REFERENCE

Implicit Identity Based Device Attestation

Version 1.0
Revision 0.93
March 5, 2018
Published

Contact: admin@trustedcomputinggroup.org

TCG Published

TCG

Disclaimers, Notices, and License Terms

Table of Contents

1. Scope

This reference describes the foundational elements for Identity-Based Device Attestation. In addition to providing a strong Device Identity rooted in hardware, Device Attestation is an extension to typical attestation schemes in that it also relies, implicitly, on a device's statistically unique, cryptographically strong, identity. This solution is compatible with IEEE 802.1AR - Secure Device Identity and is intended for devices containing a Device Identifier Composition Engine.

The approach described in this document builds on the Trusted Platform Architecture Hardware Requirements for a Device Identifier Composition Engine specification developed by the TCG Root of Trust for Measurement SG under the Embedded Systems WG.

The Implicit Identity Based Device Attestation architecture describes keys, cryptographic operations, and certificates for a cryptographic Device Attestation scheme. In addition to strong Device Identity and Device Attestation, one possible use for this architecture is as a foundation for a secure storage (Sealing) implementation in resource constrained devices.

2. Terms and Definitions

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

2.1 digest

result of a hash operation

2.2 device

highly integrated platform containing a programmable component with other optional programmable components and peripherals

2.3 DeviceID

asymmetric key pair that serves as a long-term identity for the device

Note 1 to entry: This definition includes assumptions specific to this use case document.

2.4 measurement

cryptographic hash of code and/or data

Note 1 to entry: 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. Acronyms

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

Abbreviation	**Description**
CDI	Compound Device Identifier
DICE	Device Identifier Composition Engine
FSD	Firmware Security Descriptor
FWID	Firmware Identity
IP	Internet Protocol
OID	Object Identifier
PAN	Personal Area Network
TLS	Transport Layer Security
UDS	Unique Device Secret

4. Introduction

This document describes the Device Identifier Composition Engine (DICE) use case that provides hardware-based Device Identity and Device Attestation. The approach taken in this document is to provide a description of each architectural element of this solution and then to enumerate the benefits and consequences of design decisions around these elements. The solution uses a DICE Compound Device Identifier (CDI) as a basis for Device Identity with some basic assumptions. The assumptions impose constraints on the solution. For example, this use case assumes the Device Identity will be represented cryptographically as an asymmetric key pair so the public portion can be freely shared while the private portion remains secret. The private portion of this key is used to prove the device's identity. The benefit of limiting the number of assumptions is that it maximizes the set of Device Identity and Attestation scenarios this reference supports.

Even though this document is intended to enable as many different scenarios as possible, it is still necessary to make some assumptions about the environment in which this use case may be implemented. This document assumes:

1. Devices following this reference are connected to, and capable of communication over, some form of network. For devices that are not IP-capable, it is assumed network connectivity is achieved via a Personal Area Network (PAN) and gateway.
2. Infrastructure and data external to the device (that enables device management, update, attestation, etc.) is based on/protected by asymmetric (public key) cryptography. Note that this does not preclude the use of symmetric cryptography to encrypt communication.
3. Individual devices are associated with a single infrastructure or cloud provider with knowledge of the device architecture. This assumption simplifies the end-to-end key derivation and use of certificate chains but involves disclosing information about a device's identity that could be used to track the device. For scenarios in which this is not a desirable tradeoff, this document discusses options for ensuring privacy sufficient for enabling devices to safely communicate with multiple service providers during their operational lifetime.
4. Devices implementing this use case are powerful enough to generate (derive) a key pair and signature. Implementation requirements for specific cryptographic algorithms are outside the scope of this document.

This use case presumes DICE support in hardware. How the Unique Device Secret is provisioned within a device is not in scope; only that it has been provisioned.

Finally, because there is a clear advantage in relieving device manufacturers and vendors of the burden of maintaining secret databases of UDS values, this architecture presents a solution that does not require secret databases of UDS values.

5. Architecture

DICE is the root of trust for measurement for this architecture. It must be inherently trusted because its misbehavior cannot be detected. This architecture relies on DICE unconditionally generating the correct CDI for layer 0. Layer 0 is the next link in the chain of trust. The purpose of DICE in this architecture is to establish that the device booted the First Mutable Code provided by the manufacturer. This enables detection of persistent modification of Layer 0 and above.

Figure 1 provides an illustration of an overall architecture for this use case.

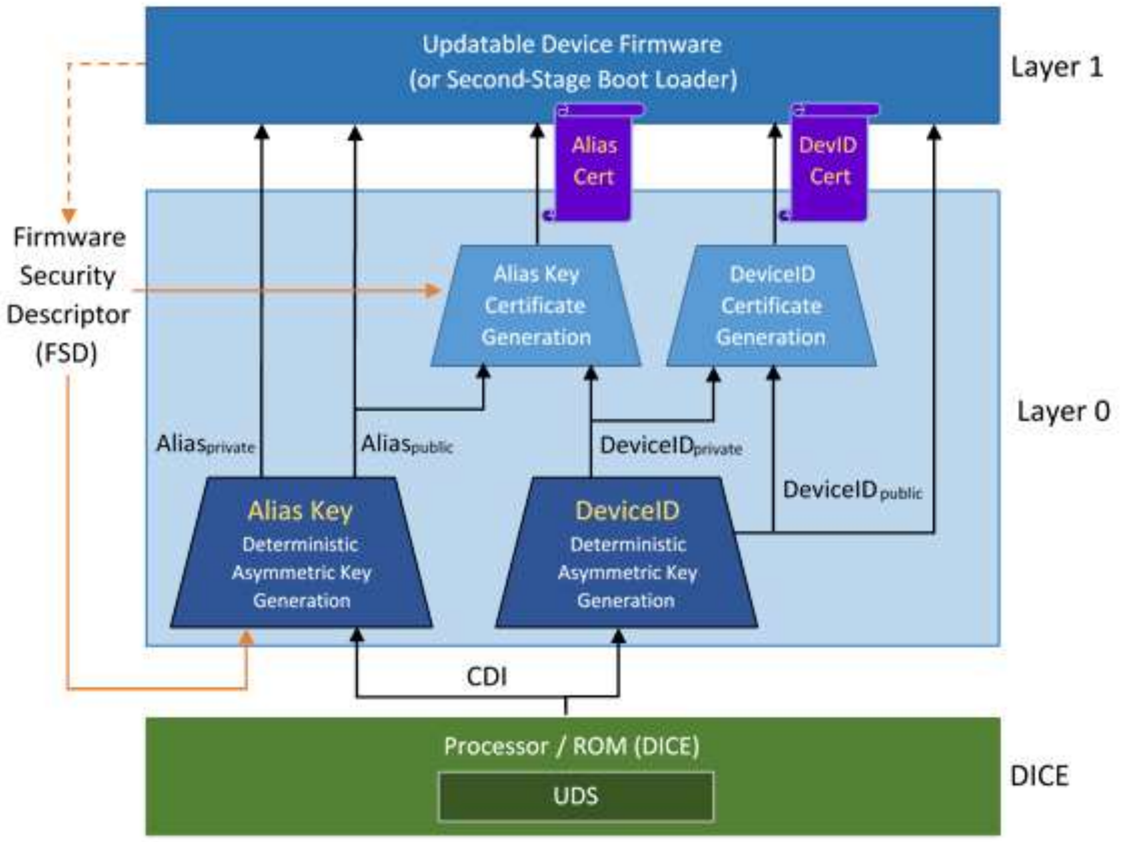
Updatable Device Firmware
(or Second-Stage Boot Loader)
Layer 1
Alias Cert
DevID Cert
Firmware Security Descriptor (FSD)
Alias Key Certificate Generation
DeviceID Certificate Generation
Layer 0
Alias$_{private}$
Alias$_{public}$
DeviceID$_{private}$
DeviceID$_{public}$
Alias Key Deterministic Asymmetric Key Generation
DeviceID Deterministic Asymmetric Key Generation
CDI
Processor / ROM (DICE)
UDS
DICE

Figure 1: Implicit Identity Based Device Identity Architecture

The diagram provides detail for First Mutable Code (Layer 0) because this layer is responsible for constructing the foundational identity and attestation elements upon which Device Firmware (Layer 1) relies. The DICE layer is described by the Device Identifier Composition Engine specification.

This use case assumes that Layer 0 will provide the outputs shown in Figure 1. Note that Layer 1 does not provide input to Layer 0. Instead, Layer 0 measures Layer 1. In this reference, the Firmware Security Descriptor (FSD) is simply the Device Firmware image. An FSD may be a vendor-defined machine-readable data structure that represents the identity of Layer 1. First Mutable Code calculates the digest of this Layer 1 identity. This digest is referred to as the Firmware Identity (FWID).

6. DICE

The DICE, or Device Identifier Composition Engine, is a hardware/firmware capability that generates a cryptographically unique value, called the Compound Device Identifier, based on a Unique Device Secret and the measurement of the First Mutable Code that runs on a platform. This is represented by the base layer in Figure 1.

6.1 Purpose

The Device Identifier Composition Engine provides a way to know what mutable code is running on a specific platform. This is essential for strong Device Identity. This strong Device Identity and the DICE approach to protecting secrets and keys, also provides the foundation for Attestation and Data Protection (Sealing).

6.2 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 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 NV storage to which the DICE can restrict access. See the DICE specification for details.

6.3 Compound Device Identifier (CDI)

The DICE measures the device's First Mutable Code. This measurement is combined with the device-specific UDS to form the Compound Device Identifier. The CDI is unique not only to the device, but to the combination of the device, the cryptographic identity of the device's First Mutable Code and, optionally, configuration data.

6.4 Implementation

This Device Identity and Attestation solution is intended to support any DICE implementation that is compliant with the Device Identifier Composition Engine specification.

7. First Mutable Code (Layer 0)

After it has executed, the DICE transfers control to the first serviceable code that runs on a device. A platform's First Mutable Code should be kept very small and simple so that it may be kept free of exploitable bugs. This is important because an update to First Mutable Code on a device results in a new Device Identity (and loss of the existing Device Identity). Since it is desirable for Device Identities to be long-lived, First Mutable Code should be of sufficient quality that servicing this layer is unnecessary. A future use case will detail recovery strategies in which Device Identity can be retained despite potential updates to this early software layer.

7.1 Device Identity Key Pair (DeviceID)

During boot, the device's First Mutable Code receives the CDI from the DICE. First Mutable Code uses the CDI to derive the Device Identity asymmetric key pair. Since the derivation is based on the CDI, the DeviceID keys are based not only on the UDS but also the cryptographic identity of the device's First Mutable Code.

The DeviceID key pair may be first derived at manufacture, during which time the public portion of the DeviceID may be read from the device. A manufacturer may also choose to certify the DeviceID key. Certification of the DeviceID key is discussed in the next section.

The private portion of the DeviceID is never exposed outside of the First Mutable Code where it is derived. Further, First Mutable Code must erase any plaintext portions of the CDI value and DeviceID private key from memory, registers, etc., before control of the device is transferred to the next boot layer.

7.1.1 Options for Device Identity Key Pair

The following sections describe design decisions for derivation, persistence, and provisioning of the Device Identity Key Pair.

7.1.1.1 Key Generation

This use case assumes the Device Identity is an asymmetric key pair. There is no requirement on the algorithm used. However, ECC is usually favored for its cost/performance benefits.

7.1.1.2 Retaining Secret Values

Knowledge of a device's UDS or CDI value would allow a third party to derive the DeviceID key pair for that device. This could enable an attacker to impersonate the device and/or access device secrets. Due to the risk of disclosure, retention of UDS or CDI values for devices during manufacture is not recommended. While there may be specific scenarios or protocols that secret-value retention may enable, the risk of disclosure may exceed this benefit. Therefore, there is no requirement for knowledge or retention of UDS or CDI values by the manufacturer or vendor in this DICE architecture.

7.1.1.3 Persistence

It is not a requirement that a device derive the DeviceID key pair on every boot cycle. Some devices may not have sufficient processing power to derive the DeviceID on each boot and still maintain acceptable boot times.

Instead of deriving the DeviceID on each boot, a device may use a value based on the CDI (e.g. the product of a one-way function that takes the CDI as input) as a symmetric key to decrypt the persisted, encrypted, DeviceID key pair. On the first boot, the device would encrypt and persist the DeviceID. It would not need to derive the DeviceID key pair again until/unless the First Mutable Code is updated, resulting in a new CDI (and, therefore, a different Device Identity key pair).

One trade-off for a system designer who chooses this option is an increase in the storage requirement for the device (to contain the persisted DeviceID key pair). Choosing to persist the DeviceID key pair will also increase complexity if a determination must be made as to the correctness of decrypted values.

DeviceID persistence is of greatest benefit in environments where the identity of a device is expected to be static. Updating device firmware, in particular, a device's First Mutable Code, is out of scope for this document and will be addressed in a future use case.

7.2 Alias Key Pair (Certified Identity)

Proof of knowledge of the private portion of the DeviceID can be used as a building-block in a cryptographic protocol to identify the device, independent of Device Firmware. However, using the DeviceID directly has disadvantages. First, all traces of the private portion of the DeviceID key must be erased by First Mutable Code before control of the device passes to a loader or other application firmware. This means the private portion of the DeviceID must only be used during execution of First Mutable Code. This is not ideal. Further, it is important to limit the potential exposure of the DeviceID key where possible.

To achieve this, First Mutable Code also generates a new asymmetric key pair that it can make available to Device Firmware. This key pair is referred to as the Alias Key. The First Mutable Code may then use the DeviceID key to certify the Alias public key and pass the Alias Key pair and generated certificate that demonstrates that the key was generated securely, to Device Firmware.

The private portion of this key pair can be used during runtime by Device Firmware, rather than just during the early boot phase by First Mutable Code, like the DeviceID private key. The Alias Key pair and certificate can be safely passed to Device Firmware to be used in protocols to authenticate the device and its firmware.

It is the responsibility of Device Firmware to protect the private portion of the Alias Key pair as well as any potentially sensitive data in the Alias Key certificate. Compared to the DeviceID private key, the risk of exposure of the Alias Key is less severe because updating Device Firmware effectively re-keys a device. If an Alias Key was exposed due to a software bug, for example, then updating Device Firmware (to fix the bug) would result in a new Alias Key pair and certificate without impacting the underlying Device Identity.

7.2.1 Options for Certified Identity

The following sections discuss design decisions for the derivation, persistence, and certification of the Alias Key Pair.

7.2.1.1 Key Generation

There are few options for derivation of the Alias Key pair. The Alias Key pair is derived deterministically, using a standard KDF, from the CDI and the identity of Device Firmware. If one were to remove the dependency on the CDI this would decouple the Alias Key from the identity of the device. Removing the dependency on the identity of Device Firmware eliminates Device Attestation as a potential feature of the device. While there may be environments in which system designers favor privacy over all else, system implementers should consider the following two factors:

1. First Mutable Code is generally not changed unless the intent is to establish a new identity for a device. If the keys and certificates created by First Mutable Code do not permit Device

Attestation, for example, then it would not be possible to enable this feature at any point in the future without loss of identity.

2. Device Firmware is updatable. This is important because:
    a. Recovering from potentially exploitable software bugs is possible in Device Firmware.
    b. One may rely on firmware update to enable or disable communication of Device Identity and Attestation information to relying parties.

It is preferred that First Mutable Code follows the derivations outlined in this document.

Finally, it is assumed that the DeviceID-certified Alias Key pair will be of the same type as the DeviceID itself. Again, while not a requirement, ECC is favored for resource constrained devices.

7.2.1.2 Persistence and Provisioning

Derivation of the Alias Key is deterministic based on the CDI and a measurement of the Firmware Security Descriptor describing Device Firmware. In this scenario, the Alias Key and certificate will not remain static. Update of Device Firmware will effectively re-key a device, hence, the Alias Key and its certificate are likely to be shorter lived than the DeviceID key and its certificate.

Despite this, there may be some specialized scenario in which collection and retention of the Alias Key and certificate during manufacture would be of value. For example, scenarios where Device Firmware is also non-serviceable.

7.3 Certificates

Alias Keys and DeviceID Keys are certified with X.509-format certificates. In this reference document, the primary use of the DeviceID key is to certify Alias Keys. Fields in the generated Alias Key certificate also specify the FWID and other information to enable Device Attestation.

General purpose X.509 certificate creation represents a challenge to reducing complexity because software libraries that enable generation and manipulation of X.509 certificates are often large and complex. This is a further complicating factor for devices that are optimized for small code size.

To address this challenge, this reference recommends the use of a subset of code for building the Alias Key certificate specifically, instead of general-purpose encoding libraries. The following sections outline properties of Alias Key and DeviceID certificates. In addition to the properties outlined in this document, implementations of this use case should also adhere to RFC 5280.

7.3.1 DICE Certificate Extension

First Mutable Code is responsible for creating and certifying Alias Keys. In addition, First Mutable Code also includes an extension in the Alias Key certificate that authenticates the Device Firmware that it boots. First Mutable Code encodes the Firmware Identity (FWID) of Device Firmware within this field. The FWID is the digest of the vendor-specified data structure that describes Device Firmware. This data structure is referred to as the Firmware Security Descriptor (FSD). In the simplest case, an FSD may simply be the Device Firmware image itself. Meaning the FWID would then be the digest of the Device Firmware Image.

In Alias Key Certificates, an example of an OID-tagged X.509 extension that contains this data is called the TCG-DICE-fwid. This extension is used to convey the computed FWID for the device to relying parties. The FWID represents the identity of Device Firmware running on the device. An example for how this certificate extension may be formatted is as follows:

```
TCG-DICE-FWID ::== SEQUENCE
{
      TCG-DICE-fwid    OBJECT IDENTIFIER,
      SEQUENCE         CompositeDeviceID
}

CompositeDeviceID ::==  SEQUENCE
{
      version          INTEGER,
      SEQUENCE         SubjectPublicKeyInfo,
      SEQUENCE         FWID
}

FWID ::== SEQUENCE
{
      hashAlg          OBJECT IDENTIFIER,
      fwid             OCTET STRING
}
```

The SubjectPublicKeyInfo field contains the same key, parameters, and encoding as the Subject Public Key Info field of a DeviceID certificate.

7.3.2 Serial Number Generation

Certificate Serial Numbers distinguish different Alias Key certificates from one another. As a result, the Serial Number field should be statistically unique for each Alias Key certificate.

7.3.3 Certificate Lifetime

Devices with a secure local clock can use the clock to set the validity period of the Alias Key certificate issued by First Mutable Code. Conventionally, devices without a secure clock use generalized time values in the distant future to ensure the validity of the certificates they create. If a secure clock is unavailable, then devices can set the Not After portion of the certificate’s Validity Period to the X.509-defined GeneralizedTime value of 99991231235959Z. This value is used to indicate that the certificate has no defined expiration date.

In practice, devices implementing this use case will either “re-certify” the Alias Key on each boot (because the device is also re-creating the Alias Key Pair on each boot) or the Alias Key persists if the Device Firmware remains unchanged. In either scenario, it is assumed that expiration of the Alias Key certificate coincides with update of Device Firmware.

7.3.4 Subject Name and Issuer Name

The use of these fields in Alias Key or DeviceID certificates is not constrained by this use case reference. These fields may, for example, contain alternate representations of the DeviceID, FWID, or other data. Refer to RFC 5280 for rules and guidance on the interaction of the Subject and SubjectAlternativeName fields when both are present in a certificate.

7.3.5 Alias Key Certificate

Alias Key certificates can be created with the following extensions and constraints. Fields not included in the examples below may follow RFC 5280.

Field Name	**Contents**
Subject Alternative Name	Appropriate additional value associated with this device
Subject Public Key Info	Alias public key and algorithm
Signature Algorithm and *Signature Value*	DeviceID algorithm and public key
Key Usage	Appropriate for the cryptographic protocol in use
Extended Key Usage	Appropriate for the usage model, e.g., id-kp-clientAuth for clients
Basic Constraints	Not present, pathLengthConstraint is absent and cA is absent

7.3.6 DeviceID Certificate

Device manufacturers or system integrators can create DeviceID certificates using external public key infrastructure with, for example, the following fields.

Field Name	**Contents**
Subject Public Key Info	DeviceID public key and algorithm
Key Usage	keyCertSign is asserted Key usage appropriate for the cryptographic protocol in use
Basic Constraints	cA:TRUE pathLengthConstraint:0 (or as appropriate)
Signature Algorithm and *Signature Value*	Vendor public key and signature

7.3.7 DeviceID Certificate Signing Requests

First Mutable Code can also create PKCS#10 certificate signing requests for DeviceID keys.

8. Device Firmware (Layer 1)

This document refers to the code that receives control of the platform from the First Mutable Code as Device Firmware. The function of Device Firmware beyond Device Identity and Attestation is outside the scope of this reference document.

8.1 Keys and Certificates

Device Firmware is provided the following by First Mutable Code:

1. The public portion of the DeviceID Key pair and Device ID certificate
2. The public and private portions of the Alias Key Pair
3. The Alias Key certificate signed by the DeviceID private key

8.2 TLS

Devices should be able to authenticate themselves to services and establish secure communications. Further, devices need to be able to attest to their security configuration. These goals are achieved by encoding Device Identity and Attestation information in X.509 certificate fields.

In this use case, devices can be authenticated with TLS client certificates. One advantage of this approach is that TLS is very widely deployed. This simplifies the adaptation of this solution within existing infrastructure.

In this use case DeviceID Keys are vendor-certified during manufacture. The Alias Key certificate chains back to the DeviceID certificate and, ultimately, to the vendor certificate authority. Without a vendor-certified DeviceID Key, the certificate chain would end with the DeviceID. In some scenarios, this may be sufficient.

8.3 Design Considerations

This section provides guidance on some of the design considerations that come with implementing Device Firmware within an Implicit Identity Based Device Attestation architecture.

8.3.1 Update

Since new Device Firmware will have a new FSD and FWID, rebooting with new Device Firmware will result in a new Alias Key Pair. This further results in a new Alias Key certificate that will encode the new FWID for this device. The DeviceID will remain unchanged.

This has the effect of re-keying the device on each update. The obvious benefit of this is that when a flaw in Device Firmware that could potentially expose Alias Keys or other sensitive data is discovered, simply updating firmware securely re-provisions its keys.

8.3.2 Network Communication

It is expected that Device Firmware will be the first software layer that can access a network. It would be inadvisable for layers preceding Device Firmware to have this capability as, during their execution, they are in possession of some form of private data. Further, DICE and First Mutable Code should be kept as simple as possible. Code necessary to access a network does not fit this definition.

8.3.3 Protocols

This use case supports TLS client authentication, but use of TLS is not essential. Further, this use case does not assume or require authentication of server certificates. This implies that devices will be associated with a single infrastructure or cloud provider with knowledge of the device. If desired, devices implementing this use case may implement authentication of server certificates and limit the identity and attestation information in the Alias Key certificate to address privacy concerns when dealing with multiple infrastructure providers.

8.3.4 Privacy

It is the responsibility of Device Firmware to manage device state and establish secure communication with external services. Therefore, Device Firmware may be involved in 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 reference that may favor infrastructure models versus consumer models. While these tradeoffs are acceptable in the context of the assumptions made in this reference (see section 4), they may not be favorable in all environments.

8.3.4.1 Single Cloud Infrastructure

Of these design considerations, the most notable is that devices will communicate with a single infrastructure during their operational lifetime. Cloud infrastructure is explicitly aware of the identity of each device it services as well as attestation data and any 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 so it cannot be tracked, since it is necessary that devices have the ability to safely communicate with multiple unrelated relying parties.

One way implementers of this reference can achieve this is by continuing the key derivation and certificate chain beyond the Alias Key and certificate. This would have the effect of further abstracting the device's hardware identity and preventing relying parties from making the connection between a particular DeviceID and a specific device, unless the device shares this information with the service explicitly.

8.3.4.2 Access Control Strategies

Device Firmware can further choose to predicate access to an underlying Device Identity based on satisfaction of some access control policy or other criteria. While this is outside the scope of this document, system implementers are encouraged to carefully consider all aspects of device security and privacy in implementing solutions based on a DICE architecture.

8.3.4.3 DeviceID Reprovisioning

There is often the expectation that devices can be reprovisioned to erase any existing device state, and resultantly, Device Identity. This use case 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 First Mutable Code (Layer 0)
3. Both 1 and 2
4. For devices with immutable UDS values and long-lived First Mutable Code, the derivation of the DeviceID key pair cannot not 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 reprovisioning.

To protect against roll back of Device Identity on a device that has been reprovisioned, changes to the UDS or First Mutable Code will 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.