier. Evidence may be certified by the inspecting TCB layer's ECA or signed by an attestation key (See Section 8.1) such that a verifier can associate the attestation key with the trustworthiness properties of the TCB that controls the key.

7.3 Authentication

Authentication of a DICE layered component refers to the use of a symmetric or asymmetric key that is authorized to authenticate the device. A DICE layer that participates in device authentication protocols (e.g., RFC 8366 [10], IEEE 802.1X [11]) includes this capability in its TCB.

7.4 Summary

A DICE layer TCB can be configured to include additional capabilities beyond those required for secure transition to a next layer. This specification identifies additional certification, attestation and authentication capabilities for DICE layer TCBs. Other additional capabilities may be appropriate. Designers SHOULD determine whether a TCB capability is essential. Limiting TCB complexity helps manage the TCB attack surface. Therefore, careful consideration should be made before including additional capabilities.

This DICE layering architecture anticipates increased TCB complexity in later layers. Consequently, later layers may be more vulnerable to compromise. A possible response to compromise may involve returning control to lower DICE layers that may be less vulnerable to compromise and capable of restoring proper operation.

8 KEYS AND CREDENTIALS

The TCB in each layer may manage several types of keys. This section defines key types, expected use, generation and derivation requirements, and protection assumptions.

8.1 Key Types

The following sections are descriptions of the types of keys (both symmetric and asymmetric) and credentials that may be used in a DICE layering architecture.

8.1.1 Asymmetric Keys

8.1.1.1 Embedded Certificate Authority (ECA) Keys

An ECA Key is an asymmetric key that a TCB component uses to issue (sign) certificates for keys generated for the current or subsequent TCB layer. For example, keys generated at a higher TCB layer could obtain a certificate issued by a lower TCB layer. When the derivation of an ECA Key is deterministic and dependent on the combination of a CDI value and the TCB component identifier (TCI), then use of an ECA Key (for signing ECA certificates) is an implicit statement of Layered Identity. ECA Keys MUST only sign data that is known to the TCB layer. For example, ECA Keys are not used to sign opaque structures from outside the TCB layer.

8.1.1.2 Attestation Keys

An Attestation Key is any asymmetric key used to sign attestation evidence. When the derivation of an Attestation Key is deterministic and dependent on a CDI value then use of an Attestation Key to sign attestation evidence is an implicit statement of Layered Identity. Like ECA Keys, Attestation Keys MUST only sign data structures that are known to the TCB. For example, Attestation Keys are not used to sign opaque structures from outside the TCB layer. However, including data from outside the TCB layer, like a random nonce, is expected as part of a signed response to an attestation request.

8.1.1.3 Identity Keys

An Identity Key is an asymmetric key used for signing authentication challenges (i.e., statements of Layered Identity, whether implicit or explicit) is an identity key. For example, the leaf in an ECA x509 certificate chain used for TLS client authentication uses an Identity Key.

8.1.1.4 Summary

Table 1 provides a summary of asymmetric key types and their respective functions. Asymmetric keys can have multiple attributes.

	CERTIFICATION	ATTESTATION	IDENTITY	NOTES
ECA Key	✓	✓	✓	Only sign known data
Attestation Key		✓	✓	Only sign known data
Identity Key			✓	May sign opaque challenge

Table 1: Asymmetric key types and their permitted usage types

8.1.1.5 DeviceID and Alias Keys

The DeviceID and Alias Keys are instances of DICE key types. The DeviceID key is the asymmetric key derived from the CDI value that is computed by the DICE. It is dependent on the combination of the device's UDS value and the measurement of Layer 0. The DeviceID key is a long-lived identifier for a device and is used to sign one or more certificates for keys generated on behalf of a higher TCB layer. Therefore, the DeviceID key is an ECA Key. The DeviceID key may be used to sign certificates that also contain attestation evidence, so it is also an Attestation Key. The DeviceID key is typically certified during manufacturing and therefore provides device provenance.

The Alias Key is an asymmetric key that is computed using the last CDI value in the chain of TCB components. The Alias Key and corresponding certificate are typically leaf nodes in the DICE derivation chain and contain attestation information about top-level device firmware. The Alias Key is used to sign attestation evidence; therefore, it is an attestation key.

Lastly, since the DeviceID and Alias Key are both CDI-derived keys, both are DICE Layered Identities and considered Identity Keys.

8.1.1.6 Design Considerations

While CDI values are secret, TCI values might not be, see section 6.2.1. A combination of these values forms a seed for a keypair, the usage of which constitutes an implicit statement of Layered Identity. For a keypair to act as an implicit identity key, the seed from which it is derived MUST contain the measurement (i.e., CDI) of the TCB component that it identifies.

8.1.2 Symmetric Keys

8.1.2.1 Symmetric Alias Key

While most of this specification pertains to scenarios in which asymmetric cryptography is used, there are important use cases that use only symmetric encryption. For these scenarios, a Symmetric Alias Key is used, analogous to the asymmetric Alias Key described in section 8.1.1.5. A Symmetric Alias Key is generated based on the CDI value and, optionally, a PSK ID Hint chosen by the verifier. Like the asymmetric Alias Key, this pre-shared key is used for Symmetric Key Attestation and Layered Identity. See the Symmetric Identity Based Device Attestation specification [12] for more information.

8.1.2.2 Wrapping Keys

For scenarios in which it is undesirable to regenerate asymmetric key pairs on each boot, a symmetric wrapping key may be used to persist previously derived asymmetric keys. For example, instead of layer 0 recreating the DeviceID key on each boot, layer 0 may use a value derived from the CDI as a seed for a symmetric key that is used to encrypt the DeviceID before persisting it to storage. Since the wrapping key is based on the CDI value, layer 0 is unable to reconstitute the correct DeviceID key pair without also possessing the correct CDI value.

8.1.3 Credential Types

IEEE802.1AR compliant credentials are referred to as Initial Device Identifiers (IDevID) and Local Device Identifiers (LDevID) [4]. They relate to DICE layer identities as follows:

- **IDevID** – A credential containing a DeviceID (key) that is issued by a device manufacturing process. The credential describes device manufacturer provenance. The DICE TCB layer that generates an IDevID (typically layer 0) SHOULD be either immutable or mutable under the control of the device manufacturer. A DICE DeviceID becomes an IDevID when a certificate is issued that conforms to IEEE802.1AR requirements. For a DICE DeviceID to be considered an IDevID, the identity MUST be an instance identifier (see Section 6.2.1). The IDevID issuer MUST verify that the component containing, i.e., protecting, the private key and identifier is bound to the device. DeviceID keys may be enrolled with a manufacturer's certificate authority (CA) and may comply with IEEE802.1AR requirements.
- **LDevID** – A credential containing a DeviceID-derived (key) that is issued by the device owner process, typically created on deployment in the owner's network. The DICE TCB layer that generates an LDevID (typically layer *n*, where $n > 0$) SHOULD allow creation of an owner specific LDevID. If the TCB layer is mutable, the owner SHOULD control mutability.

Key type specific seed derivations SHOULD include data input to the one-way function that disambiguates the key type. For example, an LDevID key could be seeded where the seed = PRF (CDI, "LDevID").

8.2 Key Creation

There are multiple acceptable ways to derive symmetric and asymmetric keys from CDI values. This specification does not provide an exhaustive list of algorithms that may be used for this purpose. However, this section does provide examples of commonly used algorithms.

8.2.1 Asymmetric Key Generation

Asymmetric keys can be used to attest trustworthiness properties of a TCB layer. If implicit attestation is used, the key generation function MUST be deterministic based on the CDI.

Since asymmetric key generation is seeded using the CDI, layering semantics are implicitly represented in the resulting asymmetric key pair. In Figure 3 for example, layer n uses the CDI value it received from layer n-1 to generate a layer n key.

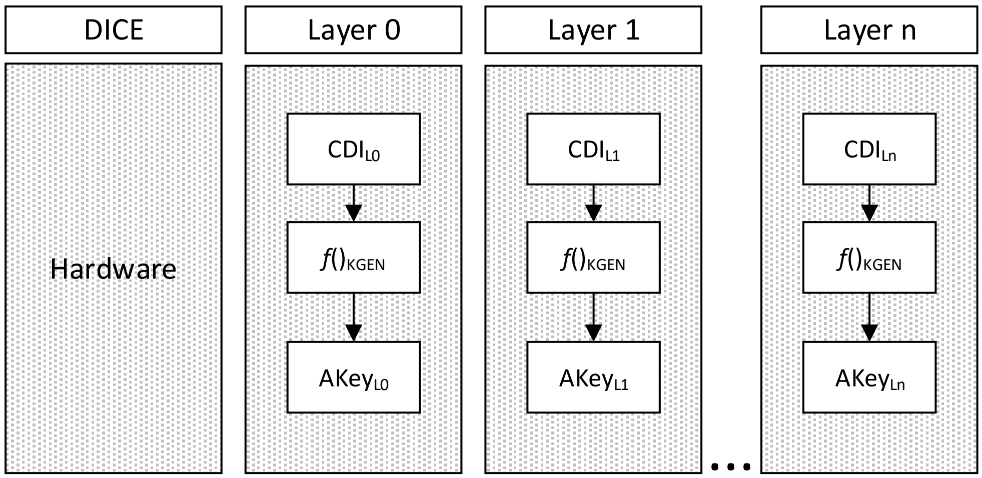

Figure 3: Asymmetric key generation example

8.2.1.1 ECDSA

The ECDSA key generation function can be made deterministic by selecting a repeatable seed value, see RFC6979. The seed SHOULD be based on the TCB context by using a value derived from the current layer CDI value. The resulting random number MUST be used as the random number input to the ECDSA key generation function.

8.2.1.2 RSA

The RSA key generation function can be based on the TCB context by using a value derived from the current layer CDI value to seed the random number generator that produces *p* and *q* primes.

Start of informative comment

ECDSA is usually favored over RSA for constrained environments due to smaller key size and improved sign operation performance.

End of informative comment

8.2.2 Symmetric Key Derivation

Symmetric key creation uses a Key Derivation Function (KDF) that complies with NIST SP800-56C recommendations. The UDS, or a CDI value derived using the UDS, MUST be used to seed the KDF. The length of CDI MUST be sufficient to ensure derived symmetric keys do not cryptographically overlap the CDI seed value, or the CDI value MUST be augmented with additional information that ensures cryptographic overlap is avoided. Figure 4 illustrates an example of this.

8.2.2.1 Device Identity and Attestation Using Symmetric Keys

See the Symmetric Identity Based Device Attestation specification [12] or more information on device identity and attestation using symmetric keys.

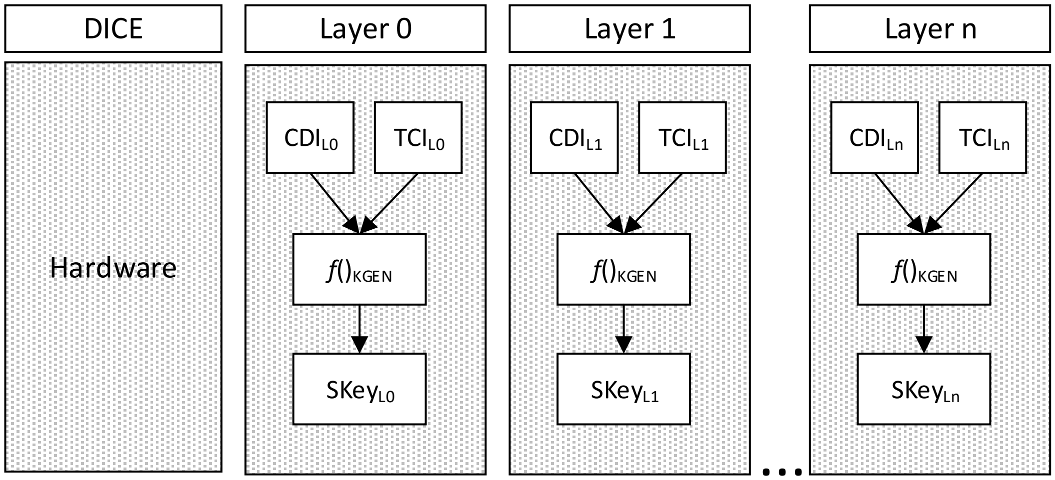

Figure 4: Symmetric key derivation example

8.3 Security Considerations

The following sections outline security considerations for key derivation, protection, and management in a layered architecture.

8.3.1 Key Protection

Knowledge of layer specific TCB context values (TCI, CDI, UDS, etc.) would allow an attacker to derive or generate the keys for a given layer. The attacker could then impersonate the layer. The TCB layer MUST protect UDS, CDI, private keys, and symmetric keys to a level that is sufficient for the expected use.

The private portion of a key MUST NOT be exposed above the layer that is trusted to protect it. Furthermore, before key protection responsibilities are passed to the next DICE layer, the current DICE layer may need to erase private key material or inputs used to generate keys in shielded locations to avoid inappropriate duplication or unauthorized use.

8.3.2 Key Persistence

In a DICE architecture it is expected that CDI values will change when there is a code change to a layer TCB. If private keys are made persistent with no reliance on a CDI value then a change to the layer TCB code would be ignored by the persisted key, resulting in a key that would implicitly attest to a configuration that is not in use. This misrepresents the actual identity, behavior, and trustworthiness state of the TCB layer.

DICE layer keys or secret values used to generate them MUST NOT be stored in unprotected storage that persists beyond the lifetime of the DICE layer that controls it. The keys and secret values provided to a device by a manufacturer MUST NOT be allowed to persist outside of the device's shielded locations.

Nevertheless, it is not a requirement that layer specific key pairs be re-generated on every reset cycle. Some devices may not have sufficient processing power to enable acceptable boot times. Several acceleration strategies exist, one example is key wrapping, see section 8.1.2.2. Using the CDI value received from the previous layer or component, a symmetric wrapping key is derived. This key is then used to wrap (encrypt) and unwrap (decrypt) persisted asymmetric private keys. On first boot, the asymmetric keys are wrapped and may be persisted in untrusted storage. Upon subsequent reboots the asymmetric keys are obtained from storage and unwrapped. Note that if the layer TCB code is updated, the current set of asymmetric keys must be revoked, and new asymmetric keys generated from the new TCB layer.

9 LAYERED CERTIFICATION

Device manufacturer certificates describe the trust properties of the DICE HRoT. Manufacturers of DICE devices may implement a CA hierarchy consisting of a manufacturing root CA with one or more subordinate CAs that issue device identity and attestation certificates.

Device identity certificates (e.g., IDevID, or DeviceID certificates) embed a cryptographic identity in the device early in the manufacturing process so as to satisfy device provenance expectations – namely, that a verifier may be reasonably assured that the device interacting with the verifier is of reputable origin.

Start of informative comment

The IEEE 802.1AR use of the term "device" (e.g., in IDevID) more closely aligns with the TCG definition of the term "platform" (see Section 3). While the DICE hardware specification [1] is typically associated with embedded devices specifically, this is not a requirement. In this specification the term "device" differs from the 802.1AR definition in that a "device" is not required to have an IDevID.

End of informative comment

Device attestation certificates assert that the device's manufacturer has embedded a cryptographic key in a device. It allows a verifier to determine which manufacturer created the device. The device certificate is used to authenticate evidence (statements about its current configuration and operational state) supporting the claim that the device is not under the control of malicious actors. Traditionally, this process is referred to as attestation [2].

Device onboarding should seek to evaluate both forms of attestation as a condition of the device's acceptance into an owner's domain. Acceptance can be described as a transfer of ownership of the device from the manufacturing and supply chain 'domain' that currently has physical and virtual control over the device to another domain that physically acquires, manages, and uses the device. The new owner typically configures the device with a new device identity (e.g., LDevID) and new attestation certificate so the device can be managed and used in the new domain and so control surfaces that existed in the previous domain are closed or placed under joint control with the new owner.

DICE certificate enrollment and certification practices SHOULD establish trust following a coordinated hand-off of the device from the current owner's domain to the new owner's domain. It also should support the new owner's desired device management and trust practices. This may include periodic reattestation, reconfiguration, and reinstallation or update of device firmware including the TCB elements.

9.1 Certificate Hierarchy

A DICE architecture may rely on Public Key Infrastructure (PKI) for device provenance. The device manufacturer SHOULD implement a certificate hierarchy that issues device certificates. The device manufacturer and other supply chain entities SHOULD issue device attribute certificates or manifests containing trustworthiness assertions that pertain to DICE trustworthiness. Figure 5 shows an example certificate hierarchy consisting of a root CA that certifies

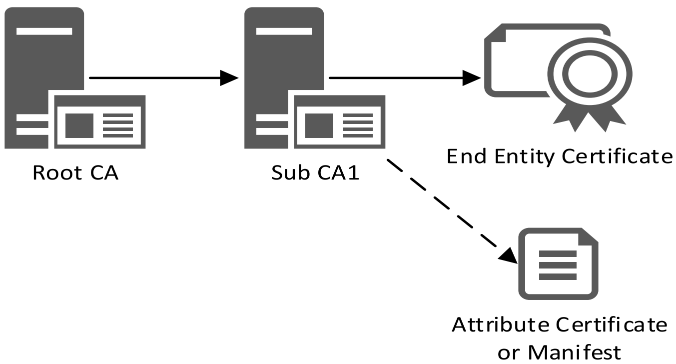

Figure 5: Certificate hierarchy with Attribute Certificate or Manifest

a subordinate CA that issues an end entity certificate. The subordinate CA may issue attribute certificates or sign manifests containing reference assertions. The end entity certificate may also contain reference assertions.

Figure 6 illustrates a sample certificate hierarchy with an Embedded CA (ECA). The device containing an ECA performs both the ECA role and the end-entity role. The ECA issues the end-entity certificate, which extends the certificate path length. Some services commonly performed by a CA may not be supported by the ECA. These may include issuing certificate revocation lists (CRL), performing online certificate status checking, and certificate storage.

Identity, attestation, and embedded certificate authority (ECA) keys can be certified using both external and embedded CAs. This results in issuance of a certificate. The decision to enroll and which CA to enroll with is use case specific.

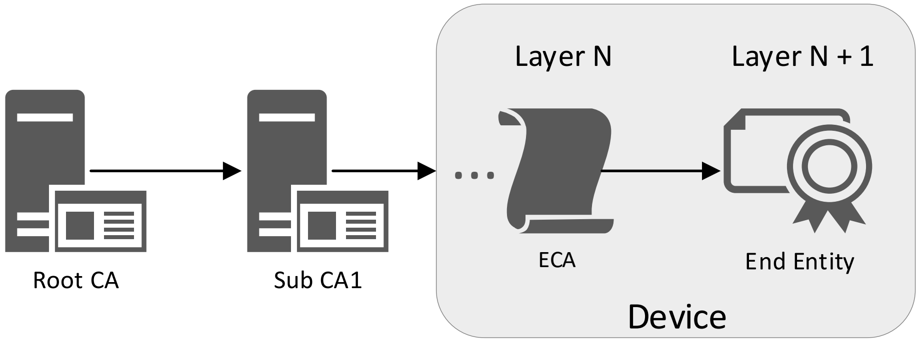

Figure 6: Certificate hierarchy with Embedded CA

A DICE TCB layering architecture anticipates the possibility for enrollment at any layer, beginning with layer 0, and with both external and embedded CAs. It also anticipates attestation as a precondition to certificate issuance. Embedded CAs may issue attribute certificates or sign manifest structures containing attestation information that pertains to a layer specific end entity certificate, see Figure 7.

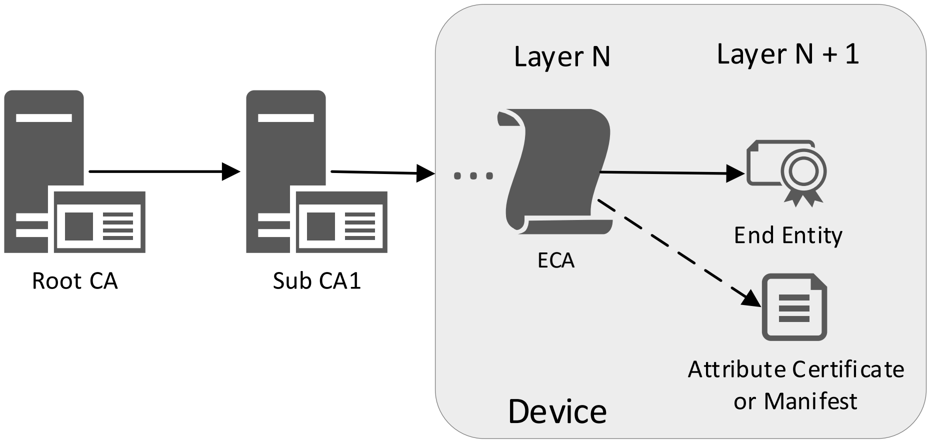

Figure 7: Certificate hierarchy with Embedded CA

Certificate formats are described in this specification according to x509v3 (see RFC5280 [7]) but are not intended to be exclusively x509.

9.2 Certification

An embedded certificate authority (ECA) is functionality contained within a DICE layer TCB that allows DICE layer keys to be certified. Certification allows higher DICE layers and external entities to verify trustworthiness at or below the DICE layer that contains the ECA. Trust in the ECA may depend on trust found in dependent layers and is rooted in the DICE HRoT. Therefore, a consumer of an ECA issued certificate may need to trace trust dependencies through

DICE layers to DICE manufacturers. Consumers of an ECA issued certificate may also expect to obtain the manufactures' root certificates to terminate path validations.

9.2.1 Layered Certification

In the layered enrollment example in Figure 8 each TCB layer has generated a layer specific identity key. Layer 0 has obtained a manufacturing certificate ($CA_{MFG}Cert_{L0}$) that can be used as initial device identity, i.e., an IDevID.

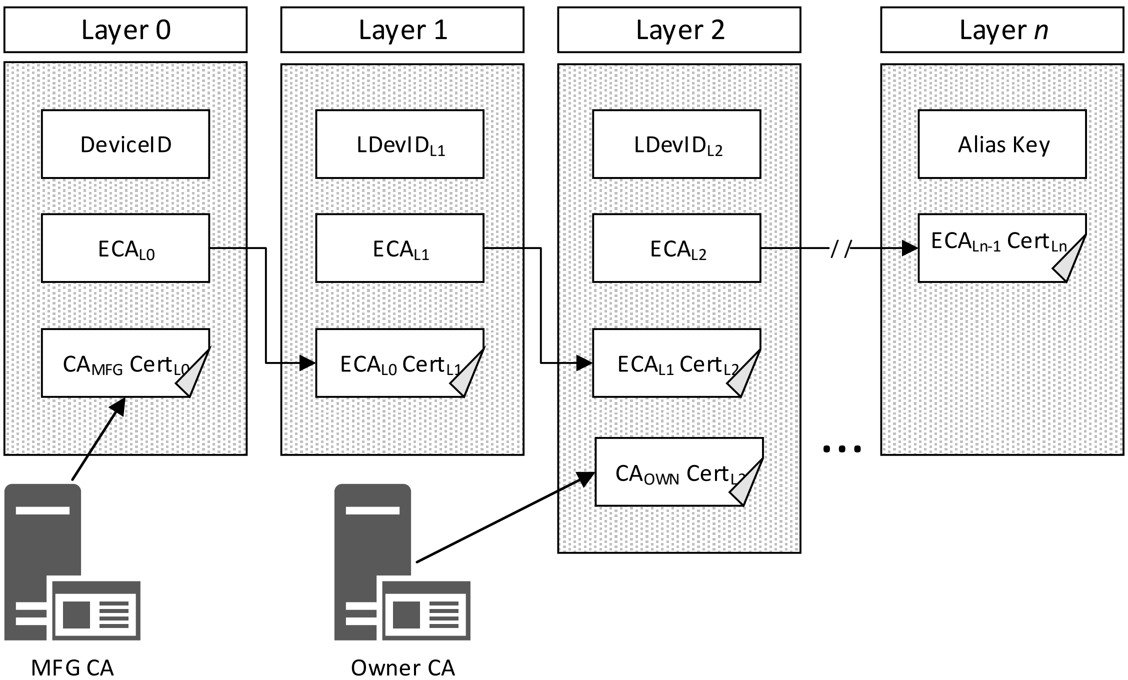

Figure 8: Layered certification example

Layer 0 implements an embedded certificate authority (ECA_{L0}) that was used to issue a certificate ($ECA_{L0}Cert_{L1}$) to the layer 1 TCB. Layer 1 also implements an ECA. It was used to issue a layer 2 certificate ($ECA_{L1}Cert_{L2}$). The device owner issued a local device identity (LDevID) certificate to the layer 2 TCB ($CA_{OWN}Cert_{L2}$). The layer 2 TCB implements an ECA that was used to issue a certificate to the next layer.

9.2.2 Certification Using Embedded CA

Certification using an Embedded Certificate Authority (ECA) allows a current layer TCB to issue a certificate that extends trust to a higher layer TCB. There are two models for intra-layer certification: (i) the ECA policy prescribes when and how to issue higher layer certificates and (ii) a higher layer creates a Certificate Signing Request (CSR).

9.2.2.1 Direct Layered Certificate by an ECA

The ECA may issue certificates according to a policy that is embedded in the ECA firmware or is securely configured. The policy describes how and when to issue layer specific certificates. The ECA generates the to-be-certified key pair and authorizes ECA certificate issuance. The ECA may securely provision the key pair to the higher layer TCB or may allow the key to be accessed by the higher layer TCB over a secure channel.

The certification steps, illustrated in Figure 9, are:

1. The layer n TCB measures layer n+1 firmware to create a TCB identifier TCI_{Ln+1}.
2. The layer n TCB uses the TCI value to compute a CDI for layer n+1.
3. The layer n TCB uses CDI_{Ln+1} to generate a key pair for layer n+1.
4. The layer n ECA issues a certificate certifying the identity of layer n+1.

5. The layer n TCB provisions the layer n+1 TCB with its CDI, private key, and certificate. Supplying the private key is optional at this step since layer n+1 could use the CDI to regenerate the private key.

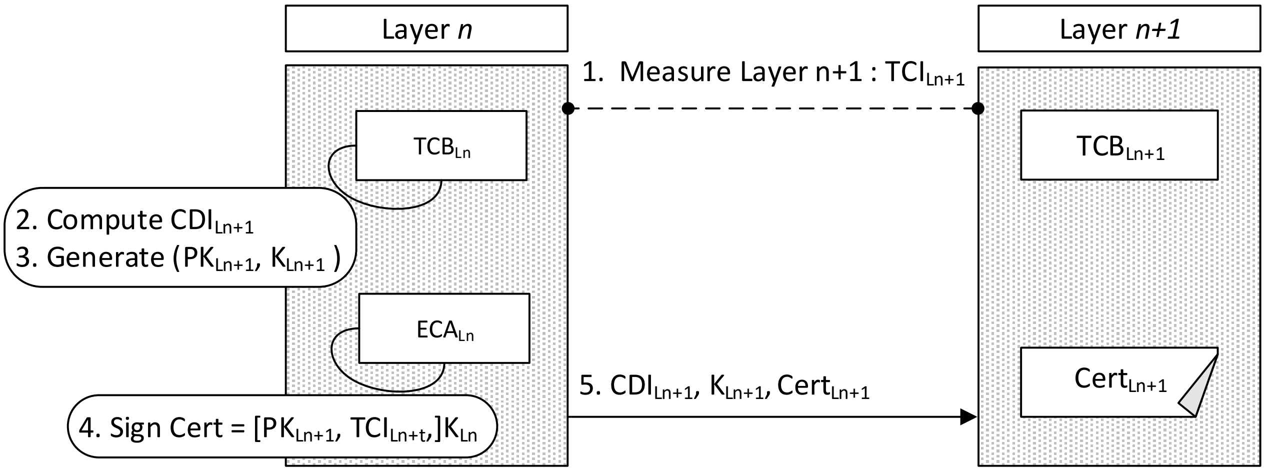

Figure 9: Direct Layered Certification by an ECA

9.2.2.2 Layered TCB Certification using CSR

An ECA may accept certificate enrollment requests from a higher layer TCB. The ECA MUST verify the CSR originates from the TCB component named in the CSR Subject. For example, a CSR may be accompanied by a layer attestation that proves layering semantics to the ECA or the CSR arrives over a trusted communication path.

The enrollment steps are, illustrated in Figure 10, are:

1. layer n TCB measures layer n+1 firmware to create a TCB identifier TCI_{Ln+1}.
2. The layer n TCB uses the TCI value to compute a CDI for layer n+1.
3. The layer n TCB securely provisions CDI_{Ln+1} to layer n+1.
4. The layer n+1 TCB uses CDI_{Ln+1} to generate a key pair for layer n+1.
5. The layer n+1 TCB constructs a CSR consisting of the public key (PK_{Ln+1}), TCB identifier (TCI_{Ln+1}) and other information that may be needed for layer n to correctly identify the layer n+1 component (such as TCI_{Ln+1} + index). Note the layer n+1 TCB may have other values it wishes to include in the CSR.
6. The layer n ECA verifies the values received from layer n+1 by verifying that the signature was created by the layer n+1 private key (K_{Ln+1}) using the CSR public key (PK_{Ln+1}).
7. The layer n ECA uses the original layer n+1 TCI and CDI values or rederives them to form CDI_{new}.

Start of informative comment

The CDI value is regenerated because layer n does not trust information provided by layer n+1.

End of informative comment

8. The layer n ECA extracts the supplied TCI value from the certificate and verifies that it matches the original value. The layer n ECA ensures the CDI_{new} is the same as CDI_{Ln+1}. Computation of the CDI value is described in section 6.2.1.
9. The layer n ECA regenerates the layer n+1 key pair (PK_{new}, K_{new}).
10. The layer n ECA verifies the layer n+1 public key is the same as the regenerated key (PK_{new}) and removes the private key (K_{new}).
11. The layer n ECA issues a certificate using the CSR template.

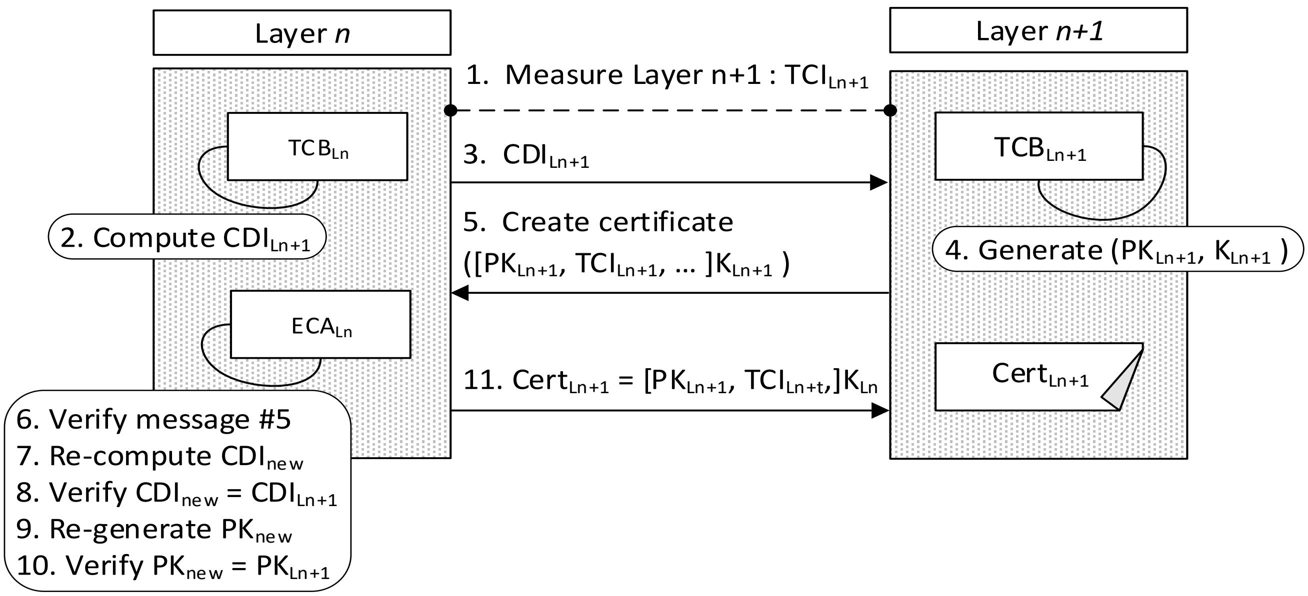

Figure 10: Layered TCB Certification using a CSR

9.2.2.3 ECA Certificate Issuance Considerations

Some ECA implementation details are out-of-scope for this specification. This section contains guidelines to ECA implementers.

(i) An ECA may not remember whether certificates have been previously issued, hence the ECA may unknowingly re-issue the same certificate if asked. It may be desirable for ECAs to select certificate attributes such as Issuer, Validity Period, Subject, or Serial Number that have deterministic properties.
(ii) Excessive use of an ECA signing key may decrease its effective lifetime due to cryptanalysis.
(iii) An ECA may issue certificates for different key usages beyond those defined by the RFC5280 KeyUsage constraint. For example, attestation usage is not defined at the time of this writing. The issuing ECA may wish to disambiguate fixed purpose certificates by adding additional common name parameters to the Subject name or by including certificate extensions that indicate an expected usage semantic.
(iv) An ECA may not be able to accept certificate revocation requests. Therefore, setting the KeyUsage cRLSign attribute to FALSE and specifying a certificate revocation distribution point (CRLDistributionPoints) extension that refers to a non-ECA entity may be appropriate.

9.2.3 Certification with External CAs

A DICE TCB layer may interact with an external CA (i.e. not an ECA) in order to obtain device identities. Device identities may be obtained during manufacturing or when the device is onboarded into a network.

Device manufacturers typically provision device identities during manufacturing and follow one of two general approaches; (i) both the device keys and device identity credential are provisioned to the device while in a non-operational state or (ii) the device keys are generated by the device and the identity credential is created in response to a credential creation request from the device.

9.2.3.1 Manufacturer Issued IDevID Certificate with Device Generated Keys

This section describes the steps for initial device identity (e.g., IDevID) creation when the manufacturer's provisioning system is not trusted to protect device secret keys.

Start of informative comment

Special manufacturer firmware may be used during manufacture of a device. The firmware has knowledge of the measurements of production firmware. After provisioning the device, the manufacturer firmware is removed and replaced with a production firmware image.

End of informative comment

The provisioning steps, illustrated in Figure 11, are:

1. The manufacturer, using a manufacturing process and possibly manufacturer provisioned device firmware, provisions the layer 0 TCB and TCB identity (TCI_{L0}).
2. The layer 0 TCB or manufacturing firmware computes its CDI value.
3. The layer 0 TCB or manufacturing firmware generates an initial device identity key pair (PK_{L0}, K_{L0}).
4. The layer 0 TCB or manufacturing firmware supplies a certificate creation request (a.k.a., CSR) to the manufacturer's CA containing its identity (TCI_{L0}) and public key (PK_{L0}). The request is integrity protected using the private key (K_{L0}) or another key used by the manufacturer.
5. The manufacturer's CA verifies the CSR signature.
6. The manufacturer's CA verifies the CSR TCI value matches the provisioned TCI value.
7. The manufacturer's CA issues an IDevID certificate using the TCI and public key.
8. The $IDevID_{L0}$ device certificate may be provisioned to the device.

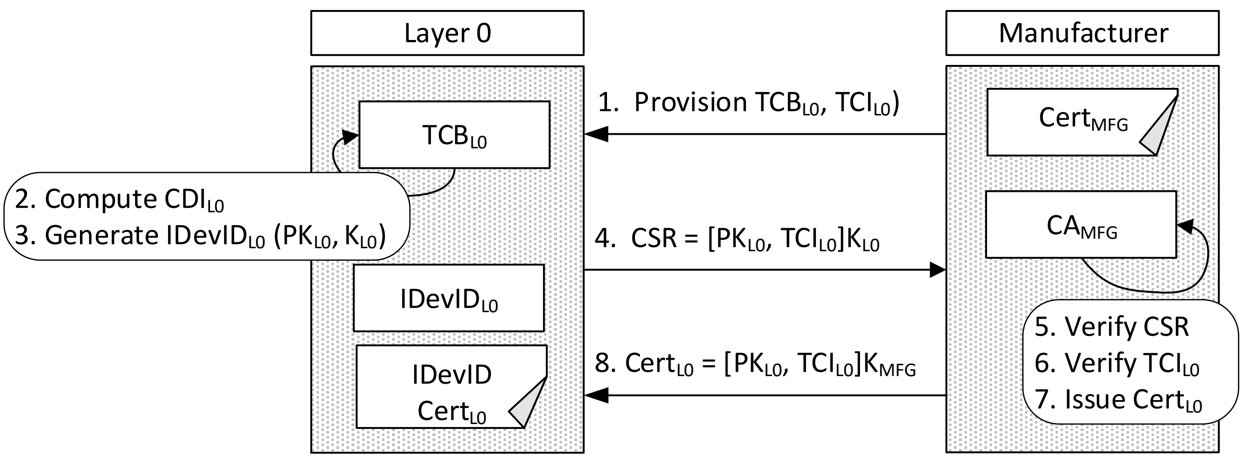

Figure 11: Example initial device identity (IDevID) certification by a manufacturer.

A traditional certificate signing request (CSR) is self-signed by the private portion of the device identity key (e.g., K_{L0}) to prove to the registration entity (e.g., CA) that it possesses the private key. However, self-signing doesn't attest the security properties employed to protect the private key. For this, manufacturing processes and firmware may be used to convey the appropriate device state.

9.2.3.2 Manufacturer Issued IDevID Certificate with Provisioned Keys

This section describes the steps for initial device identity (e.g., IDevID) creation when the manufacturer's provisioning system is trusted to protect device secret keys.

The provisioning steps, illustrated in Figure 12, are:

1. The manufacturer computes the layer 0 TCI
2. The manufacturer computes the layer 0 CDI.
3. The manufacturer generates the device identity key pair (PK_{L0}, K_{L0}).
4. The manufacturer's CA issues an IDevID certificate using the TCI and public key.

5. The manufacturer provisions the layer 0 TCB, TCB identity (TCI_{L0}), CDI, private key (K_{L0}) and $IDevID_{L0}$ certificate using a manufacturing process.

The manufacturing certificates issued by the manufacturing CA (e.g., CA_{MFG}) may contain additional information or implied semantics relating to attestation attributes. Alternatively, the manufacturer and supply chain entities may issue attribute certificates or reference manifests.

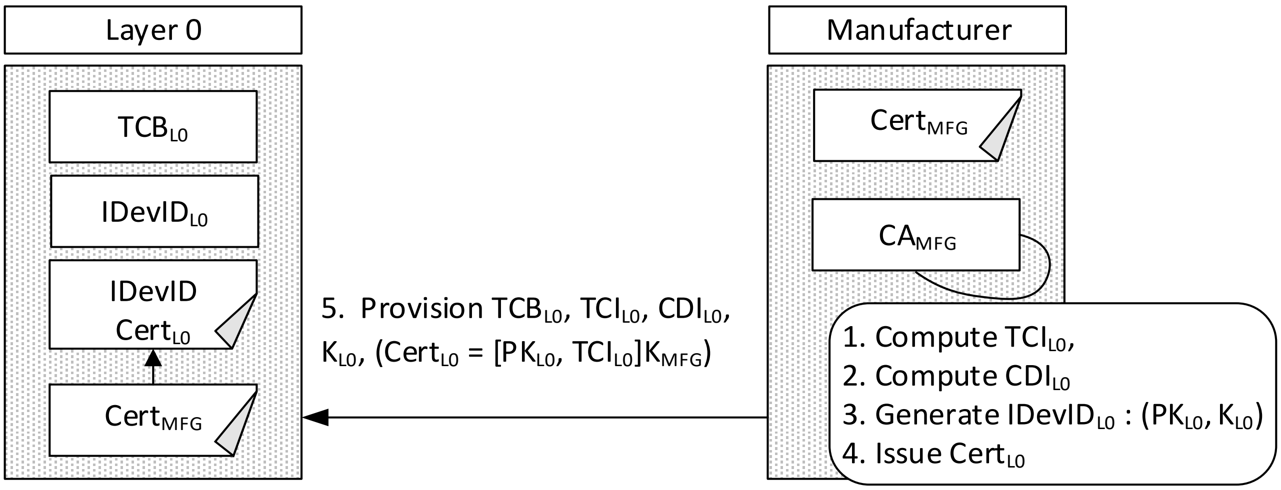

Figure 12: Example initial device identity (IDevID) certification by a manufacturer.

9.2.3.3 Owner Issued LDevID Certificate

Figure 13 illustrates an example of a device owner who may take possession of the device from a supply chain entity and issue a local device identity (e.g., LDevID) using a CA of the owner's choice. The owner may likewise include attestation attributes and issue attribute certificates and attestation manifests to facilitate attestation operations within the local owner's network.

The owner may determine a different device layer (e.g., Layer 2) is most appropriate for operation within the owner's network. Hence, a device onboarding step might create a CSR using a local (LDevID) public key (e.g., PK_{L2}). The device attests to the security of the LDevID by supplying attestation evidence of the TCB layering back to the Root of Trust layer. For example, the manufacturer issued certificate and embedded CA issued certificates that were supplied to the owner CA for trust evaluation. The owner CA will issue local device ID certificates if attestation evidence is sufficient.

The onboarding and ownership acquisition steps, illustrated in Figure 13, are:

1. The layer 2 TCB creates a CSR using a layer 2 generated public key (PKL2). Note: this example shows the TCI_{L2} as the local device identity, but the owner may select a different value.
2. The owner supplies an attestation nonce N1.
3. The layer 2 TCB attests the device by returning the nonce, certificate chain generated by the device's ECA. The attestation message is signed using a layer 2 attestation key.
4. The owner CA verifies the CSR signature.
5. The owner CA verifies the device attestation. Note: the certificate chain contains attestation evidence at each layer supplied by the ECA at each layer as part of certificate issuance.
6. The owner CA issues a local device identity ($LDevID_{L2}$) certificate.
7. The owner CA may provision the $LDevID_{L2}$ certificate to the layer 2 TCB.

Start of informative comment

The device will not be able to authenticate to the local network unless it completes booting into layer 2 where the $LDevID_{L2}$ private key resides.

End of informative comment

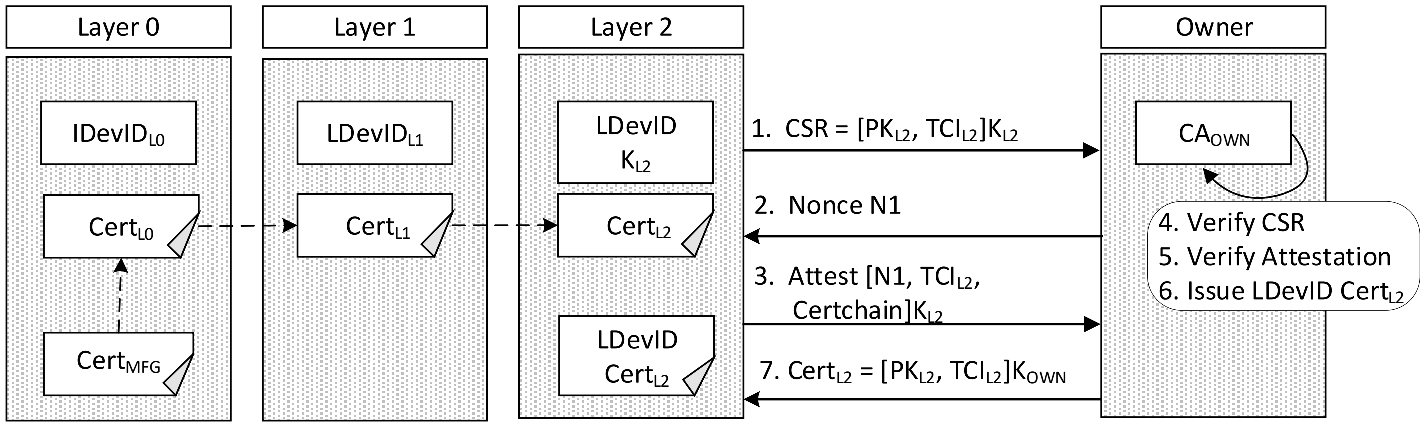

Figure 13: LDevID certification by owner CA

10 DESIGN CONSIDERATION

This section provides guidance on some of the design considerations that come with implementing a layered DICE architecture.

10.1 External Communication

Communication over a network or external bus is typically complex and not expected to be present in early layers of a DICE architecture, even if it will eventually be part of the device TCB. The DICE HRoT and layer 0 code should be kept as simple as possible. Code necessary to access a network, for example, is too complex.

10.2 Privacy

It is the responsibility of device firmware to manage device state and establish secure communication with external services. Therefore, device firmware is responsible for fulfilling user or device privacy requirements to the extent necessary, given the requirements of the environment in which a device operates.

There are design considerations in this specification that may favor infrastructure models versus consumer models. While these tradeoffs are acceptable in the context of the assumptions made here, they may not be favorable in all environments.

10.2.1 Single Cloud Infrastructure

Of the privacy-relevant design considerations, the most notable is whether a device will be expected to communicate with a single infrastructure during its operational lifetime. Cloud infrastructure may be explicitly aware of the identity of each device it services as well as attestation data, in addition to other device characteristics. There are environments in which this is not only a feature, but a necessity.

Conversely, there are device vendors and users for whom this is unacceptable. For these scenarios, device firmware must be implemented in a way that minimizes the potential for device tracking.

One strategy for minimizing tracking is by continuing the key derivation and certificate chain beyond the Alias Key and certificate or by periodically recycling an Alias Key using a local CA that hides the device certificate path but vouches for the trustworthiness of the Alias Key. This abstracts the device's hardware identity and prevents relying parties from corelating a DeviceID to multiple transactions involving the same device. The device may authorize the correlation by explicitly sharing this information with the service.

10.2.2 Factory Reset

There is often the expectation that devices can be reprovisioned to erase any existing device state including initial device identity. This specification does not preclude the implementation of a device reset mechanism.

To implement device reset, system designers have four basic options:

1. Modification of the UDS value on the device, for example, via the use of a random element to obtain a new UDS value
2. A modification or configuration change to Layer 0
3. Both 1 and 2
4. For devices with immutable UDS values and long-lived Layer 0 Code, the derivation of the DeviceID key pair cannot depend solely on the CDI value for the device. The DeviceID key pair for these devices cannot avoid including a modifiable value that can be changed as part of device identity re-provisioning.

To protect against roll back of device identity on a device that has been reprovisioned, any changes to the UDS or Layer 0 for the purpose of reprovisioning MUST be irrevocable. Each of the above options will result in a new device identity and an unrecoverable loss of the existing device identity and any derived secrets.

