TRUSTED® COMPUTING GROUP

SPECIFICATION

TPM 2.0 Keys for Device Identity and Attestation

Version 1.00
Revision 12
October 8, 2021

Contact: admin@trustedcomputinggroup.org

PUBLISHED

DISCLAIMERS, NOTICES, AND LICENSE TERMS

CONTENTS

Acknowledgements

The TCG wishes to thank those members and others who contributed to this specification.

Name	Affiliation
Carolin Baumgartner (Co-chair)	Carolin Baumgartner
Monty Wiseman (Co-chair)	General Electric
Tom Laffey (Editor, Co-chair)	Hewlett Packard Enterprise
Graeme Proudler	Graeme Proudler
Henk Birkholz	Fraunhofer SIT
Dave Challener	JHU/APL
Michael Eckel	Huawei Technologies
Guy Fedorkow	Juniper Networks
Ken Goldman	IBM
Bill Jacobs	Cisco Systems
Suzanne Leicht	United States Government
Stanley Potter	United States Government
Bill Sulzen	Cisco Systems
Ludovic Jacquin	Hewlett Packard Enterprise
Ned Smith	Intel
Chris Fenner	Microsoft
Davide Rutigliano	Politecnico Di Milano
Brandon Weeks	Google

1 Acronyms

AIK	Attestation Identity Key, a TPM 1.2 key type
AK	Attestation Key. A key that can sign TPM-internal data
ASN.1	Abstract Syntax Notation One
Binding	A cryptographic link between items, e.g. a certificate and key to a device by means of a CA signature.
BIOS	Basic Input/ Output System
CA	Certificate Authority
CP	Certificate Policy
CPS	Certificate Practice Statement
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
DOS	Denial Of Service
DevID	A Device IDentity certificate. Refer also to IDevID and LDevID.
ECDSA	Elliptic Curve Digital Signature Algorithm
EH	TPM Endorsement Hierarchy
EK	Endorsement Key
HMAC	Hashed Message Authentication Code
IAK	An OEM or Manufacturer provided AK Key and certificate. The IAK certificate contains the public attestation key and the certificate subject matches the IDevID certificate.
IDevID	An Initial (factory installed) Device IDentity certificate
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task-Force
IKE	Internet Key Exchange
IPSec	Internet Protocol Security
Local CA	A CA that issues LDevID certificates. This is typically an Enterprise or delegated CA and is not an OEM CA.
LAK	Local Attestation Key or Certificate. A field-installed certificate containing a public attestation key, with the certificate subject matching the LDevID certificate
LDevID	A Locally significant (field-installed) Device IDentity certificate
KDF	Key Derivation Function
MITM	Man-In-The-Middle
OEM	Original Equipment Manufacturer
OID	Object IDentifier
PCR	Platform Configuration Register, a TPM resource
PH	TPM Platform Hierarchy
PICS	Protocol Implementation Compliance Statement (from IEEE 802.1AR)
PKCS	Public Key Cryptographic Standard
PKI	Public Key Infrastructure
PKINIT	Public Key Cryptography for Initial Authentication in Kerberos
POP	Proof-Of-Possession
RA	Registration Authority. A front-end to a CA.
RFC	IETF Request For Comment
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman public key cryptosystem
SH	TPM Storage Hierarchy
SRK	Storage Root Key, a TPM key type
TCG-CSR	A Device Identity CSR that includes TCG-specific OIDs and data
TLS	Transport Layer Security
TPM	Trusted Platform Module

TSS	TPM Software Stack
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

2 Introduction

2.1 Scope and Audience

This specification applies the IEEE Standard for Local and Metropolitan Area Networks, Secure Device Identity (802.1AR) [1] device identity module definition and formatting to keys protected by a TPM 2.

Secure device identities (“DevIDs”) that are not easily spoofed or copied from memory or storage are of paramount importance in secure network deployments. This specification assumes an intention to protect Device Identities (as defined below) against any compromise that could lead to a “lying endpoint”. Digital identity certificates are assigned to a single entity and must not be transferrable. This specification describes methods for remotely proving key attributes that are expected to be performed by a Certification Authority.

A TPM has capabilities to protect keys against compromise and misuse over a product’s lifetime. This specification addresses ways to incorporate TPM 2 created keys into solutions that protect device identities and help prevent a “lying endpoint”. It addresses how the resulting device identities interface with, and are represented within, an existing public key infrastructure.

Proving that a key belongs to a specific device (or platform) requires binding of that key to the device’s TPM using carefully constructed protocols. By signing IDevID certificates, an OEM CA makes an assertion that is a primary security dependency for later DevID certificate creation. OEM signed certificates are definitive evidence to applications that need proof that a key belongs to a specific device.

This specification applies only to TPM 2 [2]. (For TPM 1.2, refer to the TCG Specification “TPM Keys for Platform Identity for TPM 1.2” [3].) This specification addresses which TPM hierarchies and keys are to be used, how they are reflected in an X.509 [4] certificate and how consuming parties will be able to recognize and use device identities to authenticate to a network using existing authentication protocols.

This specification acknowledges that various certificates using TPM 1.2 may already exist. Every effort is made to maintain compatibility and use of existing certificates and relying party functionality. However, TPM 2 introduces cryptographic agility to the TPM library with the suite of available cryptographic algorithms expanded beyond those supported by TPM 1.2. The cryptographic options available with TPM 2 are expected to change over time. Other than implicitly recognizing this ongoing change process, it is a goal of this specification that no new requirements be imposed on a relying party, thereby allowing TPM based certificates to be used wherever a digital certificate is used in common protocol practice.

Architects, designers, developers, and technologists interested in the development, deployment, and interoperation of trusted systems will find this document essential when making use of TPM based keys within certificate structures and across enterprise networks as identities for devices. This specification refers to TPM commands only and is expected to be useful to implementers of a TPM Software Stack (TSS) or to other software developers writing software directly interfacing to the TPM. For higher level interfaces, consult the TCG TSS Specification [5].

Before reading this document, the reader should review and understand the IEEE 802.1AR standard as published in [1].

Note that placing a TPM in a product does not imply that the product must be provisioned with IDevID certificates. The need for IDevID provisioning is both product and use case dependent.

2.2 TPM Specification Version

This specification assumes use of a TPM 2 compliant to revision of TPM 2.0 Library Specification Rev 1.38 or later. For purposes of illustration, this specification will refer to TPM 2.0 Library Specification, Rev 1.38 [2] when making reference to the TPM Specification.

2.3 Definition of Terms

This section defines terms used in the context of this specification.

Acronyms are explained in section 1.

2.3.1 Key

When used in this document, the term “key” will often refer to the public key of an asymmetric key pair, either RSA or ECC. Where the private key is being referred to, the term private key will be used. Where a symmetric key is being referred to, the term symmetric key will be used.

2.3.2 Certificate

When used in this document, a certificate is an X.509v3 digital certificate. A certificate includes a public key (or “key” as described above), data about the “subject” (identity) conveyed and a signature over the entire structure signed by some entity. Digital certificates, as used here, are ASN.1 structures assembled using DER encoding rules.

An entity validating a certificate will generally need a chain of certificates in order to verify a chain of trust to a trust anchor—often a root certificate. When discussing this chain, the term “certificate chain” will be used. Otherwise the chain is assumed to be present but not in scope for the current discussion.

2.3.3 Credential

When used in this document, a credential is a combination of a private key, accessible for the required purpose (e.g. creating a signature) and a certificate, which binds the corresponding public key to a subject (identity) by means of a signature. Thus an entity that has a credential is able to present a certificate as well as use the key to prove that they possess the corresponding private key, usually to perform authentication.

A private key without a certificate is not a credential. A certificate without the corresponding private key is not a credential.

This definition is consistent with that used in IEEE 802.1AR [1], section 6.4, Trust Model.

2.3.4 Identity

When used in this document, an identity is the combination of the X.509 certificate fields Subject and subjectAltName in a certificate issued to a device.

Identities are unique within the scope of the Certificate Authority signing the certificate. Additionally, in an IDevID/IAK, each of Subject and subjectAltName SHOULD independently be unique in the scope of the Certificate Authority. This allows flexibility in creating a unique LDevID/LAK derived from an IDevID/IAK.

2.3.5 Keywords

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [6].

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 does not require any action).

End of informative comment

3 Helpful Information

3.1 Brief Summary of IEEE 802.1AR

Start of informative comment

The IEEE 802.1 working group defined the IEEE 802.1AR Standard for Local and Metropolitan Area Networks Secure Device Identity [1]. That standard defines a per-device unique identity installed at manufacturing time and used subsequently in device-to-device authentication exchanges. This standards-based device identity can also be coupled in multiple ways with user identification.

The 802.1AR standard defines a secure device identifier (DevID) as “an identifier that is cryptographically bound to a device”. It further specifies:

the Identifier consists of:

an X.509 public key certificate, containing

a globally unique-per-device identity, and bound to

a unique-per-device secret (private key) capable of creating a signature

management and cryptographic binding of a device to its identity

the relationship between an initially installed identity and subsequent locally significant identities, and

interfaces and methods for use of DevIDs with existing and new provisioning and authentication protocols

The 802.1AR specification also specifies a conceptual DevID module containing a service interface, storage holding a DevID secret and certificate, secure hashing functions, a random number generator (RNG) and asymmetric cryptographic functions.

End of informative comment

3.2 Use of Terms – IDevID/IAK, LDevID/LAK, DevID

Start of informative comment

The 802.1AR standard defines two Device Identity (DevID) types, depending on the CA signing the issued certificates and expectations for certificate lifetimes.

The initially installed identity is defined as an IDevID/IAK (“I” for initial) and is installed by the product OEM. The IDevID credential is intended to be usable for the life of the product. The IDevID/IAK is expected to be created at device manufacturing time.

On the other hand, owner-created and signed identities are named LDevID & LAK (“L” for local). LDevID/LAK credentials may “overlay” IDevID/IAK credentials, thereby replacing IDevID with LDevID in operation. Alternatively, IDevID/IAK and LDevID/LAK may be used for different purposes. LDevID/LAK certificates are not expected to be long-lived certificates. LDevID/LAK credentials are expected to be removed when a device is “zeroized” or is at its end of life.

Where the distinction of which CA signs the certificate is inconsequential, this specification uses DevID as a general term to mean any of IDevID/IAK or LDevID/LAK.

The IEEE standard anticipates only keys used for a general signing operation. Additional keys and certificates are needed in order that TCG-specific operations such as Attestation and proof of a key’s TPM protection can be performed. This is discussed further in section 3.5.

End of informative comment

3.3 Why use a TPM for DevID?

Start of informative comment

Factory-installed Device Identity (IDevID/IAK) credentials are intended to be present on a device for the lifetime of that device. This means that DevID private keys must be available and protected for a device's lifetime. The TPM provides comprehensive protections that satisfy these requirements.

A fundamental aspect of a Device Identity is the ability to use that identity for network authentication, with the remote entity (the relying party) having confidence that the device is what it is represented to be. Should a device identity private key be transferrable from one device to another, then there is no assurance that the associated certificate belongs to, and identifies, that one device and only that one device. The TPM eliminates this concern (when keys are created as described in this specification), because it is a secure Root of Trust for Storage (RTS) that provides protection for private keys, preventing use of keys from one device on another device or with another TPM.

Some devices require privacy controls for use of device identity credentials. Proper operation and privacy on these devices require controls that enable a device owner, administrator or user to determine when a DevID key is available for use. A TPM's Authorization and Policy controls satisfy this requirement because they are a comprehensive technical means of restricting the use of keys stored in a TPM.

Some devices may need to be transferred from one user or application to another. In these circumstances, it is desirable to "zeroize" the device, removing all data and key material from the first user before transferring the device to a second user. Devices with a TPM can be provisioned with IDevID from the OEM such that the IDevID key can be regenerated after typical zeroization procedures. This allows the device to participate in zero-touch provisioning for the second user in the same manner as afforded to the first user.

End of informative comment

3.4 TPM 2 Key Terminology

This document uses the following terminology related to TPM-based keys:

- An Attestation Key, or an AK, is a non-duplicable Restricted signing key. A certificate associated with this key will be referred to as an AK Certificate. The manufacturer installed AK certificate is referred to as an Initial AK (IAK) certificate, while a user installed certificate is referred to as a Local AK (LAK) certificate.
- A key used strictly for Device Identity (a signing-only key) is called a Signing key. A certificate associated with this key is referred to as a Signing Certificate.
- A key that can be used in a context combining identity and encryption, historically called a "Legacy" Key, is called a Combined key in this document. A certificate associated with this key will be referred to as a Combined Certificate.

  There are deprecated protocols still in use today that require asymmetric decryption. A Combined key can be used for both decryption and signing operations and may be the right choice in some rare cases. However, this is not recommended as it is safer (and better) to choose a signing key[1].

- A key that is created only for encryption (a TCG storage key) is not used in a DevID context.
- A non-duplicable key is a key created by a TPM and not usable outside of that TPM. This specification uses only non-duplicable keys.

[1] US NIST SP800-57 [13] in section 5.2 does not allow the same key to be used for both decryption and signing, and recommends that applications not share keys.

3.5 Number of TPM-Protected Keys for DevID

In order to enforce best security practices, a TPM places restrictions on a key's use, so multiple keys and certificates are likely to be needed when operations such as device attestation or key certification are included in a product's use cases.

As a result of key creation decisions made in support of different use cases, it may be that an implementer needs to create more than one DevID key. That will, of course, lead to more than one DevID certificate.

Section 6 of IEEE 802.1AR [1] describes a single IDevID certificate but allows for multiple DevID certificates to be issued to a device as specified. That section states "A DevID module includes an IDevID for each of the signature suites (Clause 9) that it supports." Further, IEEE 802.1AR allows for multiple LDevIDs (section 1).

As described in section 3.8 of this specification, the TPM can create and use keys that are either Restricted keys or non-Restricted keys. This distinction leads directly to splitting DevID operation into two parts. The first part is a non-Restricted signing key and the associated certificate that is useful for general device authentication purposes, such as a TLS client certificate. The second part is a Restricted signing key and the associated certificate that is useful for key certification (proving that a new key is in the same TPM) or for Attestation (signing TPM-internal PCRs or audit sessions).

A Restricted signing key may only sign a digest that has been produced by the TPM, so a Restricted signing key is primarily used for operations where the TPM signs its internal data. When a digest is signed by a Restricted signing key, there is no ambiguity about whether the signed data is a forgery of the TPM's internal data. Using an Unrestricted signing key to sign TPM internal data enables spoofing, as there is no proof that the data is, in fact, in the TPM.

The unique value, TPM_GENERATED_VALUE, begins a TPM-generated attestation structure. This value prevents a Restricted key from signing a forgery of an attestation because a TPM will not allow a Restricted signing key to sign any external data beginning with this unique value.

Table 1 illustrates how the "xDevID" and "xAK" terms are used in this specification.

Identity Certificate Type	Signing Key	Attestation Key	Creator
IDevID	X		OEM
IAK		X	OEM
LDevID	X		Non-OEM
LAK		X	Non-OEM

Table 1 - DevID Keys and Creators

The DevID software implementer must decide whether their application requires one, two, or more keys. This specification assumes that at least two keys are likely to be required to support most applications. The first key is an Attestation Key (AK) used to certify that the second key (either a DevID signing key or DevID legacy key) is held by the same TPM. Key applications and key properties are discussed further in later sections.

When IAK and IDevID certificates are issued to a device, they MUST include (i.e. describe) the same device identity and the same TPM. OEM issued IDevID certificates (both signing and attestation) SHOULD have matching notBefore and notAfter dates and times.

Certificate policy OIDs are specified in section 8.2 to allow the relying party to distinguish TPM related key attributes as well as TPM dependent actions of the issuing CA.

3.6 TPM Key Hierarchies

As described in the TPM 2.0 Library Specification [2] Part 1, Section 13 "TPM Control Domains", the TPM has three control domains, each with their own key hierarchy under which a new key may be created. These are:

- Endorsement Hierarchy
- Platform Hierarchy
- Storage (“user”) Hierarchy

Hierarchy selection affects the authorizations and policies available for the lifecycle of the key. Selection of key hierarchy and key type is of particular importance for IDevID and IAK keys as these keys are required to persist after a device has been cleared of user-created keys. If a device is cleared, it is expected that it is then in a factory default state ready for redeployment to another user, potentially requiring an IDevID or IAK for participate in zero-touch or secure remote deployment scenarios.

All three TPM key hierarchies are enabled after a TPM reset and may need to be configured on each boot. The platform boot firmware must implement hierarchy management by enabling/disabling each hierarchy as configured or as required for a platform type. The Platform Hierarchy authorization value (platformAuth) must be configured and managed as required for the platform.

Unrelated to the above, the TPM 2.0 Library Specification [2] Part 1, Section 23 “Hierarchical Relationship between Objects” describes parent-child relationships between keys (key hierarchies). These key relationships can also affect key administration and key usage authorizations.

3.7 TPM Key Types, Options and Restrictions

A TPM based key can be created in one of three ways:

1. A Primary key [TPM 2.0 Library Specification [2], Section 4.56], which is created by the TPM based on the current Primary Seed when executing the TPM2_CreatePrimary command. A Primary key, unless persisted within the TPM, must be recreated before it can be used after a TPM reset.

   IDevID and IAK keys created by an OEM SHOULD be Primary keys.

2. An Ordinary key [TPM 2.0 Library Specification [2], Section 4.44] produced with a seed taken from the TPM RNG when executing the TPM2_Create command. An Ordinary key may be persisted within the TPM or it may be persisted external to the TPM in the form of an encrypted key blob. This blob is only loadable by the TPM that created it. An ordinary key must be the child of another key and loading it requires using the parent key’s authorization.

   LDevID & LAK keys SHOULD be Ordinary keys.

3. In TPMs implementing version 1.38 or later of the TPM 2.0 Library Specification, the TPM2_CreateLoaded command may be used to create and load either of the key types above. This command does not return creation data, so would preclude use of the creation ticket methods recommended by this specification for an IDevID/IAK. The TPM2_CreateLoaded command method MUST NOT be used for creating IDevID/IAK keys but MAY be used for LDevID/LAK keys.

Some decisions on key attributes must be made in addition to Primary/Ordinary and persistence before creating a key in the TPM. These selections are retained by the TPM as part of the key itself and checked by the TPM when performing operations using the key.

The TPM can work with both symmetric and asymmetric keys. Only asymmetric keys are of interest for X.509 certificates (and hence for DevID) so use of asymmetric keys is assumed in this specification.

For User Devices, appropriate consideration for privacy must be provided as described in section 11.

Key hierarchies need to be managed at each boot. Refer to section 3.6 for details.

3.8 Introduction to TPM 2 Key Types

TPM 2.0 Library Specification [2], Section 25 describes base attributes and other attributes of a key. These attributes are important to a CA evaluating security properties of TPM based keys as they identify the purpose for which a key may be used. These attributes are briefly summarized here for convenience.

Restricted: A key with this attribute has restrictions on the Sign and Decrypt operations, limiting those operations to TPM generated data. Refer to the TPM 2.0 Library Specification [2], Part 1 for more information on the Restricted attribute and the TPM_GENERATED_VALUE field.

Sign: A key with this attribute set is allowed to create a signature.

When combined with the Restricted attribute, the key may only sign a digest that has been produced by the TPM. As described in section 2.3, this specification refers to a Restricted signing key as an Attestation Key, since this is how this restriction is used within the DevID context.

When both the Sign attribute and the Decrypt attribute are set, a Combined key is created.

Decrypt: When used alone, the decrypt attribute is used for TPM binding operations. This use is outside the scope of this specification.

A key with both the decrypt attribute and the Restricted attribute is known as a Storage Key (a key for protecting other keys). Such a key is not usable as a DevID.

When both the Decrypt attribute and the Sign attribute are set, a Combined key is created.

Before creating a Combined key, consider the following:

- US NIST SP800-57 [7] in section 5.2 does not allow the same key to be used for both decryption and signing, and recommends that applications not share keys.
- A key that can be used for both signing and decryption cannot be a Restricted key—it cannot be used to sign TPM internal data.

For a comparison of TPM characteristics to TPM key attributes, refer to the TPM 2.0 Library Specification [2] Part 1, Table 25.

3.9 Introduction to TPM 2 Key Attributes

Additional attributes of importance for identity and attestation certificates are described here. For more information, refer to the TPM 2.0 Library Specification [2] sections related to TPMA_OBJECT.

These attributes are important to a CA evaluating security properties of TPM based keys as they identify the key properties related to duplication (i.e. migration), key Administration and key Use authorizations.

FixedTPM: this attribute indicates that the key cannot be duplicated.

FixedParent: this attribute indicates that the key may be copied or migrated only if the parent key may be duplicated. A key that is fixedTPM will also be fixedParent.

EncryptedDuplication: If a key is allowed to be duplicated, indicates that the private data area may not be exported from the TPM in the clear.

SensitiveDataOrigin: this attribute indicates that the private key was generated by the TPM. A TPM-generated IDevID/IAK key guarantees that the OEM is known never to have possession of the IDevID/IAK private key. This assurance that there is no other copy of the key provides added assurance for prevention of spoofing, privacy, etc.

UserWithAuth: this attribute refers to key use policy and indicates that the object's authValue may be used to provide the USER role authorizations for the object. If this attribute is CLEAR, then USER role authorizations may only be provided by satisfying the object's authPolicy in a policy session. Refer to section 3.10 for more details.

AdminWithPolicy: This attribute indicates that authorization for an action requiring the ADMIN role requires that the authPolicy of the object be satisfied. If this attribute is CLEAR, then the authValue may be used in an HMAC session to perform operations that require ADMIN role. Any ADMIN role action may be authorized with a policy session that satisfies the authPolicy. Refer to sections 3.10 for more details.

3.10 Key Authorizations

When it's intended that applications use DevID keys without Platform Owner or User involvement (i.e. authorization is controlled with Policy) the key MUST have the userWithAuth bit CLEAR and the Platform Manufacturer MUST create the required use and admin policy values when creating the key.

When applications use DevID keys with User involvement, the key MUST have the userWithAuth bit SET with the user selecting an authorization value when creating the key.

Applications using only Policy to control key administration MUST SET the adminWithPolicy attribute when creating the key. When adminWithPolicy is CLEAR, the authValue may be used in an HMAC session to perform Admin operations.

There is no loss of security if the OEM creates IDevID and IAK keys in all devices with the same authorization values, provided OEMs follow this specification's recommendation to ship devices with IDevID and IAK certificates and not supply the actual IDevID and IAK keys.

3.11 EK Certificate Identifies the TPM

Because the Endorsement Key (EK) has a long lifetime and is persistent, using it as the DevID may initially appear logical. However:

- TPM Endorsement Keys (in both TPM 1.2 and TPM 2) are Storage Keys, not signing keys.
- The EK and EK certificate identify a TPM, while a DevID certificate identifies a device.

For more information on TPM EKs, refer to the TCG EK Certificate Profile [7], Section 2.1 for a summary of TPM EK characteristics.

4 IDevID/IAK CA Requirements

Issuing a DevID certificate binds together a fixedTPM DevID key and device information using a CA signature. In practice it may be a Registration Authority that processes device identity certificate requests. Regardless of authority configuration, the issuer of device identity certificates will be referred to as a Certificate Authority (CA) in this specification.

A CA issuing LDevID/LAK certificates may have practices that follow IDevID/LAK enrollment practices or may use less strict practices for evaluating the chain of proof. The requirements in this section are intended for OEMs issuing IDevID/IAK certificates to devices. Those same requirements may be adopted for CAs issuing LDevID/LAK certificates according to the CA's policies and practices.

An IDevID/IAK supports non-repudiation of device transactions while an LDevID/LAK can support privacy of a user. There are cases where non-repudiation is a required property. This can be the case in infrastructure devices where the device must always be positively identified and the situations described in section 11 do not apply. IDevID/IAK certificates meet the non-repudiation requirements of IETF RFC 4211 [8], section 9.

Because of its important security and identity role, the OEM CA MUST be diligent in verifying all aspects of TPM binding to the device and verifying key Attributes and Policies. The OEM MUST follow the method set forth in section 7.2 (IDevID/IAK Hierarchy Selection) on platforms intended for mass market operating systems and those platforms required to support Privacy protections as described in section 11.

Details of issued certificates are described in Section 8, "Certificate Fields in Detail." The identity provisioning descriptions in this section include requirements specific to each use case. General requirements are:

- A Certificate Authority (CA) MUST verify TPM residency of a key before signing a certificate that includes TCG OIDs described in section 8.2. TPM data in the CSR MUST be evaluated before issuing certificates with TCG-specific CPS OIDs described in section 8.2.
- The CA SHOULD support a standard certificate transport protocol that provides protection from replay attacks and provides for confidentiality and integrity. An example of such a transport protocol is Enrollment over Secure Transport (EST) [9].

4.1 Verifying TPM Protection of an Attestation Key

An AK MUST have the following characteristics:

- Restricted
- Signing
- Not-decrypting
- FixedTPM

The CA must receive key data from the device sufficient to make the TPM-residency assessment, regardless of the transport method used.

4.2 Verifying TPM Protection of a DevID Key

A DevID Signing MUST have the following characteristics:

- Not-Restricted
- Signing
- Not-decrypting (unless a Combined key)
- FixedTPM

The CA must receive key data from the device sufficient to make the TPM-residency assessment, regardless of the transport method used.

5 DevID Enrollment Use Cases

Start of informative comment

Depending on the manufacturing processes used to create a platform, the platform's application and the user's privacy requirements, different enrollment use cases arise for DevID certificates.

Proving that a new key belongs to a specific device requires first binding the TPM to the OEM device and then binding an AK to a TPM using the TPM's EK. An AK certificate is the trust anchor for certificates created later, as it is the certificate that ties a new key, generated after the AK, to the same TPM containing both keys. By signing an IAK certificate, the OEM CA makes an assertion that is a primary security dependency for later DevID certificate creation. An OEM-supplied IAK certificate is a definitive assertion to applications that need proof that the IAK belongs to a specific device.

The following enrollment use cases state the specific requirements and preconditions to create the desired binding. The use cases are structured as "what is created" with the implementation "based on what is available" (or used). This means that trust for enrolling a new certificate is "based on" a prior step (an existing certificate) and assurance for the new certificate is based on the previously enrolled certificate. The trust chain for each enrollment case is shown in the related dependency figure.

End of informative comment

5.1 OEM Creation of an IAK Certificate

It is expected that the OEM-provided IAK certificate will typically be the trust anchor for enrollment of other DevID certificates, especially infrastructure devices where the privacy concerns of section 11 are unlikely to apply. Alternatively, a device's owner or user may be provided the means to create their own LAK certificate using the TPM binding created by a platform certificate, see section 5.6.

OEMs may provide a means to remotely enroll a device or to perform device attestation using factory provisioned certificates. Creation of an IAK certificate is described in section 6.1.

Figure 1 illustrates this use case.

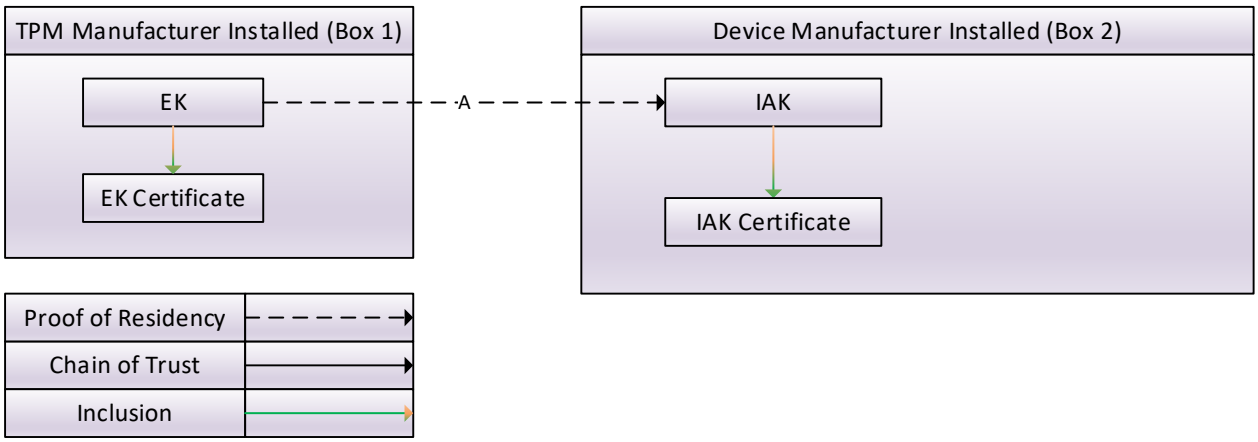

Figure 1 – Dependency of IAK key and certificate on TPM EK

In this use case, the OEM creates an IAK certificate that binds an AK to the device identity (e.g. model and serial numbers). The key is proven to be contained in the TPM described by the EK certificate.

- Box 1 shows the EK and EK Certificate resident in the TPM as installed by the TPM manufacturer. The EK certificate is signed by the TPM Manufacturer and binds the EK to a specific TPM from that TPM manufacturer.

- Box 2 shows the IAK certificate installed by the OEM.
    - Line A illustrates that the IAK created during the enrollment is verified by the OEM CA to be resident in the same TPM as the EK using the methods of this specification.

The IAK certificate is signed by the device OEM's CA.

5.2 OEM Creation of IAK and IDevID in a Single Pass

Infrastructure OEMs may create the IAK and IDevID certificates at the same time, potentially creating a more efficient work flow. This procedure is described in section 6.2 with dependencies show in Figure 2.

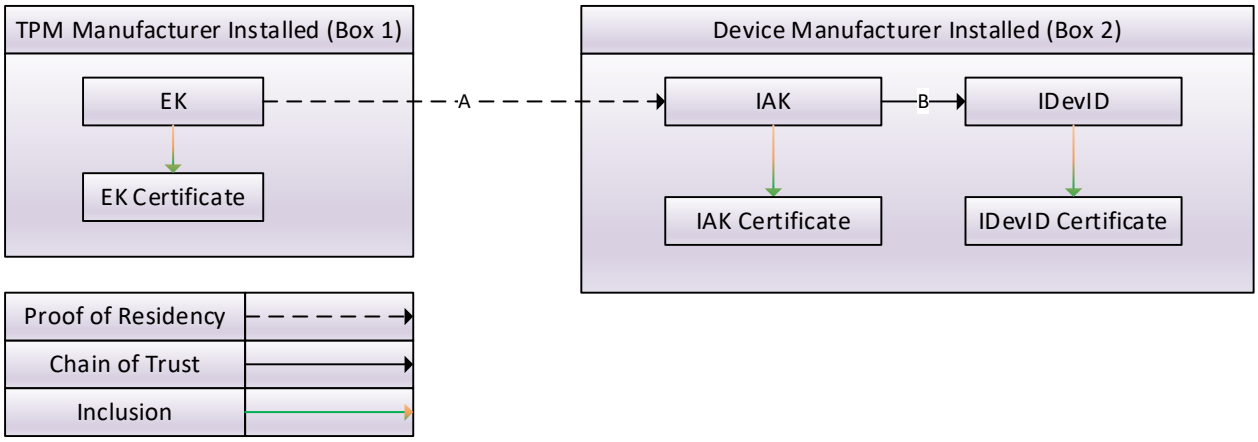

Figure 2 – Dependency of IAK and IDevID on TPM EK

In this use case, the OEM creates certificates binding IDevID & IAK keys to the device identity (e.g. model and serial numbers). The IAK key is proven to be contained in the TPM described by the EK certificate and the IDevID key is proven to be in the same TPM as the IAK.

- Box 1 shows the EK and EK Certificate resident in the TPM as installed by the TPM manufacturer. The EK certificate is signed by the TPM Manufacturer and binds the EK to a specific TPM from that TPM manufacturer.
- Box 2 shows the IDevID and IAK certificates installed by the OEM.
    - Line A illustrates that the IAK created during the enrollment is verified by the OEM CA to be resident in the same TPM as the EK using the methods of this specification.
    - Line B shows that the IDevID key has been verified, using the methods of this specification, to have the correct key properties and be resident in the same TPM as the IAK.
    - Both IAK and IDevID certificates are signed by the device OEM's CA.

5.3 Owner Creation of a LAK Certificate based on an IAK Certificate

A device that provides an IAK certificate usually allows the user to overlay (functionally, i.e. temporarily replace) the OEM installed IAK certificate with a Locally-signed AK certificate. Creation of this certificate is described in section 6.3, with the key and certificate relationships shown in Figure 3, below.

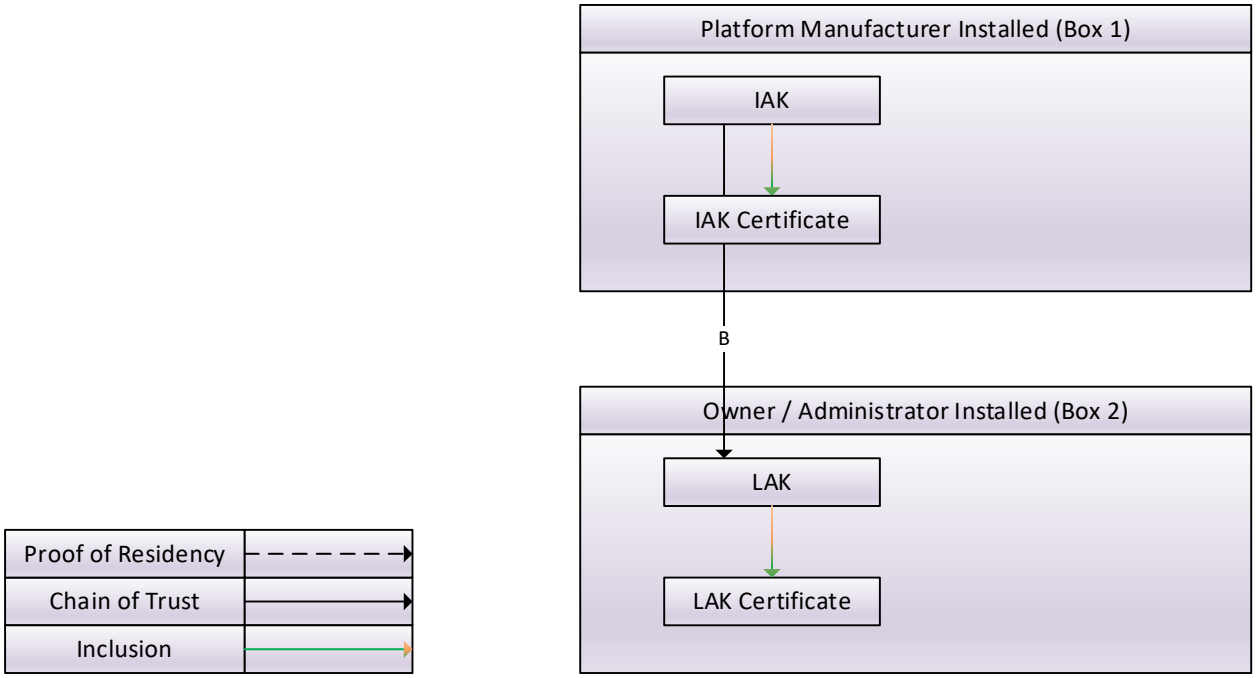

Figure 3 – Relationship of LAK to IAK certificates

This use case uses the factory installed IAK to prove that the new LAK is in the same TPM as the IAK and therefore is on the device identified in the IAK certificate.

- Box 1 shows the IAK key and certificate installed by the OEM.
- Box 2 shows the LAK key and certificate created by the device owner.
    - Line B shows that the LAK key has been verified, using the methods of this specification, to have the correct key properties and be resident in the same TPM as the IAK.

The IAK certificate is signed by the device OEM's CA and the LAK certificate is signed by the Local (owner) CA.

5.4 Owner Creation of an LDevID Certificate based on an IAK Certificate

A device that provides an IAK certificate can use that certificate to prove that a key created for a Locally-signed Device Identity certificate is TPM resident on that device. Creation of this certificate is described in section 6.4 with dependencies shown in Figure 4.

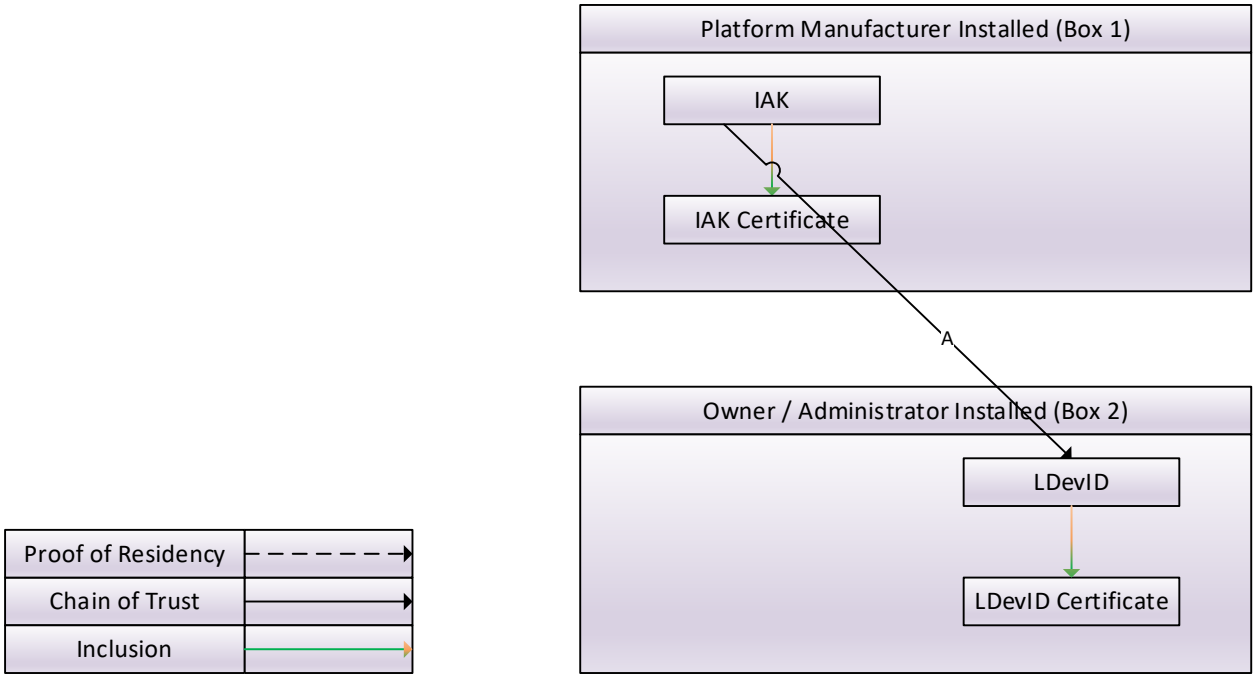

Figure 4 – Relationship of LDevID to IAK

This use case uses the factory installed IAK to prove that the new LDevID key is in the same TPM as the IAK and therefore is on the device identified in the IAK certificate.

- Box 1 shows the IAK key and certificate installed by the OEM.
- Box 2 shows the LDevID key and certificate created by the device owner.
    - Line A shows that the LDevID signing key has been verified, using the methods of this specification, to have the correct key properties and be resident in the same TPM as the IAK.

The IAK certificate is signed by the device OEM's CA and the LDevID certificate is signed by the Local (owner) CA.

5.5 Owner Creation of an LDevID Certificate based on LAK Certificate

After installing a Locally-signed AK Certificate, a user can use the issued LAK certificate to remotely create a Locally-signed DevID certificate using the method described in section 6.5 with dependencies shown in Figure 5.

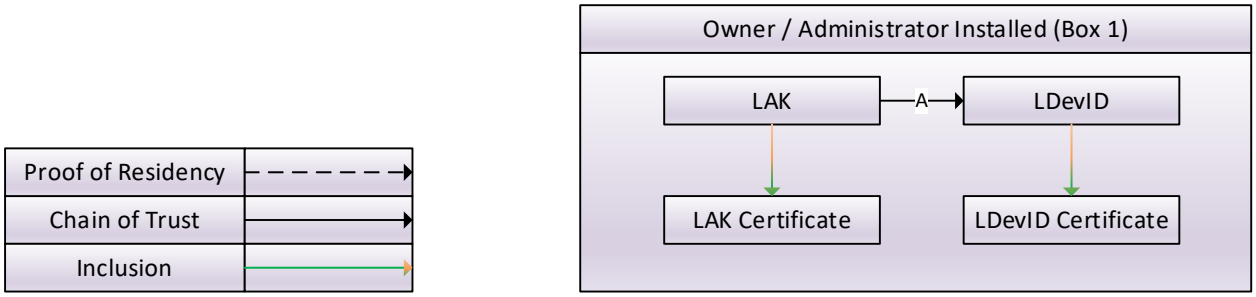

Figure 5 – Installing an LDevID certificate based on existing LAK certificate

This use case uses the owner installed LAK to prove that the new LDevID key is in the same TPM as the LAK and therefore is on the device identified in the LAK certificate. Note that the assurance provided using this method depends on the trustworthiness of the CA issuing the LAK certificate.

4. Box 1 shows the LDevID key and certificate installed by the device owner based on trust in the LAK key and certificate.
    - Line A shows that the LDevID signing key has been verified, using the methods of this specification, to have the correct key properties and be resident in the same TPM as the LAK.

The LAK and LDevID certificates are both signed by the device owner's CA.

5.6 Owner Creation of a LAK Certificate based on a Platform Certificate

A Platform Certificate can be thought of as a signed manifest in certificate form. The Platform Certificate binds the TPM (and other installed components) to the device by means of an OEM signature. A platform certificate does not contain a key, so it is not a DevID certificate. An OEM may provide a platform certificate in addition to or instead of IDevID/IAK certificates. Platform certificates are defined in the TCG Platform Attribute Credential Profile [10].

As shown in Figure 6, a Platform certificate may be used during the LAK certificate enrollment process to provide OEM evidence that a remote device's identifying information as well as its TPM are bound to the specific device instance.

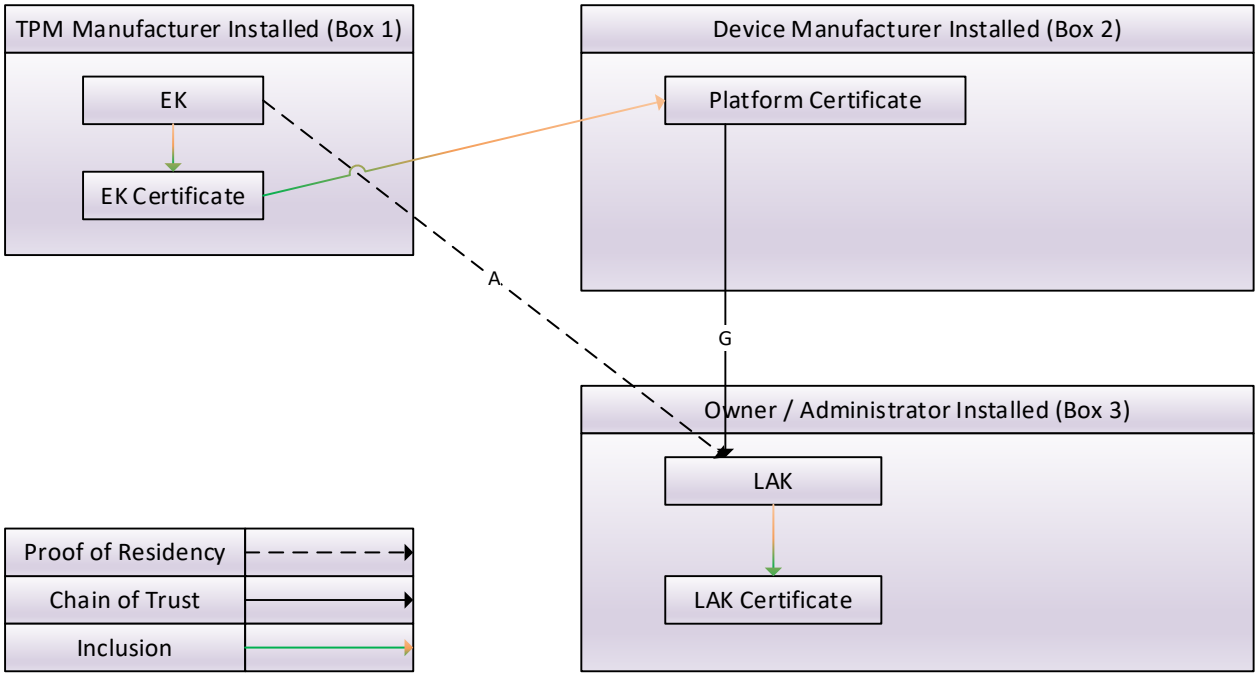

Figure 6 – Creating LAK Certificate using the Platform Certificate

The TPM EK Certificate serial number included within the Platform Certificate is used to bind the TPM to the device being issued an LAK certificate. During LAK Certificate enrollment, the user utilizes the same proof process that an OEM would use (i.e. as described in section 6.1) to prove that the LAK is in the same TPM as the EK and is therefore bound to the device identity (e.g., model and serial number) provided in the OEM signed Platform Certificate.

- Box 1 shows the EK and EK Certificate resident in the TPM as installed by the TPM manufacturer. The EK certificate is signed by the TPM Manufacturer and binds the EK to a specific TPM from that TPM manufacturer.
- Box 2 shows the Platform Certificate installed by the OEM.
    - The Platform Certificate provides signed data about the device, including a reference to the EK certificate. The device OEM signature on the Platform Certificate provides assurance that the TPM is installed on a specific device.
- Box 3 shows the Owner / Administrator installed LAK certificate. This use case results in the same LAK and LDevID certificates as shown in section 5.3, but uses a different proof chain during certificate enrollment. In this case, enrollment depends on the TPM EK during the proof and uses the OEM assertion in the Platform certificate that the TPM belongs to the specific platform identified in the Platform Certificate.
    - As line G indicates, information is extracted from the Platform Certificate. This provides device details, including information on the specific TPM installed in the device.

- Line A illustrates that the LAK created during the enrollment is verified by the Owner's CA to be resident in the same TPM as the EK using the methods of this specification.

In some cases, the TCG specified Platform Certificate does not contain the serial number of the device, as this field is optional. To be used with this specification, the Platform Serial Number, shown as OPTIONAL in the Platform Attribute Credential Profile [10], MUST be included for use in creating LDevID/LAK certificates when an IDevID/IAK conforming to this specification is not provided along with the Platform Certificate.

5.7 Summary of EK, IDevID/IAK and LDevID/LAK Relationships

Figure 7 illustrates the relationship between certificates installed by the different entities when the device OEM has installed IDevID and IAK certificates.

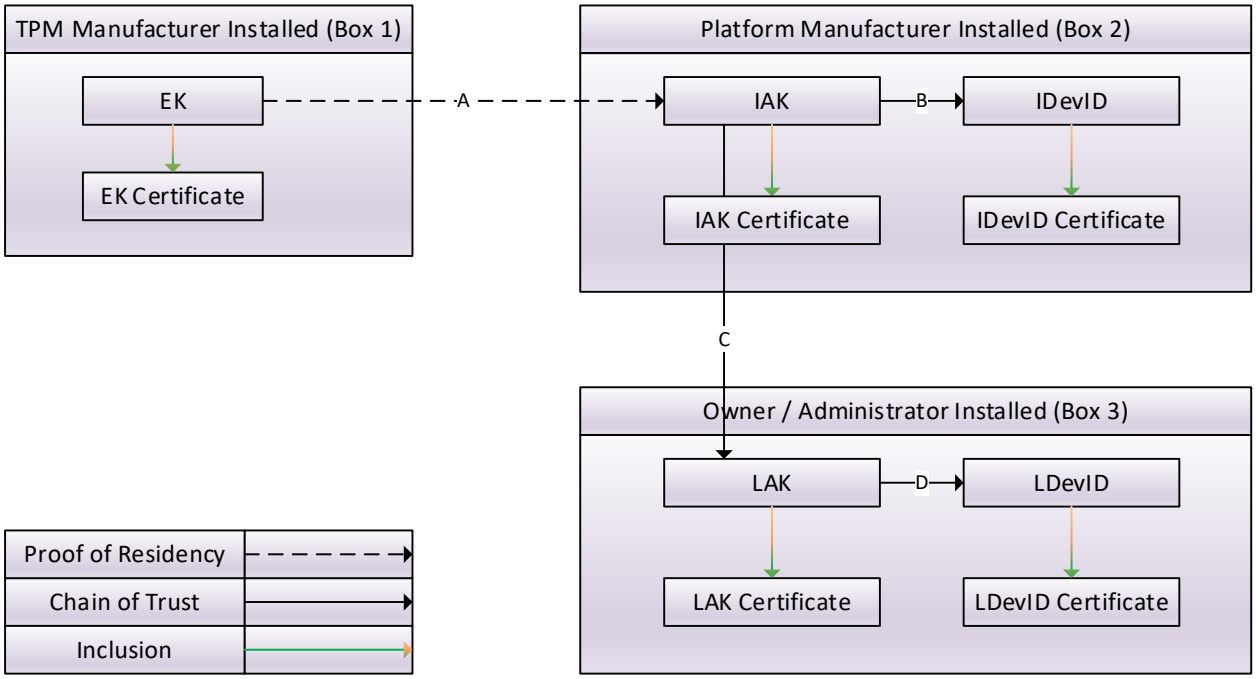

Figure 7 – Summary of Relationships of IDevID/IAK Keys and Certificates

- Box 1 shows the EK and EK Certificate resident in the TPM as installed by the TPM manufacturer. The EK certificate is signed by the TPM Manufacturer and binds the EK to a specific TPM from that TPM manufacturer.
- Box 2 shows the IDevID and IAK certificates installed by the OEM as described in section 5.2.
  - Line A illustrates that the IAK created during the enrollment is verified by the OEM CA to be resident in the same TPM as the EK using the methods of this specification.
  - Line B shows that the IDevID key has been verified, using the methods of this specification, to have the correct key properties and be resident in the same TPM as the IAK.
  - Both IAK and IDevID certificates are signed by the device OEM's CA.
- Box 3 shows Owner / Administrator installed certificates as described in sections 5.3 and 5.4. This use case uses the OEM CA assertion that the IAK belongs to a specific platform (identified in the IAK Certificate) to prove that the new LAK key is held by the same TPM and thus represents the same platform.
  - As line C indicates, the LAK can be remotely verified by the owner PKI CA to be in the same TPM as the IAK using the methods of this specification. The IAK certificate shows which device this key resides on.

- Line D shows that the LDevID key is proven to be in the same TPM as the LAK using the methods of section 5.3.

5.8 Unsupported Use Cases

This specification applies only where X.509 certificates are created using keys created or protected by a TPM for remote device authentication. Use of keys created outside of the TPM is out of scope for this specification. Authentication methods that do not use X.509 certificates or that do not use a TPM are not addressed by this specification.

6 Identity Provisioning

Start of informative comment

This section describes methods for creation of DevID certificates that maintain a cryptographic evidentiary chain linking the DevID to a specific TPM. A provisioning organization will use these methods to:

- Create and enroll an Attestation Key and Certificate.
- Create and enroll one or more DevID Key(s) and Certificate(s). The option for more than one DevID key and certificate is described later.

In the following Identity Provisioning sections, the OEM is the entity provisioning IDevID/IAK certificates and the Enterprise or Owner provisions LDevID/IAK certificates on their device. These are expected to be the common roles and deployment models for DevID certificates.

Details of provisioning recoverable IDevID and IAK keys are described in section 7.2. The following descriptions assume that IDevID and IAK keys are provisioned as described in section 7.2.3.

End of informative comment

6.1 OEM Creation of an IAK Certificate

As noted in section 5, an AK certificate is the trust anchor for most other certificates, as it is the only certificate that ties a particular AK to a specific device instance. While a device's owner or user might be provided the means to create their own AK certificate (using a platform certificate, for example, section 6.6), it is expected that the OEM-provided IAK certificate will typically be the trust anchor for enrollment of DevID certificates, particularly infrastructure devices where the privacy concerns of section 11 are unlikely to apply.

Proving that a key belongs to a specific device requires binding of the AK key to a TPM using the EK and then TPM binding to the OEM device. By signing an IAK certificate, the OEM CA makes an assertion that is definitive evidence for later DevID certificate creation. An OEM supplied IAK certificate is important to other applications that need proof that a key belongs to a specific device.

Combining an initial attestation with IAK enrollment is described in Section 6.1.4, "Included Attestation Variation".

6.1.1 Requirements

Software under control of the OEM MUST be able to create keys as described in section 3.7 and retrieve the TPM's EK certificate. The software must be able to securely create and sign a TCG-CSR-IDEVID (refer to section 13.1).

The OEM CA MUST validate TPM key certification data as signed by the IAK using methods described in the procedure below.

IEEE 802.1AR requires that use of IDevID MUST be enabled by default and it MUST be possible for the user to disable it. In the context of TPM keys, use of IAK keys is enabled and disabled by controls on use of a Hierarchy (e.g. enabling or disabling the Endorsement Hierarchy) or by use of TPM User and Admin authorizations of DevID keys.

For maximum assurance when creating a TCG-CSR-IDEVID, the "Included Attestation Verification" of section 6.1.4 is RECOMMENDED for attestation of device state while performing the enrollment procedure. The device SHOULD be freshly booted before IAK enrollment.

Details of issued certificates are described in Section 8, "Certificate Fields in Detail."

6.1.2 Procedure

The Procedure (below) provides the following assurances:

A. A TPM is authentic (by validating the EK certificate).

B. This specific TPM is authentic and is represented by the EK certificate (the EK private key resident in this TPM corresponds to the public key in this EK certificate.)
C. This AK is authentic (the AK is resident in the same TPM as the EK private key corresponding to the public key in this EK certificate.)

The CA's Certificate Practice Statement SHOULD state how the CA performs the operations above.

This procedure does not include the option to perform an initial attestation during IAK enrollment. That option is described in section 6.1.4 (Included Attestation Variation).

OEMs may want to consider whether they have requirements for endpoint authentication for factory based enrollment processes. Such requirements are outside the scope of this specification.

Detailed Procedure:

1. Platform software creates the IAK, a Restricted, fixedTPM, fixedParent Primary signing key in the Endorsement Hierarchy as described in section 3.7 and section 7.2.
2. Platform software builds the TCG-CSR-IDEVID structure as described in section 13.1 including:
    a. Platform Identity Information. This is the non-TPM device information needed to build the 802.1AR certificate. This includes the device model and serial number encoded within the TCG-CSR-IDEVID prodModel and prodSerial fields.
    b. The TPM EK Certificate.
    c. The IAK Public Area (TPMT_PUBLIC structure of the AK).
3. Platform software applies a signature to the TCG-CSR-IDEVID by first using the TPM to create the signature hash then using the IAK to sign that signature hash. This signature serves as an integrity check of the device information when in transit from the device to the CA, preventing undetected modification by a MITM. Note that using the TPM to create the hash is required in order to create a signature with the IAK.
4. The device software sends the TCG-CSR-IDEVID to the CA.
5. The CA verifies the received data:
    a. Verify the signature on the TCG-CSR-IDEVID:
        i. Extract the IAK Public Key from the IAK Public Area in the TCG-CSR-IDEVID.
        ii. Use the IAK public key to verify the signature on the TCG-CSR-IDEVID.
    b. Extract the EK certificate from the TCG-CSR-IDEVID and verify the EK Certificate using the indicated TPM manufacturer's public key.
    c. Verify the attributes (TPMA_OBJECT bits) of the IAK Public Area to ensure that the key is a Restricted, fixedTPM, fixedParent signing key. Ensure all other attributes meet CA policy.
6. The CA issues a challenge to the device to validate that the TPM has the EK matching the certificate in step 2b and that the IAK whose public area is included in 2c is loaded in the device's TPM. The challenge data blob is created using the following procedure:
    a. Calculate the cryptographic Name of the IAK, by hashing its public area with its associated hash algorithm, prepended with the Algorithm ID of the hashing algorithm. Refer to TPM 2.0 Library Specification [2], Part 1, Section 16 ("Names").
    b. Using the sequence described in TPM 2.0 Library Specification, Part 3, section 12.6.3 ("TPM2_MakeCredential Detailed Actions"), create the encrypted "credential" structure to be sent to the device. When building this encrypted structure, objectName is the Name of the IAK calculated in step 'a' and Certificate (which is the payload field) holds a nonce (whose size matches the Name hash). Retain the nonce for use in later steps.
    c. The CA sends the encrypted "credential" blob to the enrolling TPM device.
7. The device software uses the TPM2_ActivateCredential command to release CertInfo data. During this operation, the TPM verifies the IAK's Name (and therefore the entire public area) using the EK. Because the EK is used, this operation requires Endorsement Hierarchy Admin Authorization. The returned CertInfo is

the Proof of Possession (POP) that is the nonce provided in step 6b above. Note that the single round-trip method in section 6.1.3 may be used to eliminate this step.

8. The device returns the recovered CertInfo (the nonce) to the CA. The CA matches the received value with the nonce retained in step 6b. The CA's receipt and validation of the CertInfo nonce is a Proof of Possession that demonstrates that the IAK is resident in the TPM described by the EK Certificate, is non-duplicable and fixedTPM.
9. The CA continues by issuing an IAK certificate and returning it to the device for persistent storage.

At this point the IAK can sign further keys using TPM2_Certify to provide proof to the CA as needed.

Start of informative comment

Note that the outsideInfo field provided to the TPM in the TPM2_CreatePrimary command needs to be known to the CA or evaluator of the creation ticket.

End of informative comment

6.1.3 Single Round-Trip Variation

It is possible to perform the operations of section 6.1.2 with a single round trip from the device to the CA by using the methods described in TPM 2.0 Library Specification [2] Part 1, "Credential Protection" and the TPM2_ActivateCredential command. Using this method, the CA issues the certificate using the CA's equivalent of the TPM2_MakeCredential command. This method uses a random symmetric key to encrypt the certificate (the CA's response to the CSR) and then encrypts the random symmetric key such that the TPM can decrypt it using the TPM2_ActivateCredential command. In the TPM2_MakeCredential command description, the key to be encrypted is the "credential" and is limited in size to the digest size associated with the nameAlg associated with the TPM EK. With this method, only the TPM with the EK in the TCG-CSR-IDEVID (section 13.1) is able to decrypt and use the created certificate. However, because the CA does not explicitly require proof-of-possession of the EK, this method does not explicitly provide assurance B of section 6.1.2.

6.1.4 Included Attestation Variation

It is RECOMMENDED that a CA performs device validation during the OEM enrollment process. This is very desirable so that a remote CA can be assured it is issuing certificates to a device running trusted software.

When creating a new IAK, the TPM puts the digest of selected creationPCRs in the IAK's creationData and the creationTicket is signed by the TPM once the key is created. This creationTicket is then given to the same TPM once the newly created IAK has been loaded, and then the new key is certified by the TPM. The TPM creates a signed attestation structure, including the digest of the selected PCRs. The CA can then verify the PCRs as of the time of creation of the IAK.

Start of informative comment

Depending on the size of data to be hashed, it may be necessary to use the command sequence TPM2_HashSequenceStart, TPM2_SequenceUpdate, TPM2_SequenceFinalize instead of using TPM2_Hash when creating the required digest.

End of informative comment

The procedure is:

1. Create a new IAK in the Endorsement Hierarchy using the TPM2_CreatePrimary command.
2. Certify the creation ticket with the new IAK using the TPM2_CertifyCreation command.
3. Create the TCG-CSR-IDEVID (see section 13.1) that the CA will use for enrollment.
    a. Include a boot event log

   b. Include the signed attestation structure (certifyInfo and its signature) and the creation data from step 2 of this procedure.
4. Create a digest of the TCG-CSR-IDEVID with the TPM2_Hash command and sign the TCG-CSR-IDEVID with the IAK using TPM2_Sign.
5. Send TCG-CSR-IDEVID to the CA
6. The CA performs checks to:
   a. Validate the TPM EK cert
   b. Validate IAK key attributes
   c. Verify that the boot event log, audit log, creation data and the signed attestation structure include expected attributes and meet requirements.
7. When the single round-trip variation of section 6.1.3 is not used, the CA sends a challenge using the TPM2_ActivateCredential method to confirm that the TPM contains the Endorsement private key. This provides Assurance B of section 6.1.2. Refer to step 7 of section 6.1.2.
8. Platform firmware starts a new audited session with the TPM and uses these steps:
   a. Issue the TPM2_ActivateCredential command using the challenge from step 7. This will release the CA's nonce in the context of the session.
   b. Read the appropriate set of PCRs (for verification that no additional firmware has been loaded).
   c. Close the audit session and send the session log and signature to the CA as the challenge response.
9. The CA receives the reply, verifies the signature, verifies the nonce and validates that PCRs have not changed from their expected state.
10. The CA then creates and issues certificates.
11. The device receives and stores certificates.

6.2 OEM Creation of IAK and IDevID in a Single Pass

An OEM can augment the process of section 6.1 to include an IDevID (signing) key and its certification information (TPM2_Certify of the IDevID key using the IAK) to produce both the IAK and IDevID signing certificates in the same transaction.

The variation of section 6.1.4, which describes how an initial attestation can be combined with enrollment, can also be combined with this enrollment protocol.

Details of issued certificates are described in Section 8, "Certificate Fields in Detail."

6.2.1 Requirements

Software under control of the OEM MUST be able to create keys as described in section 3.7, retrieve the TPM's EK certificate and build the TCG-CSR-IDEVID. The software must be able to securely create and sign a TCG-CSR-IDEVID (refer to section 9.1).

The OEM CA MUST validate TPM key certification data as signed by the IAK using methods described in the procedure below.

IEEE 802.1AR requires that use of IDevID MUST be enabled by default and it MUST be possible for the user to disable it. In the context of TPM keys, use of IDevID keys is enabled and disabled by controls on use of a Hierarchy (e.g. enabling or disabling the Endorsement Hierarchy) or by use of TPM User and Admin authorizations of DevID keys.

Note that the single round-trip method in section 6.1.3 may be used in combination with this single pass method, but doing so will result in the CA not receiving proof-of-possession of the EK and the nonce provided in the POP challenge will not be available.

The “Included Attestation Verification” of section 6.1.4 is RECOMMENDED for attestation of device state while performing the enrollment procedure. The device SHOULD be freshly booted before IAK and IDevID enrollment.

Details of issued certificates are described in Section 8, “Certificate Fields in Detail.”

6.2.2 Procedure

The Procedure (below) provides the following assurances:

A. A TPM is authentic (by validating the EK certificate).
B. This specific TPM is authentic and is represented by the EK certificate (i.e. the EK private key resident in this TPM corresponds to the public key in this EK certificate.)
C. This IAK is authentic (the IAK is resident in the same TPM as the EK private key corresponding the public key in this EK certificate.)
D. The Attestation Key (IAK) is in the TPM identified by the EK Certificate.
E. The IDevID key is authentic and is resident in the same TPM as the IAK.

OEMs may want to consider whether they have requirements for endpoint authentication for factory based enrollment processes. Such requirements are outside the scope of this specification.

Detailed Procedure:

1. Platform software creates a Restricted, fixedTPM, fixedParent Primary signing key (the IAK) in the Endorsement Hierarchy as described in section 3.7 and section 7.2.
2. Platform software creates an Unrestricted, fixedTPM, fixedParent Primary signing key (the IDevID key) in the Endorsement Hierarchy as specified in section 3.7.
3. Platform software uses TPM2_Certify to certify the IDevID key (the key created in step 2) with the IAK (created in step 1), producing a signed TPMB_Attest structure. This will show that the IDevID key is in the same TPM as the IAK and, by including the Name field, serves as a signature of the IDevID key public data which specifies the required key attributes verified in step 7.e.
4. Platform software builds the TCG-CSR-IDEVID structure described in section 13.1 including:
    a. Platform Identity Information—the non-TPM device information needed to build the 802.1AR certificate. This includes the device model and serial number.
    b. EK Certificate
    c. IAK Public Area (TPMT_PUBLIC structure)
    d. IDevID key Public Area (TPMT_PUBLIC structure)
    e. IDevID key TPMB_Attest structure with signature
5. Apply a signature to the TCG-CSR-IDEVID using the IDevID key. This signature serves as an integrity check of the device information when in transit from the device to the CA, preventing undetected modification by a MITM.
6. The device software sends the TCG-CSR-IDEVID to the CA.
7. CA verifies the received data:
    a. Extract IDevID public key and verify the signature on TCG-CSR-IDEVID
    b. Verify the EK Certificate using the indicated TPM manufacturer’s public key
    c. Verify TPM residency of IDevID key using the IAK public key to validate the signature of the TPMB_Attest structure.
    d. Verify the attributes of the IDevID key public area.
    e. Verify the attributes of the IAK public area.
    f. Calculate the Name of the IAK, by hashing its public area with its associated hash algorithm, prepended with the Algorithm ID of the hashing algorithm. Refer to TPM 2.0 Library Specification [2] Part 1, Section 16, “Names”.
    g. Using the sequence described in the TPM 2.0 Library Specification [2] Part 3, section 12.6.3 (TPM2_MakeCredential Detailed Actions), create the encrypted “credential” structure to be sent to

the device. When building this encrypted structure, objectName is the Name of the IAK calculated in step ‘a’ and Certificate (which is the payload field) holds a nonce (whose size matches the Name hash). Retain the nonce for use in later steps.

8. The CA sends the encrypted blob created in step 7g to the enrolling device.
9. The device software uses the TPM2_ActivateCredential command to release the CertInfo data. During this operation, the TPM verifies the IAK’s Name (and therefore the entire IAK public area) using the EK. Because the EK is used, this operation requires EH Admin Authorization. The returned CertInfo is the Proof of Possession (POP) that is the nonce provided in step 7.g above. Note that the single round-trip method in section 6.1.3 may be used to eliminate this step.
10. The device returns the recovered CertInfo (the nonce) to the CA. The CA matches the received value with the nonce retained in step 7.g. The CA’s receipt and validation of the CertInfo nonce is a Proof of Possession that demonstrates that the IAK is resident in the TPM described by the EK Certificate, is non-duplicable and fixedTPM.
11. The CA continues by issuing IAK and IIDevID certificates and returning them to the device for persistent storage.

6.3 Owner Creation of an LAK Certificate based on an IAK Certificate

6.3.1 Requirements

An existing IAK Certificate is required to perform this operation. Software under control of the administrator or owner must be able to retrieve the IAK Certificate as well as load and use the IAK key.

For maximum assurance when creating a TCG-CSR-LDEVID, the device should be freshly booted and boot attestation validated using the IAK before LAK enrollment.

Creation of the LAK key in the Storage Hierarchy is recommended. Including device information, including the serial number, from the IAK certificate Subject in the new LAK certificate is recommended.

The signature over the CSR must be created by the TPM using TPM2_Hash in order for the new LAK to sign the CSR.

The CA’s Certificate Practice Statement SHOULD state how the CA verifies TPM residency of the LAK.

Details of issued certificates are described in Section 8, “Certificate Fields in Detail.”

Start of informative comment

Depending on the size of data to be hashed, it may be necessary to use the command sequence TPM2_HashSequenceStart, TPM2_SequenceUpdate, TPM2_SequenceFinalize instead of using TPM2_Hash when creating the required digest.

End of informative comment

6.3.2 Procedure

The Procedure (below) provides the following assurance:

A. The new LAK is resident in the same TPM as the IAK.

Detailed Procedure:

1. Platform software creates a Restricted, fixedTPM, fixedParent signing key (the LAK) using a method selected from section 3.7 and section 7.2.
2. Platform software uses TPM2_Certify to certify the new LAK created in step 1 with the IAK, producing a signed TPM2B_Attest structure. This process proves that the new LAK is on the same TPM as the IAK.

3. Platform software builds the TCG-CSR-LDEVID structure described in section 13.2 including:
    a. The IAK certificate. This certificate identifies the device as well as providing device binding of the TPM to the stated device.
    b. The TPM2B_ATTEST structure returned by TPM2_Certify on the LAK.
    c. The TPMT_SIGNATURE structure returned by TPM2_Certify on the LAK.
4. Platform software creates a digest of the TCG-CSR-LDEVID using the TPM2_Hash command and then uses TPM2_Sign with the new LAK to create a signature for the TCG-CSR-LDEVID. This signature serves as an integrity check of the device information when in transit from the device to the CA, and proves control of the LAK to the CA.
5. The device software sends the signed TCG-CSR-LDEVID to the CA.
6. The CA verifies the received data:
    a. Extract the LAK public key from the TCG_CSR-LDEVID and verify the signature on the TCG-CSR-LDEVID.
    b. Verify the IAK certificate using the OEM's trust anchor certificate public key.
    c. Verify the signature on the TPM2B_ATTEST structure using the IAK certificate public key.
    d. Verify the attributes (TPMA_OBJECT bits) of the LAK Public Area to ensure that the key is a Restricted, fixedTPM, fixedParent signing key. Ensure all other attributes meet CA policy.
7. With all data verified, the CA continues by issuing the LAK certificate and returning it to the device for persistent storage.

6.4 Owner Creation of an LDevID Certificate based on an OEM installed IAK Certificate

6.4.1 Requirements

Software under control of the administrator or owner will need access to the IAK Certificate as well as use of the IAK key to verify residency of the new LDevID in the same TPM as the IAK, allowing the CA to assert that the new certificate identifies the same device as the IAK certificate.

The OEM needs to provide a software feature for the device administrator or owner to securely create, validate and sign a TCG-CSR-LDEVID (refer to section 13.2) in the Storage Hierarchy and the administrator or user must have control of the Storage Hierarchy during LDevID enrollment.

Creation of the LAK key in the Storage Hierarchy is recommended. The LDevID enrollment process should validate TPM key data to verify that the new LDevID key belongs to the device described by the IAK certificate. The Local CA SHOULD verify that the LDevID key's properties match the CA's certification practice policy by verifying TPM binding to the device and verifying key Attributes and Policies.

For maximum security assurance when creating the TCG-CSR-LDEVID, the device should be freshly booted and boot attestation validated before LDevID enrollment.

Details of issued certificates are described in Section 8, "Certificate Fields in Detail."

6.4.2 Procedure

Proving TPM residency of the LDevID key is performed in a similar manner to the procedure of section 6.3: the IAK credential is used to prove that the new LDevID signing key (instead of the LAK) belongs to the TPM using the IAK.

6.5 Owner Creation of an LDevID Certificate based on LAK

6.5.1 Requirements

Software under control of the administrator or owner must be able to retrieve the LAK Certificate as well as load and use the LAK key.

Creation of the LDevID key in the Storage Hierarchy is recommended.

The administrator, owner or user will need to securely create and sign a TCG-CSR-LDEVID (refer to section 13.2).

The Local CA should validate TPM key certification data as signed by the LAK using methods similar to section 6.3.2. The CA must therefore trust the security assertions made in the Certificate Practice Statement regarding creation of the LAK Certificate.

For maximum assurance when creating a TCG-CSR-LDEVID, the device should be freshly booted and boot attestation validated before LDevID enrollment.

Including device information, including the serial number, from the LAK certificate Subject in the new LDevID certificate is recommended. The Local CA should verify that the key properties match the CA's certificate practice policy by verifying key Attributes and Policies.

Details of issued certificates are described in Section 8, "Certificate Fields in Detail."

6.5.2 Procedure

Proving TPM residency of the LDevID key is performed in a similar manner to the procedure of section 6.4: the LAK and its certificate are used to prove that the new LDevID key belongs to the TPM, with the identity of the device conveyed using the LAK and certificate instead of the IAK and certificate.

6.6 Owner Creation of an LAK Certificate based on a Platform Certificate

6.6.1 Requirements

The TCG Platform Attribute Credential Profile [10] describes Platform Certificate requirements, including details of data fields included within the platform certificate. The platform certificate binds the TPM (as described by its EK certificate) to a platform by means of an OEM signature on the certificate.

The CA issuing an LAK certificate to the device should verify the TPM EK Certificate Serial Number against the TPM EK Serial Number included in the Platform Certificate.

The CA should perform a proof of possession of the EK, as described in the procedure of section 6.6.2, to prove that the CA is issuing the certificate to the TPM (and hence the same device) described in the platform certificate. Proving that the new LAK is in the specified TPM requires using the TPM's Endorsement Key and thus requires use of Endorsement Hierarchy authorization.

Creation of the LAK key in the Storage Hierarchy is recommended. Including the device information, such as the serial number, from the platform certificate in the LAK certificate Subject is recommended. The Local CA should verify that the key properties match the CA's certificate practice policy by verifying TPM binding to the device and verifying key Attributes and Policies as described in the detailed of section 6.6.2, below.

Details of issued certificates are described in Section 8, "Certificate Fields in Detail."

6.6.2 Procedure

This procedure is based on the IAK enrollment procedure of section 6.1.2, with the device data coming from the platform certificate.

This sequence provides the following assurances:

A. A TPM is authentic (by validating the EK certificate).
B. This specific TPM is authentic and is represented by the EK certificate (i.e. the EK private key resident in this TPM corresponds to the public key in this EK certificate.)
C. This LAK is authentic (the LAK is resident in the same TPM as the EK private key corresponding to the public key in this EK certificate.)

D. This LAK is bound to the device (the device OEM affirms in a Platform Certificate that the TPM bound to the described device has the EK PubKey described in the EK Certificate).

The detailed Two Round-Trip Procedure is:

1. Platform software uses the TPM to create a Restricted, fixedParent, fixedTPM Ordinary signing key (the LAK) using TPM2_Create.
2. Platform software builds the TCG-CSR-LDEVID structure described in section 13.2 including the:
    a. Platform certificate.
    b. TPM EK Certificate.
    c. LAK Public Area (TPMT_PUBLIC structure of the LAK).
3. Platform software applies a signature to the TCG-CSR-LDEVID by first using the TPM to create a digest and then using the LAK to sign the digest. This signature serves as an integrity check of the device information when in transit from the device to the CA, preventing undetected modification by a MITM. Note that using the TPM to create the digest is unavoidable in order to create a signature with the LAK.
4. The device software sends the TCG-CSR-LDEVID to the CA.
5. The CA verifies the received data:
    a. Verify the signature on the TCG-CSR-LDEVID:
        i. Extract the LAK public key from the CSR.
        ii. Use the LAK public key to verify the signature on the CSR.
    b. Extract the EK Certificate from the CSR and then verify the EK Certificate using the indicated TPM manufacturer's public key.
    c. Verify the attributes (TPMA_OBJECT bits) of the LAK public area to ensure that the key is a Restricted, fixedTPM, fixedParent signing key. Ensure that all other attributes meet the CA's policy.
6. The CA issues a challenge to the device to validate that the TPM has the EK matching the certificate in step 2b and that the LAK whose public area is included in 2c is loaded in the device's TPM. The challenge data blob is created using the following procedure:
    a. Calculate the cryptographic Name of the LAK, by hashing its public area with its associated hash algorithm, prepended with the Algorithm ID of the hashing algorithm. Refer to TPM 2.0 Library Specification [2] Part 1, Section 16, "Names".
    b. Using the sequence described in the TPM 2.0 Library Specification [2] Part 3, section 12.6.3 (TPM2_MakeCredential Detailed Actions), create the encrypted "credential" structure to be sent to the device. When building this encrypted structure, objectName is the Name of the LAK calculated in step 'a' and Certificate (which is the payload field) holds a nonce (whose size matches the Name hash). Retain the nonce for use in later steps.
    c. The CA sends the encrypted "credential" blob to the enrolling device.
7. The device software uses the TPM2_ActivateCredential command to release the CertInfo data which is the nonce provided in step 6b above. During this operation, the TPM verifies the LAK's Name (and therefore the entire LAK public area) using the EK. Because the EK is used, this operation requires Endorsement Hierarchy Admin Authorization. The returned CertInfo is the Proof of Possession (POP) that is the nonce provided in step 7.g above. Note that the single round-trip method in section 6.1.3 may be used to eliminate this step.
8. The device returns the recovered CertInfo (the nonce) to the CA. The CA matches the received value with the nonce retained in step 6b. The CA's receipt and validation of the CertInfo nonce is a Proof of Possession that demonstrates that the LAK is resident in the TPM described by the EK Certificate, is non-duplicable and fixedTPM.
9. The CA continues by issuing an LAK certificate, using device information from the platform certificate. The LAK certificate is returned to the device for persistent storage.

7 TPM-based Device Identity Lifecycle Requirements

7.1 Provisioning

Placing a TPM in a product does not imply that the product is provisioned with IDevID and IAK certificates. The need for IDevID and IAK provisioning is both product and use case dependent.

TPM based IDevID and IAK certificates are expected to have a long life. The initial provisioning and registration processes are important to the integrity and security of the device identity. For this reason, the IAK and IDevID certificates SHOULD be created in conjunction with device creation.

7.2 IDevID/IAK Hierarchy Selection

This section provides a description of the recommended method to provide and protect IDevID and IAK keys in a device with a TPM, including the hierarchy, authorization and policies to be used during OEM creation of IDevID/IAK keys.

By placing IDevID/IAK keys in the Endorsement Hierarchy, as recommended by this specification, the key authorization, usage policy and administration policy are placed under the control of the Privacy Administrator.

The recommended method enables a device's Privacy Administrator:

- To optionally recreate an IDevID/IAK key described by an IDevID/IAK certificate supplied by the device's manufacturer. The IDevID/IAK key can be recreated at any time during the lifetime of the TPM. In other words, the Privacy Administrator determines whether an IDevID/IAK key is in existence.
- To choose a password that authorizes administration and use of the IDevID/IAK key.
- To optionally choose a policy that delegates administration and use of the IDevID/IAK key to any number of entities.

The recommended method includes:

- The manufacturer creating an IDevID/IAK key and IDevID/IAK certificate for the device and supplying the IDevID/IAK certificate either with or to the device. It is unnecessary for the manufacturer to supply the IDevID/IAK key either on or to the device.
- The device's Privacy Administrator setting a password to authorize control and use of the IDevID/IAK key.

7.2.1 Recoverable IDevID/IAK Key

Key authorizations are described in detail in TPM 2.0 Library Specification [2], part 1. Figure 8 (IDevID/IAK Key Authorization Relationships) illustrates the key authorization controls used in this section.

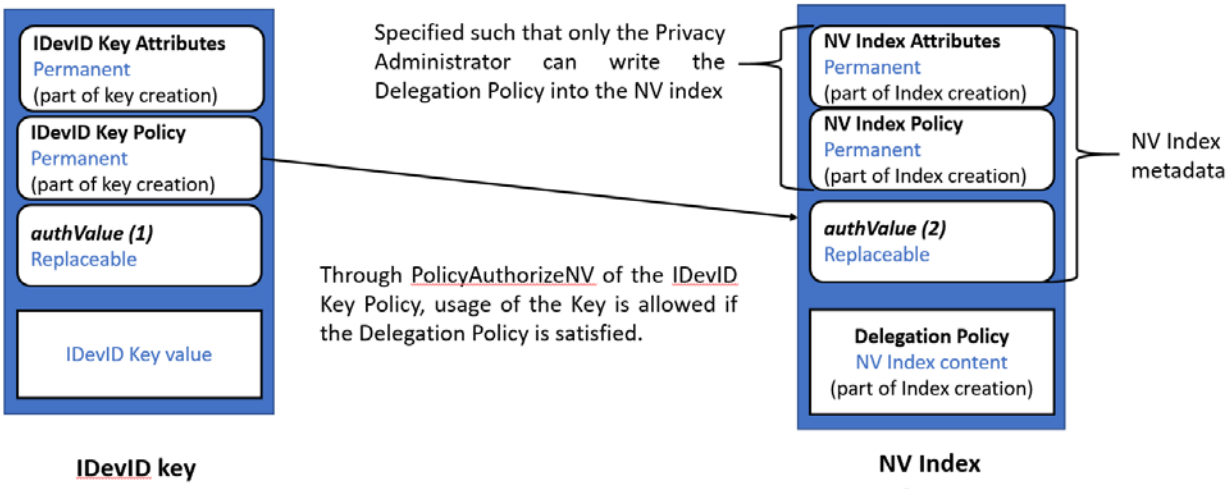

Figure 8 - IDevID/IAK Key Authorization Relationships

The recommended method is to create primary keys in the TPM's Endorsement Hierarchy using the command TPM2_CreatePrimary and to use those primary keys as the IDevID/IAK keys. This method is recommended for all general-purpose computing devices where product features require an IDevID/IAK to be installed.

Using this method, IDevID/IAK keys can be recreated at any time during the lifetime of the TPM (strictly, during the lifetime of the TPM's Endorsement Primary Seed). Thus the IDevID/IAK keys can be recreated after a device has been reinitialized with a new Operating System, for example. Because the IDevID/IAK keys can be recreated on demand, the IDevID/IAK keys do not need to be supplied with (i.e. stored on) the device.

Since an IDevID/IAK key is a primary key in the Endorsement Hierarchy, an IDevID/IAK key can be recreated only by the current Privacy Administrator. When creating IDevID/IAK keys, an OEM SHOULD use the templates provided in section 7.3 of this specification, as these templates allow the device firmware or operating system to act on behalf of the device owner to administer and use IDevID/IAK keys with standard methods.

The recommended key templates in section 7.3.4 enable the IDevID/IAK key's authValue password to authorize use of the IDevID/IAK key but not administration of the IDevID/IAK key. The mechanism is that userWithAuth == SET enables the USER role via authValue, but adminWithPolicy == SET disables the ADMIN role with authValue. Therefore, someone who knows the IDevID/IAK key's authValue password can use the IDevID/IAK key, but can't administer the IDevID/IAK key. Once an IDevID/IAK has been created, an administrator can make copies of the IDevID/IAK key with a different authValue by recreating the key with the new authValue. Refer to section 7.2.4.

The recommended key templates are augmented by combination with the recommended NVindex policy in section 7.3.5. The recommended key templates provide the recommended IDevID/IAK Key Policy. The recommended NVindex Policy enables the EH password (endorsementAuth) and the Delegation Policy to authorize both usage and administration of the IDevID/IAK Key.

Refer to Figure 8 - IDevID/IAK Key Authorization Relationships": The policy mechanism is configured with userWithAuth = SET to enable the USER role via policy, with adminWithPolicy = SET to enable the ADMIN role with policy. The IDevID/IAK Key Policy includes a TPM2_PolicySecret() component referencing endorsementAuth and a TPM2_PolicyAuthorizeNV() component referencing an NV index controlled by the recommended NVindex Policy and containing the Delegation Policy.

The IDevID/IAK Key Policy has a TPM2_PolicyAuthorizeNV component that includes the Name of the NV index, which includes that NV index's NVindex Policy, which restricts who can create and write the NV index. Therefore the Privacy Administrator (who has endorsementAuth) can both use and administrate the IDevID/IAK key, and anyone allowed by the Delegation Policy can use and/or administer the IDevID/IAK key as allowed by the policy.

7.2.2 Implementation options

The most complex implementation optionally uses endorsementAuth to authorize use and administration of an IDevID/IAK key, optionally uses a chosen authValue to authorize use of the IDevID/IAK , and uses a chosen Delegation Policy to authorize use and administration of an IDevID/IAK key. In this most complex implementation, the authValue field in the template used to create the IDevID/IAK key is either set to a very long random value and immediately forgotten, or set to a chosen password, which is ideally a random value obtained via TPM2_GetRandom. An NV index must be created and the chosen Delegation Policy written to the NV index. The Delegation Policy should delegate administration of the IDevID/IAK (so the IDevID/IAK can be administered by someone other than the Privacy Administrator) and may delegate different aspects of administration to different entities. The Delegation Policy may delegate use of the IDevID/IAK to different selected entities identified via different authentication methods, or under different constraints, such as time and device state. This method is RECOMMENDED for general purpose or multi-user devices and explained in more detail in section 7.3.

A less complex implementation uses endorsementAuth to authorize use and administration of an IDevID/IAK key, and a chosen authValue to authorize use of the IDevID/IAK . The main drawback of this implementation is that the Privacy Administrator must be available to administer the IDevID/IAK key (because endorsementAuth, which authorizes the Endorsement Hierarchy, should not be shared). In this less complex implementation, the authValue field in the template used to create the IDevID/IAK key is set to a chosen password, which is ideally a random value obtained via TPM2_GetRandom. There is no need to create an NV index. This method is NOT RECOMMENDED for general purpose or multi-user devices.

The simplest implementation uses endorsementAuth to authorize use and administration of an IDevID/IAK key. The main drawback of this implementation is that anyone who can use the IDevID/IAK key can also administer the Endorsement Hierarchy, because endorsementAuth authorizes the Endorsement Hierarchy. In this simplest implementation, the authValue field in the template used to create the IDevID/IAK key is set to a very long random value and immediately forgotten. The random number is ideally obtained via TPM2_GetRandom. There is no need to create an NV index. This method is NOT RECOMMENDED for general purpose or multi-user devices but may be acceptable for embedded systems.

By design, the Privacy Administrator (who controls the EH password) is always in control of IDevID/IAK usage and administration. It is essential for the recommended method that the NVindex Policy gives the Privacy Administrator exclusive authority to replace the NV index itself and gives the Privacy Administrator exclusive write access to the Delegation Policy stored at the index.

7.2.3 Creating Recoverable IDevID/IAK keys

The Manufacturer acts as the current Privacy Administrator.

The Privacy Administrator uses TPM2_CreatePrimary to create a primary key in the TPM’s Endorsement Hierarchy using the template described in 7.3.4.

At the time of IDevID Key and Attestation Key creation, the Privacy Administrator SHOULD use a well-known key authValue. For simplicity, the Privacy Administrator SHOULD use the Empty Auth value as described in the TPM Specification.

To facilitate Zero Touch Provisioning, the Manufacturer creates an IDevID/IAK certificate for the IDevID/IAK keys and SHOULD store the certificates on the device. In lieu of storing the IDevID/IAK certificates on the device, the manufacturer MAY make them available externally, e.g. on a public web site.

7.2.4 Initializing Usage of a Recoverable IDevID/IAK key

The Owner (or operating system, acting on behalf of the Owner) acts as the current Privacy Administrator.

If it is necessary to remove all current personalization from the TPM, the Privacy Administrator issues TPM2_Clear, which clears endorsementAuth and changes ehProof, thereby cryptographically erasing all extant keys.

The Privacy Administrator uses TPM2_CreatePrimary to recreate a primary key in the TPM's Endorsement Hierarchy using the template described in section 7.3.4. All inPublic parameters in TPM2_CreatePrimary must be exactly the same as those used by the Platform Manufacturer. The authValue for use of the IDevID/IAK key is provided through the inSensitive parameter.

When TPM2_CreatePrimary is used to create a new version of the IDevID/IAK key with a new authValue, the old version is over-written, as the new key has the same Name as the old key (and is therefore the same object, notwithstanding the different authValue.)

The device operating system may either recreate the IDevID/IAK keys on demand, e.g. at each boot, or may create them with the desired authValue and then persist them in the TPM using the TPM2_EvictControl command. Persisting the IDevID/IAK keys is the only way to maintain a specific authValue and therefore SHOULD be performed when a Delegated Policy for the IDevID/IAK keys is not used.

When the Privacy Administrator uses TPM2_CreatePrimary to recreate the IDevID/IAK key, the Privacy Administrator inherently must set the IDevID/IAK 's authValue. If the OS controls access to the TPM and the OS will control use of the IDevID/IAK, the OS can just use the default TPM Empty Auth value to control the IDevID/IAK . If the OS can't control access to the TPM or the Privacy Administrator doesn't want IDevID/IAK to be authorized via passwords, the authValue in the IDevID/IAK key must be a large random value that is immediately forgotten outside the TPM.

The Privacy Administrator may optionally create an NV index before or after creating the IDevID/IAK key at the index assigned (by TCG) for storing an IDevID/IAK Delegation Policy. The Name of the NV index needs to exactly match the Name of the NV index used by the Manufacturer to compute the policyDigest created by TPM2_PolicyAuthorizeNV included in the IDevID/IAK Key Policy.

The Privacy Administrator chooses the desired IDevID/IAK Delegation Policy and writes that Delegation Policy into the NV index.

7.3 Delegated Usage of a Recoverable IDevID/IAK Key

TPM2_AuthorizeNV command required for delegation using NV is available only in TPMs implementing version 1.38 or later of the TPM 2.0 specification.

When the TPM supports the TPM2_PolicyAuthorizeNV command, and assuming that the Privacy Administrator has recreated the IDevID/IAK key and created an NV index containing an IDevID/IAK Delegation Policy, any entity authorized by that Delegation Policy can use the recreated IDevID/IAK key.

The delegated entity executes a policy authorization session that enumerates all the policy assertions required to satisfy the Delegation Policy. This policy session constructs a policyContext and policyDigest that was approved by the Privacy Administrator. Then the delegated entity executes TPM2_PolicyAuthorizeNV, pointing to the NV index. Provided the Delegation Policy stored in the NV Index matches the TPM's current policyDigest, the TPM replaces the current policyDigest with a replacement-policyDigest (which inherently indicates the Name of the NV index used by PolicyAuthorizeNV). Provided this replacement-policyDigest matches the IDevID/IAK Key Policy in the IDevID/IAK key, the entity can use the IDevID/IAK key.

7.3.1 Recoverable IDevID/IAK Key and Policy Details

At the time of key creation, the *unique* field MUST be set to one of the strings shown in Table 2, indicating the key's purpose.

Key Purpose	Required Unique String	Optional Unique String
IDevID	'IDEVID' (49 44 45 56 49 44_{16})	'IDEVID'|<device_str>
IAK	'IAK' (49 41 $4B_{16}$)	'IAK'|<device_str>

Table 2 - IDevID/IAK Unique Values

Table 2 illustrates that the string MAY also include an optional added device string. When used, the device_str is a sequence of printable ASCII octets, unique per device, that are concatenated to the required unique string. The combined string size MUST NOT be larger than the allowed *unique* size for the crypto suite selected.

If the device string is included, that string MUST be stored on the device so that it is available each time Primary keys are recreated. Storage of the device unique data is outside the scope of this specification as the method used is at the option of the OEM. The TPM NV Index 0x01C90020 has been reserved for storing this optional value in TPM NV when that method is chosen.

Neither version of the unique string is NUL terminated (i.e., the NUL termination is not supplied to the TPM.)

The TPM 2.0 Library specification [2] defines authentication mechanisms for the user of a TPM object, including an enhanced authorization mechanism based on policies. Refer to Part 1 (Authorization Roles) for information on Roles and to Part 3 (Table Decorations, Auth Role) for information on how commands (e.g. TPM2_PolicyCommandCode, TPM_CC_Certify, TPM_CC_ActivateCredential) are authorized based on Role.

As specified in this document, recoverable IDevID/IAK keys are created using enhanced authorization by including policy-based authorization in the templates.

IDevID/IAK certificates associated with the Recoverable IDevID/IAK Key Templates defined in this section are stored in the NV Indices specified in section 7.3.2. These templates specify RSA 2048 bit, ECC NIST P256, NIST P384, NIST P521, and SM2 P256 bit Restricted signing or Attestation keys whose administration and use authorizations are allowed with either the key's authValue or the key's authPolicy. The authPolicy in Templates H-1 to H-5 is set to IDevIDKeyPolicy.

Policies are expressed as a single digest, but that digest may be formed from an arbitrarily complex sequence of commands. Note that there is a TPM 2.0 command for PolicyOR, while the AND operation is implicit, due to use of the "extend" method used in creating a policyDigest.

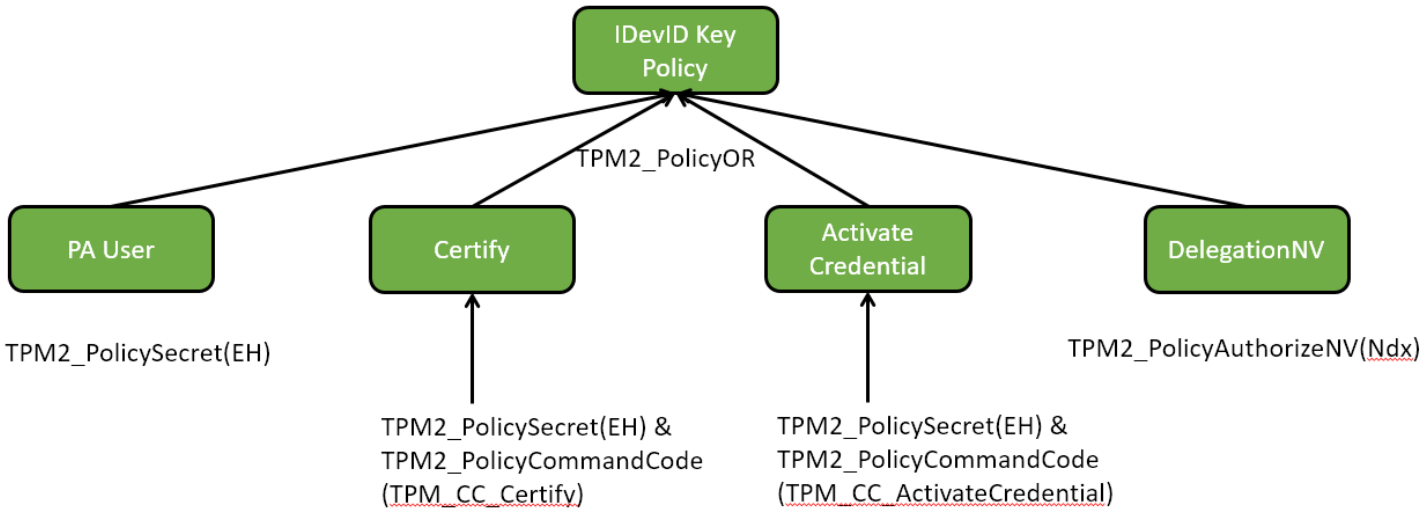

Figure 9 - Policy OR with Delegation Policy

Referring to Figure 9, IDevID Key Policy is an example policy OR with four constituents (i.e. showing up to four ways to *satisfy* the policy):

- The PA User policy digest allows the privacy administrator to use IDevID/IAK keys (i.e. to access the key's User role). The User digest is created with TPM2_PolicySecret(), where the authorization value is endorsementAuth.
- The Certify policy allows the Privacy Administrator to certify a key, which requires using the TPM2_PolicyCommandCode command with TPM_CC_Certify parameter when a Policy Session is used.
- The ActivateCredential policy allows the Privacy Administrator to use the key to release an associated credential via TPM2_ActivateCredential, which requires using the TPM2_PolicyCommandCode command with TPM_CC_ActivateCredential parameter.
- The PA User, Certify, and ActivateCredential branches exist to provide Privacy Administrator access to IDevID/IAK keys, but the DelegationNV branch exists for the Privacy Administrator to provide User access to IDevID/IAK keys. Once configured, the DelegationNV branch allows the User to use the IDevID/IAK keys, for example, in a signing operation.

  The DelegationNV policy is a digest created by executing TPM2_PolicyAuthorizeNV. This digest will provide authorization if the Delegation Policy in the authorizing NV Index has been satisfied. The authorizing NV index contains a Policy that enforces the Privacy Administrator's exclusive write-access to the NV Index while assigning the Administrator's choice of IDevID/IAK privileges to Users. This is explained at the end of section 7.3 "Delegated Usage of a Recoverable IDevID/IAK Key".

A policyDigest of all zeroes is the starting point for the hash algorithm used in the figure above. For simplicity this is not shown in the diagram, though it must be explicitly included in the policies used when issuing the required TPM 2.0 commands. (For more information, refer to Trusted Platform Module 2.0 Library Specification [2], Part 1, sections "Policy AND" and "Policy OR".)

The Privacy Administrator can create an IDevID/IAK Delegation Policy using the recommended policy in the Templates (section 7.3.5), which allow use (but not administration) of the IDevID/IAK keys. To activate a Delegation Policy, an NV index is created (at a reserved index location) using the recommended NVindex Policy and written with the Delegation Policy by the Privacy Administrator. The Delegation Policy can be replaced by the Privacy Administrator at any time. The Delegation Policy is stored in a reserved NV index as defined in section 7.3.2.

Creation of the NV Index is performed with Storage Hierarchy (owner) authorization, but the NV Index can be written only with Privacy Administrator authorization (TPM2_PolicySecret for the Endorsement Hierarchy.) Privacy Administrator authorization is required because only the Privacy Administrator has sufficient privilege to write the Delegation Policy that determines how the User can use and administer an IDevID/IAK key.

Device firmware may act as an agent for the Privacy Administrator when the device is first installed to support zero-touch provisioning.

7.3.2 IDevID/IAK Policy NV Indices for Recoverable Keys

The IDevID Policy NV Indices for IDevID Signing keys are defined as follows:

0x01C90010 : Policy Index DS1, nameAlg = SHA256
0x01C90011 : Policy Index DS2, nameAlg = SHA384
0x01C90012 : Policy Index DS3, nameAlg = SHA512
0x01C90013 : Policy Index DS4, nameAlg = SM3_256

The IAK Policy NV Indices for IAK keys are defined as follows:

0x01C90018 : Policy Index DA1, nameAlg = SHA256
0x01C90019 : Policy Index DA2, nameAlg = SHA384
0x01C9001A : Policy Index DA3, nameAlg = SHA512
0x01C9001B : Policy Index DA4, nameAlg = SM3_256

7.3.3 IDevID/IAK Unique String

As described in section 7.3.1, if IDevID/IAK Unique includes a random element, and if that random string is stored in TPM NV, then the random string MUST be stored using NV Index 0x01C90020.

7.3.4 Key Creation Templates for Recoverable Keys

7.3.4.1 Template H-0: Template Common Values

Parameter	Type	Content
objectAttributes	TPMA_OBJECT	fixedTPM = 1 stClear = 0 fixedParent = 1 sensitiveDataOrigin = 1 userWithAuth = 1 adminWithPolicy = 1 noDA = 0 encryptedDuplication = 0 **DevID (signing): Restricted = 0, AK: Restricted = 1** decrypt = 0 sign = 1

Table 3 - Values Common to Templates H-1 Through H-5

7.3.4.2 Template H-1: RSA 2048

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_RSA
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA256
objectAttributes	TPMA_OBJECT	Refer to Table H-0
authPolicy	TPMB_DIGEST	
size	UINT16	32
buffer	BYTE	PolicyDelegationNV
parameters	TPMS_RSA_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_NULL
symmetric->keyBits	TPMI_AES_KEY_BITS	NULL
symmetric->mode	TPMI_SYM_MODE	NULL
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ASYM_SCHEME	TPM_ALG_NULL (Signing) TPM_ALG_RSASSA (AK – PKCSV1_5) TPM_ALG_RSAPSS (AK)
scheme->details		NULL (Signing) TPM_ALG_SHA256 (AK)
keyBits	TPMI_RSA_KEY_BITS	2048
exponent	UINT32	0
unique	TPMB_PUBLIC_KEY_RSA	Refer to section 7.3.1
size	UINT16	Refer to section 7.3.1
buffer	BYTE	Refer to section 7.3.1

Table 4 - Default IDevID/IAK Template (TPMT_PUBLIC) H-1: RSA 2048

7.3.4.3 Template H-2: ECC NIST P256

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_ECC
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA256
objectAttributes	TPMA_OBJECT	Refer to Table H-0
authPolicy	TPMB_DIGEST	
size	UINT16	32
buffer	BYTE	PolicyIDevIDKey
parameters	TPMS_ECC_PARMS	

symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_NULL
symmetric->keyBits	TPMI_AES_KEY_BITS	NULL
symmetric->mode	TPMI_SYM_MODE	NULL
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ECC_SCHEME	TPM_ALG_ECDSA
scheme->details		NULL
curveID	TPMI_ECC_CURVE	TPM_ECC_NIST_P256
kdf->scheme	TPMI_ALG_KDF	TPM_ALG_NULL
kdf->details		NULL
unique	TPMS_ECC_POINT	Refer to section 7.3.1
x->size	UINT16	Refer to section 7.3.1
x->buffer	BYTE	Refer to section 7.3.1
y->size	UINT16	Refer to section 7.3.1
y->buffer	BYTE	Refer to section 7.3.1

Table 5 - Default IDevID/IAK Template (TPMT_PUBLIC) H-2: ECC NIST P256

7.3.4.4 Template H-3: ECC NIST P384

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_ECC
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA384
objectAttributes	TPMA_OBJECT	Refer to Table H-0
authPolicy	TPMB_DIGEST	
size	UINT16	48
buffer	BYTE	PolicyIDevIDKey
parameters	TPMS_ECC_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_NULL
symmetric->keyBits	TPMI_AES_KEY_BITS	NULL
symmetric->mode	TPMI_SYM_MODE	NULL
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ECC_SCHEME	TPM_ALG_ECDSA
scheme->details		NULL
curveID	TPMI_ECC_CURVE	TPM_ECC_NIST_P384
kdf->scheme	TPMI_ALG_KDF	TPM_ALG_NULL
kdf->details		NULL
unique	TPMS_ECC_POINT	Refer to section 7.3.1
x->size	UINT16	Refer to section 7.3.1
x->buffer	BYTE	Refer to section 7.3.1
y->size	UINT16	Refer to section 7.3.1
y->buffer	BYTE	Refer to section 7.3.1

Table 6 - Default IDevID/IAK Template (TPMT_PUBLIC) H-3: ECC NIST P384

7.3.4.5 Template H-4: ECC NIST P521

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_ECC
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA512
objectAttributes	TPMA_OBJECT	Refer to Table H-0
authPolicy	TPMB_DIGEST	
size	UINT16	64
buffer	BYTE	PolicyIDevIDKey
parameters	TPMS_ECC_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_NULL
symmetric->keyBits	TPMI_AES_KEY_BITS	NULL

symmetric->mode	TPMI_SYM_MODE	NULL
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ECC_SCHEME	TPM_ALG_ECDSA
scheme->details		NULL
curveID	TPMI_ECC_CURVE	TPM_ECC_NIST_P521
kdf->scheme	TPMI_ALG_KDF	TPM_ALG_NULL
kdf->details		NULL
unique	TPMS_ECC_POINT	Refer to section 7.3.1
x->size	UINT16	Refer to section 7.3.1
x->buffer	BYTE	Refer to section 7.3.1
y->size	UINT16	Refer to section 7.3.1
y->buffer	BYTE	Refer to section 7.3.1

Table 7 - Default IDevID/IAK Template (TPMT_PUBLIC) H-4: ECC NIST P521

7.3.4.6 Template H-5: ECC SM2 P256

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_ECC
nameAlg	TPMI_ALG_HASH	TPM_ALG_SM3_256
objectAttributes	TPMA_OBJECT	Refer to Table H-0
authPolicy	TPMB_DIGEST	
size	UINT16	32
buffer	BYTE	PolicyIDevIDKey
parameters	TPMS_ECC_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_NULL
symmetric->keyBits	TPMI_SM4_KEY_BITS	NULL
symmetric->mode	TPMI_SYM_MODE	NULL
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ECC_SCHEME	TPM_ALG_NULL
scheme->details		NULL
curveID	TPMI_ECC_CURVE	TPM_ECC_SM2_P256
kdf->scheme	TPMI_ALG_KDF	TPM_ALG_NULL
kdf->details		NULL
unique	TPMS_ECC_POINT	Refer to section 7.3.1
x->size	UINT16	Refer to section 7.3.1
x->buffer	BYTE	Refer to section 7.3.1
y->size	UINT16	Refer to section 7.3.1
y->buffer	BYTE	Refer to section 7.3.1

Table 8 - Default IDevID/IAK Template (TPMT_PUBLIC) H-5: SM2 P256

7.3.5 Policy NV Templates for DevID Key Delegation

The Delegation Policy written in the NV Index (examples are provided in section 7.3.7) is not the authPolicy of the NV Index itself. All PolicyNV Indices contain a public structure (TPMS_NV_PUBLIC) that includes the Index's own authPolicy that controls access to the index's payload area. Once the payload is populated with a digest representing a delegation policy, this digest may then be used as an IDevID/IAK delegation policy by keys created with one of the Recoverable IDevID Key templates.

One NV Index is reserved (section 7.3.5) for each hash algorithm: SHA256, SHA384, SHA512 and SM3_256.

The auth policies included in the NV templates specify the Privacy Administrator as the authorizing entity for writing the NV Index.

The Policy NV Indices (i.e. the payload data) SHOULD NOT be populated by the OEM. Instead, a Policy NV Index MAY be defined by the device Owner using Owner Authorization.

NOTE: It is not anticipated that all possible Policy NV Indices would be populated at the same time in a TPM. Instead only the Policy Index corresponding to the IDevID & IAK keys in operation would be populated.

7.3.5.1 NV Policy Template I-0: Values common to I-1 through I-5

Parameter	Type	Content
attributes	TPMA_NV	TPM2_NT = 0 TPMA_NV_PPWRITE = 0 TPMA_NV_OWNERWRITE = 0 TPMA_NV_AUTHWRITE = 0 TPMA_NV_POLICYWRITE = 1 TPMA_NV_POLICY_DELETE = 0 TPMA_NV_WRITELOCKED = 0 TPMA_NV_WRITEALL = 1 TPMA_NV_WRITEDEFINE = 0 TPMA_NV_WRITE_STCLEAR = 0 TPMA_NV_GLOBALLOCK = 0 TPMA_NV_PPREAD = 1 TPMA_NV_OWNERREAD = 1 TPMA_NV_AUTHREAD = 1 TPMA_NV_POLICYREAD = 1 TPMA_NV_NO_DA = 1 TPMA_NV_ORDERLY = 0 TPMA_NV_CLEAR_STCLEAR = 0 TPMA_NV_READLOCKED = 0 TPMA_NV_WRITTEN = 1 TPMA_NV_PLATFORMCREATE = 0 TPMA_NV_READ_STCLEAR = 0 (0x220F1008)

Table 9 - Common NV Policy Template Values

7.3.5.2 NV Policy Template I-1: SHA256

Parameter	Content	Content
nvIndex	TPMI_RH_NV_INDEX	0x01C90010 (Signing Key), 0x01C90018 (AK)
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA256 (0x000B)
attributes	TPMA_NV	Refer to Policy Template I-0
authPolicy	TPMB_DIGEST	
size	UINT16	32 (0x0020)
buffer	BYTE	PA User Policy. Refer to section 7.3.6.1.
dataSize	UINT16	32 (0x0020)

Table 10 - NV Policy Index (TPMS_NV_PUBLIC) I-1: SHA256

NOTE: The first two bytes of the Index data area contain a TPM_ALG_ID structure, followed by the policy value (without size).

7.3.5.3 NV Policy Template I-2: SHA384

Parameter	Type	Content
nvIndex	TPMI_RH_NV_INDEX	0x01C90011 (Signing Key), 0x01C90019 (AK)
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA384 (0x000C)
attributes	TPMA_NV	Refer to Policy Template I-0
authPolicy	TPMB_DIGEST	
size	UINT16	48 (0x0030)
buffer	BYTE	PA User Policy. Refer to section 7.3.6.1.
dataSize	UINT16	50 (0x0032)

Table 11 - NV Policy Index (TPMS_NV_PUBLIC) I-2: SHA384

NOTE: The first two bytes of the Index data area contain a TPM_ALG_ID structure, followed by the policy value (without size).

7.3.5.4 NV Policy Template I-3: SHA512

Parameter	Type	Content
nvIndex	TPMI_RH_NV_INDEX	0x01C90012 (Signing Key), 0x01C9001A (AK)
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA512 (0x000D)
attributes	TPMA_NV	Refer to Policy Template I-0
authPolicy	TPMB_DIGEST	
size	UINT16	64 (0x0040)
buffer	BYTE	PA User Policy. Refer to section 7.3.6.1.
dataSize	UINT16	66 (0x0042)

Table 12 - NV Policy Index (TPMS_NV_PUBLIC) I-3: SHA512

NOTE: The first two bytes of the Index data area contain a TPM_ALG_ID structure, followed by the policy value (without size).

7.3.5.5 NV Policy Template I-4: SM3_256

Parameter	Type	Content
nvIndex	TPMI_RH_NV_INDEX	0x01C90013 (Signing Key), 0x01C9001B (AK)
nameAlg	TPMI_ALG_HASH	TPM_ALG_SM3_256 (0x0012)
attributes	TPMA_NV	Refer to Policy Template I-0
authPolicy	TPMB_DIGEST	
size	UINT16	32 (0x0020)
buffer	BYTE	PA User Policy. Refer to section 7.3.6.1.
dataSize	UINT16	34 (0x0022)

Table 13 - NV Policy Index (TPMS_NV_PUBLIC) I-4: SM3_256

NOTE: The first two bytes of the Index data area contain a TPM_ALG_ID structure, followed by the policy value (without size).

7.3.6 Policy Computation

This section describes how IDevID/IAK Key policies and NVindex policies used in the IDevID/IAK key templates and the Policy NV Indices are computed. The equations in this section are copied from the TPM 2.0 Library Specification [2] Part 3. If there are any inconsistencies between the equations below and the equations defined in the TPM 2.0 Library Specification, the definitions in the Library Specification take precedence.

7.3.6.1 Computing PA User Policy

The PA User Policy is comprised of a single policy statement: TPM2_PolicySecret using the EH as the authorizing entity.

TPM2_PolicySecret() uses the PolicyUpdate function:

PolicyUpdate(TPM_CC_PolicySecret, authObject→Name, policyRef)

This is equivalent to:

policyDigestnew ≔ HpolicyAlg(policyDigestold || TPM_CC_PolicySecret || authObject→Name)

policyDigestnew+1 ≔ HpolicyAlg(policyDigestnew || policyRef.buffer)

With:

policyAlg = SHA256, or SHA384, or SHA512, or SM3_256

policyDigestold = Zero Digest (32, or 48, or 64 bytes)

TPM_CC_PolicySecret = 0x00000151

authObject→Name is TPM_RH_ENDORSEMENT (=0x4000000B)

policyRef.buffer = not available (policyRef is an Empty Buffer)

The policy digest is calculated as follows:

policyDigestnew ≔ $H_{policyAlg}$(Zero Digest || 0x00000151 || 0x4000000B)

policyDigestnew+1 ≔ $H_{policyAlg}$(policyDigestnew)

Table 14 contains the computed PolicyD values for the different hash algorithms.

PolicyUser	Value (hex)
$PolicyUser_{SHA256}$ $H_{SHA256}(H_{SHA256}(Zero\ Digest_{SHA256}$ || 0x00000151 || 0x4000000B))	837197674484b3f81a90cc8d46a5d724fd52d76e06520b64f2a1da1b331469aa
$PolicyUser_{SHA384}$ $H_{SHA384}(H_{SHA384}(Zero\ Digest_{SHA384}$ || 0x00000151 || 0x4000000B))	8bbf2266537c171cb56e403c4dc1d4b64f432611dc386e6f532050c3278c930e143e8bb1133824ccb431053871c6db53
$PolicyUser_{SHA512}$ $H_{SHA512}(H_{SHA512}(Zero\ Digest_{SHA512}$ || 0x00000151 || 0x4000000B))	1e3b76502c8a1425aa0b7b3fc646a1b0fae063b03b5368f9c4cddecaff0891dd682bac1a85d4d832b781ea451915de5fc5bf0dc4a1917cd42fa041e3f998e0ee

Table 14 - PolicyUser values

7.3.6.2 Computing Certify Policy

The policy digest for TPM2_PolicyCommandCode() is computed as defined in the TPM 2.0 Library Specification [2], Part 3:

policyDigestnew ≔ HpolicyAlg(policyDigestold || TPM_CC_PolicyCommandCode || Code)

Where policyDigestold is the Policy EH as described in Policy User above:

policyDigestnew ≔ HpolicyAlg(Zero Digest || 0x00000151 || 0x4000000B)

policyDigestnew+1 ≔ HpolicyAlg(policyDigestnew)

And so

policyDigestnew ≔ HpolicyAlg(policyDigestEH || TPM_CC_PolicyCommandCode || Code)

TPM_CC_PolicyCommandCode = 0x0000016C

Code is TPM_CC_Certify = 0x00000148

PolicyCertify results are shown in Table 15.

PolicyCertify	Value (in hex)
PolicyCertify$_{SHA256}$ H_{SHA256}(PolicyUser$_{SHA256}$ || 0x0000016C || 0x00000148)	b2a69e6391e2684a0fe752d39e14acd2e5cb922e4bd035830eea31f2aabe9870
PolicyCertify$_{SHA384}$ H_{SHA384}(PolicyUser$_{SHA384}$ || 0x0000016C || 0x00000148)	ad6007596907528f059a6121a5293c8ea3539f025640bc0920e1c9f779085934ef907a94469e25df22dbec6663b4f128
PolicyCertify$_{SHA512}$ H_{SHA512}(PolicyUser$_{SHA512}$ || 0x0000016C || 0x00000148)	bdcf7b666007254939c5fa725cb876e94ac17e033d9335a3f40267abc35b1e3e323597cd1585d991aee2d6a20e3322eae8b6f87b12953248aeb7441c01f12e60

Table 15 - Policy Certify values

7.3.6.3 Computing Activate Credential Policy

The policy digest for TPM2_PolicyCommandCode() is computed as defined in the TPM 2.0 Library Specification [2], Part 3:

policyDigestnew := HpolicyAlg(policyDigestold || TPM_CC_PolicyCommandCode || Code)

Where policyDigestold is the Policy EH as described in Policy User above:

policyDigestnew := HpolicyAlg(Zero Digest || 0x00000151 || 0x4000000B)

policyDigestnew+1 := HpolicyAlg(policyDigestnew)

And so

policyDigestnew := HpolicyAlg(policyDigestEH || TPM_CC_PolicyCommandCode || Code)

TPM_CC_PolicyCommandCode = 0x0000016C

Code is TPM_CC_ActivateCredential = 0x00000147

PolicyCertify results are shown in Table 16.

PolicyActivateCredential	Value (in hex)
PolicyActivateCredential SHA256 H_{SHA256}(PolicyUser$_{SHA256}$ || 0x0000016C || 0x00000147)	cd9917cf18c3848c3a2e606986a066c68142f9bc2710a278287a650ca3bbf245
PolicyActivateCredential SHA384 H_{SHA384}(PolicyUser$_{SHA384}$ || 0x0000016C || 0x00000147)	0a55d64c633cff7b4c782ee5464d0868a8c4d18d636f2d91f0cfc7b1beaadc1edeb6b4f545aace6d5c5cfc80c0a82492
PolicyActivateCredential SHA512 H_{SHA512}(PolicyUser$_{SHA512}$ || 0x0000016C || 0x00000147)	d291c51ea9eeabad3d858c6f4fd8bcb385e357598ff95d225405041df9ea50563f2a1c1942cb4a955ad89a6684c57365e64b208b059e188d6170825d91778342

Table 16 - Policy Activate Credential values

7.3.6.4 Computing Policy Authorize NV

The policy digest for TPM2_PolicyAuthorizeNV() is computed as defined in the TPM 2.0 Library Specification [2], Part 3:

policyDigestnew := HpolicyAlg(policyDigestold ||
TPM_CC_PolicyAuthorizeNV ||
nvIndex→Name)

Where

policyAlg = SHA256, SHA384, SHA512, or SM3_256

policyDigestold = Zero Digest (32, 48, or 64 bytes)

TPM_CC_ PolicyAuthorizeNV = 0x00000192

nvIndex→Name is the Name of the NV Index containing the Delegation Policy and using Policy User as shown below.

With the Policy Index Names, Policy C is computed as shown in Table 17.

PolicyDelegationNV	Value (in hex)
Signing Key PolicyDelegationNV$_{SHA256}$ H_{SHA256}(Zero Digest$_{SHA256}$ || 0x00000192 || Name of Policy Index I-1)	629c50b05f1adb5b4297feb241549d4217a1c792c162feb861022def88fa9501
Attestation Key PolicyDelegationNV$_{SHA256}$ H_{SHA256}(Zero Digest$_{SHA256}$ || 0x00000192 || Name of Policy Index I-1)	101e689dadf145222412c05b76e14532af1e5c74208e0ae7cdb0cff1906c8a09
Signing Key PolicyDelegationNV$_{SHA384}$ H_{SHA384}(Zero Digest$_{SHA384}$ || 0x00000192 || Name of Policy Index I-2)	0275fc2ac8ae37dc923f5c1fe9bf07bde08ccf5140d437de379a522b8c5efe31a7274415c7e8b1be1e9535003975cbcc
Attestation Key PolicyDelegationNV$_{SHA384}$ $H_{SHA384(}$Zero Digest$_{SHA384}$ || 0x00000192 || Name of Policy Index I-2)	471a7ab7d988de1e558e4db956a27f57a7e569038376984363a347001e615381d25f3c9cda70971de027571f3ddb09ca
Signing Key PolicyDelegationNV$_{SHA512}$ H_{SHA512}(Zero Digest$_{SHA512}$ || 0x00000192 || Name of Policy Index I-3)	389a4467aafc683406a8113b480926e838986a898f2e2850ae316c53fa73865f91a8182afe46b09dbc7489946ad0b24d23349925e29410a056b7445d175d1d5a
Attestation Key PolicyDelegationNV$_{SHA512}$ H_{SHA512}(Zero Digest$_{SHA512}$ || 0x00000192 || Name of Policy Index I-3)	6c38a4edec27cbd537dde645a4f106bdf6137760396db462b25463b80947597c9b49b891c789fe2cae3bd3de0930e3809012cbcb4e0e07684cc7b7ebdd4481d4

Table 17 - Policy Authorize NV values

7.3.6.5 Computing NV Index Names

These are the Names for NV Indices storing Delegation Policies as used above. The NV Index Name for each key type is computed as defined as:

Name ≔ nameAlg || HnameAlg (handle→nvPublicArea):

Where

handle→nvPublicArea is the public area of the NV Index (TPMS_NV_PUBLIC) and

TPMS_NV_PUBLIC = nvIndex || nameAlg || attributes || authPolicy || dataSize

With:

nvIndex = (select from Section 7.3.5)

nameAlg = TPM_ALG_SHA256/ SHA384/ SHA512, or TPM_ALG_SM2

attributes = 0x220F1008

authPolicy = PolicyUser (this is PolicySecret(EH), prefixed by the size)

dataSize = 0x0022, 0x0032, or 0x0042

The name for the Policy Indices I-1 to I-4 is computed as shown in Table 18.

Policy Index Names	Value (hex)
DS1 Signing Key Name: 0x000B || H_{SHA256}(0x01C9001**0** || 0x000B || 0x220F1008 || 0x0020 || $PolicyUser_{SHA256}$ || 0x0022)	000bd910fb32da43317c5e7780c0ef8c4d92d9c13505267ce1fa27acc9f9478ac412
DA1 Attestation Key Name: 0x000B || H_{SHA256}(0x01C9001**8** || 0x000B || 0x220F1008 || 0x0020 || $PolicyUser_{SHA256}$ || 0x0022)	000bfa2483df71a51f073fb0480b78cea608af43fa6e0fe584a570bcdb26a7dc4a74
DS2 Signing Key Name: 0x000C || H_{SHA384}(0x01C9001**1**|| 0x000C || 0x220F1008 || 0x0030 || $PolicyUser_{SHA384}$ || 0x0032)	000cbc4a03920691920914ce27cc6032ee484367d40b731ca87c5be9c2115e07ecfa356e8f8893ab80c78cc51af4397dd47d
DA2 Attestation Key Name: 0x000C || H_{SHA384}(0x01C9001**9**|| 0x000C || 0x220F1008 || 0x0030 || $PolicyUser_{SHA384}$ || 0x0032)	000cb84c154df940036998b9059db274e5e481798b246005775cd5b36a4852e1803b23c0dc07dbc6187dc1bf7ab0d0bba2fa
DS3 Signing Key Name: 0x000D || H_{SHA512}(0x01C9001**2**|| 0x000D || 0x220F1008 || 0x0040 || $PolicyUser_{SHA512}$ || 0x0042)	000d597a1b4d48ca1845afcc842a45509505e166cbdb834a35694c57d9be738d3345745cbdb314c39bef698d356e5def46d63f4752f679e46f3f44cfdee2f2ed3876
DA3 Attestation Key Name: 0x000D || H_{SHA512}(0x01C9001**A**|| 0x000D || 0x220F1008 || 0x0040 || $PolicyUser_{SHA512}$ || 0x0042)	000d075b2d387562022ea05b134685f4f976358ab8b8f64ec78f7ba57c35ea9c37e6899689f71440d9ee68eb9fe3b7d4190f5a95422640ec0a63f39a9c1e42ad378d

Table 18 - NV Index Policy Names

7.3.6.6 Computing IDevID/IAK Key Policy

The policy digest for TPM2_PolicyOR() is computed as defined in the TPM 2.0 Library Specification [2], Part 3:

policyDigestnew := HpolicyAlg(policyDigestold || TPM_CC_PolicyOR || digests)

Where

digests := pHashList.digests[1].buffer || … || pHashList.digests[n].buffer
policyDigestold := Zero Digest (32, 48, or 64 bytes)

With

TPM_CC_PolicyOR = 0x00000171
pHashList.digests[1].buffer = PolicyUser (section 7.3.6.1)
pHashList.digests[2].buffer = PolicyCertify (section 7.3.6.2)
pHashList.digests[3].buffer = PolicyActivateCredential (section 7.3.6.3)
pHashList.digests[4].buffer = PolicyDelegationNV (section 7.3.6.4)

PolicyIDevIDKey is computed as shown in Table 19.

PolicyIDevIDKey	**Value (in hex)**
Signing Key PolicyIDevIDKey$_{SHA256}$ H$_{SHA256}$(Zero Digest$_{SHA256}$ || 0x00000171 || PolicyUser$_{SHA256}$ || PolicyCertify$_{SHA256}$ || PolicyActivateCredential$_{SHA256}$ || PolicyDelegationNV$_{SHA256}$)	ad6b3a2284fd698a0710bf5cc1b9bdf15e2532e3f601fa4b93a6a8fa8de579ea
Attestation Key PolicyIDevIDKey$_{SHA256}$ H$_{SHA256}$(Zero Digest$_{SHA256}$ || 0x00000171 || PolicyUser$_{SHA256}$ || PolicyCertify$_{SHA256}$ || PolicyActivateCredential$_{SHA256}$|| PolicyDelegationNV$_{SHA256}$)	5437182326e414fca797d5f174615a1641f61255797c3a2b22c21d120b2d1e07
Signing Key PolicyIDevIDKey$_{SHA384}$ H$_{SHA256}$(Zero Digest$_{SHA384}$ || 0x00000171 || PolicyUser$_{SHA384}$ || PolicyCertify$_{SHA384}$ || PolicyActivateCredential$_{SHA384}$ || PolicyDelegationNV$_{SHA384}$)	4db1aa836d0b5615df6ee53a40ef70c61c217f4303d44695925972bc927006cfa5cbdf6dc18c4dbe329b2f1542c3dd33
Attestation Key PolicyIDevIDKey$_{SHA384}$ H$_{SHA256}$(Zero Digest$_{SHA384}$ || 0x00000171 || PolicyUser$_{SHA384}$ || PolicyCertify$_{SHA384}$ || PolicyActivateCredential$_{SHA384}$||	129d94ebf84556652c6eef43bbb757512ac87e52be7b349ca6ce4d826f749fcf672f51716c5cbb605f313bf345aab312
Signing Key PolicyIDevIDKey$_{SHA512}$ H$_{SHA256}$(Zero Digest$_{SHA512}$ || 0x00000171 || PolicyUser$_{SHA512}$ || PolicyCertify$_{SHA512}$ || PolicyActivateCredential$_{SHA512}$ || PolicyDelegationNV$_{SHA512}$)	7dd7500fd6c1b94f97a6af910da147301ef28f662fee06f225a4ccadda3b4e6b38e66b2f3ad5dee1a0503cd2daedb1e68cfe4f84b03a8cd22bb6a976f071a72f
Attestation Key PolicyIDevIDKey$_{SHA256}$ HSHA256(Zero Digest$_{SHA256}$ || 0x00000171 || PolicyUser$_{SHA256}$ || PolicyCertify$_{SHA256}$ || PolicyActivateCredential$_{SHA256}$ || PolicyDelegationNV$_{SHA256}$)	8060d1fb31716a29e48a6e5fece088bcfc1b278fc162255e81c3eca3544cd44af94410c3715d561cccd9e39a6cb2646d43535bb54ea88710deb5f7836bd9b586

Table 19 - IDevID/IAK Key Policy values

7.3.7 Example Delegation Policy

Start of informative comment

Figure 9 shows the constituent policies used when creating a key with policies. The Delegated policies that would apply to a key are not included in key metadata and are likely not known (e.g. by an OEM) at the time of key creation. Delegated policies are instead created and stored in one or more NV Indices by the platform Owner. This section describes an example delegated policy for IDevID and IAK keys.

The example delegated IDevID policy shown in Figure 10 allows the key to be used for signing and also allows it to be certified with an attestation key (e.g. the IAK).

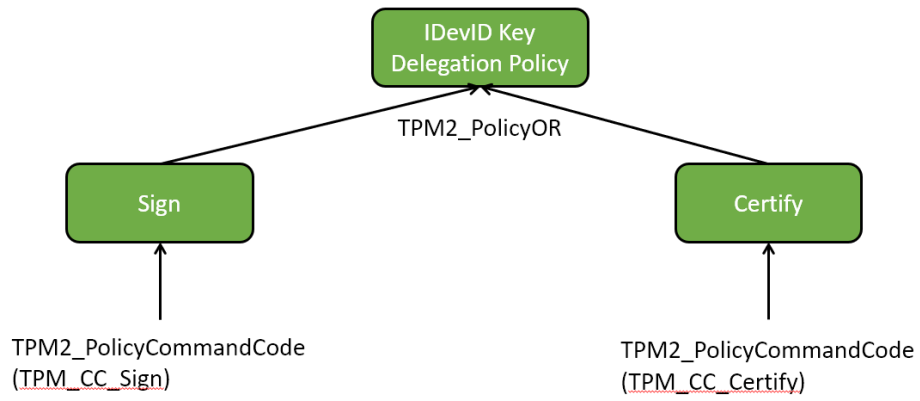

Figure 10 - Example IDevID Key Delegation Policy

The example delegated IAK policy shown in Figure 11 allows the key to be used for a Quote, for signing an Audit Digest, for Certifying another key and for releasing a secret with the ActivateCredential command.

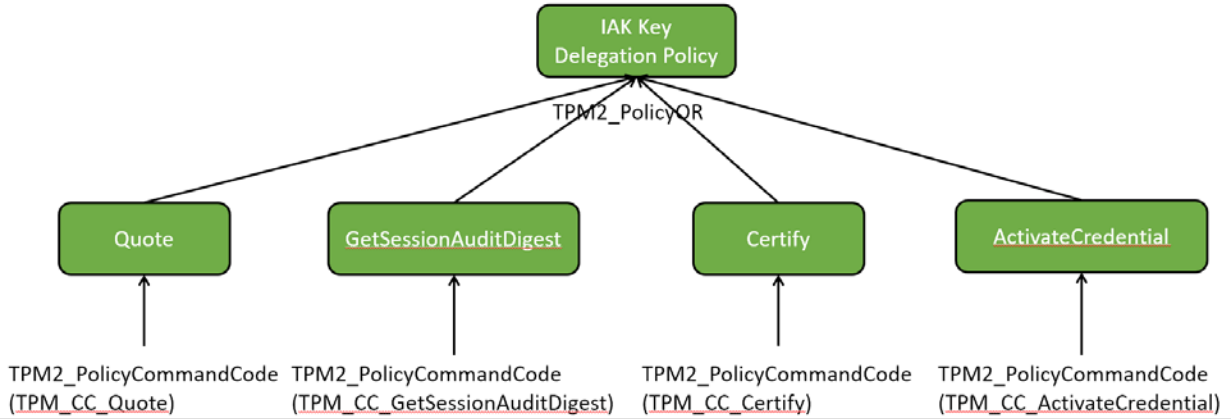

Figure 11 - Example IAK Key Delegation Policy

Table 20 provides the policy digests that represent the example Delegation Policies using three different example digest algorithms. When written into the recommended NV Indices and used with the recommended policy templates, these values implement a policy with the following characteristics:

- The IDevID signing key delegation policy:
  1. Allows use of the IDevID key for signing
  2. Allows the IDevID to be certified using the ADMIN role (e.g. using the IAK)
  3. Constrains the ADMIN role to being used only for the commands listed
- The Attestation Key delegation policy:
  1. Allows use of the IAK for Attestation (using Quote or Audited Sessions)
  2. Allows use of the IAK in key certification
  3. Allows use of the IAK with TPM2_ActivateCredential

4. Constrains the ADMIN role to being used only for the commands listed

Signing Delegation Policy	**Value (in hex)**
Policy_CommandCode Sign with leading HASH ID for SHA256	000b40c1aaa7c28fb8be9c09bdbc16c9200a40b84331068ab30f481b5a9b6efc13e0
Policy_CommandCode Sign with leading HASH ID for SHA384	000c999953e6f6127688bed63e4cd8878d37a17b9160792aca579c72045f4aafa417e6382ddd462d17ecc2e9c7d4a41a3fa2
Policy_CommandCode sign with leading HASH ID for SHA512	000d979cc3e14ba33c7699e8d6bf6032466a87500f826f0f4ceb87b1ad718311de4337b315cf9345f3f6fd10a64dcb0c588deb1217e0575640a46d6a936c1d9a6aca
Attestation Delegation Policy	**Value (in hex)**
Policy_OR CC Quote and GetSessionAuditDigest with leading HASH ID for SHA256	000bfad163be7f26cedb312ea7bcbbbf521f666ca84b592de8e6c048bb07beae774e
Policy_OR CommandCode Quote and GetSessionAuditDigest with leading HASH ID for SHA384	000c27a17d1b2062e61c3cf9431ca9a73b0e2db1310d4f54a3aadf9caaa04ea31d11e6f184ffd952c9687c3607db2f887757
Policy_OR CommandCode Quote and GetSessionAuditDigest with leading HASH ID for SHA512	000d91680fba6601542a6db7d4c69682452031ee5122d817a477fbfdf9e8158cc0f4bcf862e480ca84e3fbb3f2fd9abdae472c8f61ae9b027e443ab875141bcea2d8

Table 20 - Example Delegation Policies

End of informative comment

8 Certificate Fields in Detail

The IDevID/IAK certificates are standard X.509v3 certificates with a profile that meets the requirements of IEEE 802.1AR [6]. This section enumerates those certificate fields that contain information for a TPM-based device identity (IDevID/IAK and LDevID/LAK).

IDevID/IAK certificates associate the Distinguished Name with a TPM-based key. When IDevID/IAK certificates are created by an OEM, the subject field MUST contain a unique X.500 Distinguished Name (DN) and SHOULD include the device serial number reflected in the subject field's DN encoding as the X520SerialNumber attribute (as per 802.1AR). Note that the device serial number (X520SerialNumber) is separate and distinct from the certificate serial number (serialNumber).

Because IDevID/IAK certificates' Subject name field contains a value, the subjectAltName does not need to be a critical extension, but it MUST contain a value to identify the TPM with the HardwareModuleName and HwType. The IDevID/IAK certificates SHALL contain a TCG OID that represents TPM Version 2.0 in the hwType field together with a value in hwSerialNum. The details of the hardwareModuleName extension are described in Table 22.

8.1 DevID Certificate Fields Summary

IDevID/IAK certificates MUST be constructed to be valid when verified as described in IETF RFC-5280 [4] and IDevID/IAK certificates MUST comply with requirements of IEEE 802.1AR [6]. The DevID certificate MUST contain the following subset of fields from RFC-5280. Solutions that use the IDevID/IAK certificates SHOULD validate these fields as specified in RFC-5280.

Field Name	RFC 5280 Type	Value	TPM DevID Notes
serialNumber	serialNumber		Certificate serial number (this is not a device serial number): 1. MUST be a unique (per CA) integer. 2. MUST be >= 64-bits AND <= 160-bits in size. 3. MUST be non-negative per RFC-5280.
Validity	notBefore	Date of certificate creation.	SHALL be the date the DevID certificate is created.
	notAfter	The GeneralizedTime value "99991231235959Z" is used for "does not expire."	An IDevID/IAK is expected to exist for as long as the host IDevID/IAK module remains operational. In order to support the standard X.509 certificate format, a certificate expiration time is specified.
Subject	Name	IDevID/IAK: In compliance with 802.1AR SHOULD include Serial Number attribute. LDevID/LAK: In accordance with local CA policy.	

Table 21 - Certificate Fields Specific to TPM-Based Identities

Description of the fields shown in Table 21:

notBefore

The earliest time a certificate may be used. This SHALL be the date the IDevID/IAK certificate is created.

notAfter

The latest time a DevID certificate is expected to be used. Devices possessing an IDevID or IAK certificate are expected to operate indefinitely into the future and SHOULD use the value 99991231235959Z. Solutions verifying

an IDevID/IAK certificate are expected to accept this value indefinitely. Any other value in a DevID notAfter field is expected to be treated as specified in RFC-5280 [4].

Subject

In compliance with IEEE 802.1AR [1], Section 8.6, OEMs creating DevIDs MUST uniquely identify the device within the issuer's domain of significance. This field MUST contain a unique X.500 Distinguished Name (DN). The subject field's DN encoding SHOULD include the "serialNumber" attribute with the device's unique serial number.

In an LDevID/LAK certificate, the subject field SHOULD be the same as the subject field in the IDevID/IAK certificate.

Extensions

Field Name	RFC 5280 Type	Value	TPM DevID Notes

subjectAltName	GeneralName.otherName	OBJECT IDENTIFIER ::= {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-on ::= 8
		
		id-on-hardwareModuleName OBJECT IDENTIFIER ::= { id-on 4 }
		
		hardwareModuleName ::= SEQUENCE {
		hwType OBJECT IDENTIFIER,
		hwSerialNum OCTET STRING
		}
		
		id-on-permanentIdentifier OBJECT IDENTIFIER ::= { id-on 3 }
		
		permanentIdentifier ::= SEQUENCE {
		identifierValue UTF8String OPTIONAL,
		assigner OBJECT IDENTIFIER OPTIONAL
		}
			otherName contains two elements: id-on-hardwareModuleName and id-on-permanentIdentifier.
			
			Id-on-hardwareModuleName is defined in IETF RFC 4108 [11].
			
			The TPM 2 hwType Object Identifier is: 2.23.133.1.2
			
			Id-on-permanentIdentifier is defined in IETF RFC 4043 [12].

keyUsage	KeyUsage extension	digitalSignature (IDevID) digitalSignature (IAK) digitalSignature and dataEncipherment (Combined IDevID)	Use of digitalSignature (only) is RECOMMENDED. Refer to section 3.8. Support for older authentication protocols that use encryption means that the DevID keys MUST assert both usages.
Certificate Policy	Version 3 Extension	CP URL	The Certificate Policy extension MUST be present.

Table 22 - Certificate Extensions Used for DevID Certificates

Description of the fields shown in Table 22:

SubjectAltName

The subjectAltName extension is a standard X.509v3 extension. The X.509 certificate profile presented in RFC-5280 specifies the subjectAltName extension for allowing the binding of additional identities to the subject of the certificate. In an IDevID/IAK certificate, the subjectAltName SHOULD NOT be critical, as this could introduce interoperability concerns.

Within the subjectAltName extension, where a digest is required, the hash algorithm SHALL be SHA-256.

hardwareModuleName

The non-critical subjectAltName extension SHALL contain a hardwareModuleName as specified in RFC 4108 [10] section 5 to describe the TPM.

The object identifier for the hardwareModuleName extension is defined as:

```
id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }

subjectAltName = GeneralNames

otherName [0] OtherName

OtherName ::= Sequence {

        Id-on-hardwareModuleName OBJECT IDENTIFIER ::= iso(1) identified-organization(3) dod(6)
        internet(1) security(5) mechanisms(5) pkix(7) on(8) 4

        HardwareModuleName ::= SEQUENCE {
                hwType OBJECT IDENTIFIER
                hwSerialNum OCTET STRING
        }
}
```

The TCG registered OID (2.23.133.1.2) represents the hwType of TPM 2.0.

The hwSerialNum value is an OCTET STRING and SHALL be constructed by one of two methods:

1. When the TPM has an EK Certificate, the hwSerialNum is created by concatenating three ASCII values: The TCG TPM Manufacturer code, the EK Authority Key Identifier and the EK CertificateSerialNumber. These three fields SHALL be separated by a colon (':') character. The three values SHALL be listed in the order specified above.
2. When the TPM does not have an EK certificate, the hwSerialNum is a digest of the EK Certificate public key.

When the TPM is provisioned with both RSA and ECC endorsement key certificates, the RSA EK Certificate serialNumber SHALL be used as the EK CertificateSerialNumber. When the TPM is provisioned with both RSA and ECC endorsement keys but no EK certificates, the digest of the RSA key SHALL be used.

If an LDevID/LAK is created and an IDevID/IAK exists, then:

A) A Subject name field SHOULD be included in the LDevID/LAK certificate and SHOULD match the IDevID/IAK Subject.
B) The LDevID/LAK subjectAltName SHOULD include the entire subjectAltName from the IDevID/IAK certificate.

Note: If the LDevID/LAK creator decides to leave the subject name field empty, the subjectAltName extension MUST be critical in accordance with RFC 5280 [4].

When included, HardwareModuleName MUST be populated under subjectAltName (using OtherName within the GeneralName structure) and contain a globally unique identifier. The definition of globally unique is determined by the manufacturer or network administrator and is expected to be unique at least within the application or CA domain.

PermanentIdentifier

The non-critical subjectAltName extension SHOULD contain a PermanentIdentifier as specified by IETF RFC 4043 [12]. The PermanentIdentifier allows correlation of issued certificates to the same TPM.

The PermanentIdentifier SHALL be constructed as follows:

```
id-on-permanentIdentifier OBJECT IDENTIFIER ::= { id-on 3 }
        PermanentIdentifier ::= SEQUENCE {
                identifierValue UTF8String,
                assigner OBJECT IDENTIFIER OPTIONAL
}
```

The assigner value is the TCG OID 2.23.133.12.1 (tcg-on-ekPermIdSha256).

The identifierValue is constructed by one of the following methods:

1. When the TPM has an EK certificate, the identifierValue is a digest of the DER form of the TPM EK certificate, which is then converted to hexadecimal form using the UTF8 character set.
2. When the TPM does not have an EK certificate, the identifierValue is a digest of a concatenation of the UTF8 string “EkPubkey” (terminating NULL not included) with the binary EK public key.

IDevID/IAK certificates SHALL include both assigner and identifierValue fields. LDevID/LAK certificates MAY include only the identifierValue field, omitting the assigner field. This indicates the local nature of the certificate without requiring the local CA to obtain an assigner OID. When the assigner field is omitted, the certificate Subject DN forms part of the identifier, so comparison of permanentIdentifiers requires that both DN and identifierValue match.

Key Usage

This extension defines the purpose of the key contained in the certificate and MUST be included. When using a Signing key, digitalSignature and keyAgreement MUST be asserted according to the intended application:

- TLS 1.2 [10] requires digitalSignature for ECDHE and keyAgreement for ECDH.
- TLS 1.3 [11] requires only digitalSignature.
- When using a Combined key, keyUsage SHOULD assert keyEncipherment to allow use in key transport. Refer to RFC-5280 [4] section 4.2.1.3 for details of this extension.

Extended Key Usage

This section SHOULD be populated with the one TCG OID most appropriate for the type of key used, either “TCG-CE-FixedTPM” or “TCG-CE-FixedTPM-RestrictedKey”. Refer to section 8.2 for the details of these OIDs.

CertificatePolicy

In IDevID/IAK certificates, this extension MUST be populated as described in section 8.2.

8.2 TCG OIDs

This section lists the TCG OIDs used in this specification. CAs using these OIDs MUST comply with the requirements of section 4 in addition to requirements specified here.

An IDevID/IAK complying with this specification SHOULD include tcg-cap-verifiedTPMResidency to indicate compliance with section 4 and also one of tcg-cap-verifiedTPMFixed (IDevID) or tcg-cap-verifiedTPMRestricted (IAK).

Note that a relying party evaluating signed data purporting to be TPM internal data MUST check for the presence of tcg-cap-verifiedTPMRestricted in the IAK/LAK certificate to prevent the possibility of spoofing.

OID NAME	OID	Meaning when present in Certificate
tcg-cap-verifiedTPMResidency	2.23.133.11.1.1	Presence of this Policy OID in an end-entity certificate is an assertion by the CA that it used TCG specified methods to prove TPM residency of the key and, further, that the key is on the specified device. No data. Never Critical.
tcg-cap-verifiedTPMFixed	2.23.133.11.1.2	Presence of this Policy OID in an end-entity certificate is an assertion by the CA that it has verified that the fixedTPM bit was set in the key public area. No Data. Never Critical. This OID MUST NOT be combined with TCG-CE-FixedTPM-RestrictedKey.
tcg-cap-verifiedTPMRestricted	2.23.133.11.1.3	Presence of this Policy OID in an end-entity certificate is an assertion by the CA that it has verified both the fixedTPM and Restricted bits are set in the key public area. No data. Never Critical. This OID MUST NOT be combined with TCG-CE-FixedTPM.

Table 23 - TCG OIDs

9 Requirements for End of Life / End of Use

IEEE 802.1AR requires that use of IDevID MUST be enabled by default and it MUST be possible for the user to disable it. In the context of TPM keys, IDevIDs & IAKs are enabled and disabled by controls on use of a Hierarchy (e.g. enabling or disabling the Endorsement Hierarchy) or by use of TPM User and Admin authorizations of DevID keys.

Non-OEM configuration, including LDevID/LAK provisioning, MUST be removed when the product is “zeroized” or otherwise returned to the factory default state.

Zeroization or return to factory defaults MUST enable IDevID & IAK access and use. This allows the IDevID/IAK to be used for secure and automated enrollment and configuration procedures.

10 Security Considerations

Start of informative comment

While these security considerations may apply to a virtual platform and/or a virtual TPM, this section assumes that IDevID/IAK certificates have their primary application on physical devices using discrete TPMs.

End of informative comment

10.1 Threats and Countermeasures

Start of informative comment

1. Threat: An adversary may attempt to access or steal the DevID private key in order to spoof the identity of the device.

   Countermeasure: The private key is created and stored securely with the TPM. In order to prevent an adversary from accessing the private key and spoofing the device identity, the key is cryptographically protected and bound to the TPM and can only be decrypted within the protected execution environment of the TPM.

2. Threat: An adversary will attempt to attack the private key or interfere with the creation of the DevID during the time of its creation or by inserting false information into the provisioning infrastructure, resulting in weak or unprotected DevID keys and certificates.

   IDevID/IAK Countermeasure: Use the described TPM enrollment protocols within this specification. Monitoring the infrastructure at a manufacturing or OEM facility reduces the opportunity for manipulation of the enrollment process.

   LDevID/LAK Countermeasure: Provide monitoring and control of LDevID/LAK enrollment. Use the IDevID key and certificate to create an authenticated secure channel (e.g. with the EST protocol [8]) and use the IAK and certificate to certify the new LDevID/LAK key and provide assurance that the new key and certificate are issued to the correct device.

3. Threat: A TPM could be physically attacked or substituted with a bad, degraded or spoofed TPM.

   IDevID/IAK Countermeasure: Original equipment manufacturer (OEM) creation of an IDevID/IAK Certificate ties a key to a specific device. This creates an assertion by the system manufacturer that the TPM is properly certified and incorporated into the device and certification of the trusted building block (TBB). OEM created keys will not work with a replaced TPM, so the attack will not go undetected.

   LDevID/LAK Countermeasures: Provide monitoring and control of LDevID/LAK enrollment. Use the OEM created IAK and certificate to certify the new LDevID/LAK key; this provides assurance that the new key and certificate are issued to the correct device. Alternatively, use the EK certificate referenced in a Platform Certificate with the EK-based enrollment procedure described in section 6.6.

4. Threat: Commands to the TPM and APIs used in conjunction with the TPM could be modified or broken to result in incorrect operation of the TPM.

   Countermeasure: Use of the TCG TPM Software Stack (TSS) [5] helps to protect integrity of commands made to the TPM by making appropriate use of the TPM's security features. TPM 2.0 has been designed to encrypt the first parameter of each command, preventing capture in the clear of this parameter as it traverses the system bus. This can be extended further by attesting execution environment integrity before further use.

5. Threat: A denial of service (DOS) attack could be launched if physical access to the TPM was asserted. This DOS attack could be accomplished by using the TPM's Physical Presence command authorization to perform TPM actions causing the loss of existing keys protected by the TPM.

   Countermeasure: Platform firmware is in control of clearing the hierarchies and can prevent unauthorized access. Use of re-creatable IDevID/IAK keys allows recovery from key deletion.

6. Threat: An attacker might be able to capture data to or from the TPM.

Countermeasure: An encrypted session is used to prevent an adversary's ability to observe TPM communications.

An HMAC session can be used to provide source integrity protection.

7. Threat: An adversary will attempt an Asokan attack, which uses attestation data from a second, uninfected device to represent the state of the attesting device.

   Countermeasure: IAK certificates (and LAK certificates following recommendations of this specification) bind the key used for attestation to a particular device, ensuring that the attestation verifier knows the device identity when verifying the attestation signature.

End of informative comment

11 Privacy Considerations

This document distinguishes between User Devices and Non-User Devices, with Non-User devices referred to as Infrastructure devices. In most cases an Infrastructure device must be uniquely and unambiguously identifiable on the network. In User Device cases, the user may like to retain privacy, meaning that an identity used in one circumstance is not provably associable with an identity used in another circumstance.

An IDevID/IAK is a unique identifier meant to be a long-term identity for User or Infrastructure devices. Where privacy is a concern, it is therefore important to use an IDevID/IAK only when device based authentication is required and permitted by the user.

It might be possible to gather a lot of data about a device and/or a user when the same identity is used for different applications. It is therefore recommended that a particular LDevID/LAK be used only for one application. This specification allows creation of more than one LAK or LDevID certificate for the same device. This allows the use of pseudonyms for different applications and may preserve the privacy of the device and the user.

Recommended practice is to use the OEM installed IDevID/IAK only for secure enrollment of an LDevID/LAK using a trusted CA. Each system user may be provided with their own LDevID/LAK. This practice minimizes use of the long-term IDevID/IAK key and can help to mitigate correlation of a system's users.

When network confidentiality is a concern it is recommended that client IDevID/IAK not be used. Instead, use anonymous authentication (the client authenticates the server but not vice versa) to set up a secure tunnel. Further authentication where an IDevID/IAK is transmitted through this secure tunnel is then optional.

Selection of the appropriate TPM key hierarchy is important when DevID is used and privacy is to be preserved. This specification uses the Endorsement Hierarchy for OEM installed IDevID/IAK keys, allowing for privacy control and recreation of deleted keys. LDevID/LAK keys are created in the Storage Hierarchy so that deleted keys cannot be recreated after the TPM has been cleared. Keys created under the Platform Hierarchy are outside the scope of this specification.

12 References

[1] IEEE, "802.1AR-2018 - IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity," 14 June 2018. [Online]. Available: https://standards.ieee.org/standard/802_1AR-2018.html. [Accessed 3 March 2020].

[2] Trusted Computing Group, "Trusted Platform Module Library Specification, Family “2.0”, Level 00, Revision 01.38," September 2016. [Online]. Available: https://trustedcomputinggroup.org/tpm-library-specification. [Accessed 12 2019].

[3] Trusted Computing Group, "TPM Keys for Platform Identity for TPM 1.2," 21 August 2015. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TPM_Keys_for_Platform_Identity_v1_0_r3_Final.pdf. [Accessed 7 April 2020].

[4] D. Cooper, S.Santesson, S. Farrell, S. Boegen, R. Housley and W. Polk, "RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List Profile," May 2008. [Online]. Available: https://tools.ietf.org/rfc/rfc5280.txt. [Accessed 3 March 2020].

[5] Trusted Computing Group, "TPM Software Stack Specifications (TSS2)," 2 October 2019. [Online]. Available: https://trustedcomputinggroup.org/ work-groups/software-stack/. [Accessed 3 March 2020].

[6] S. Bradnor, "RFC 2119: Key words for use in RFCs to Indicate Requirement Levels," March 1997. [Online]. Available: https://tools.ietf.org/rfc/rfc2119.txt. [Accessed 3 March 2020].

[7] Trusted Computing Group, "TCG EK Credential Profile For TPM Family 2.0, Level 0, Specification Version 2.0, Revision 20," 3 May 2017. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-ek-credential-profile-for-tpm-family-2-0/. [Accessed 3 March 2020].

[8] J. Schaad, "RFC 4211: Internet X.509 Public Key Infrastructure Request Message Format (CRMF)," September 2005. [Online]. Available: https://tools.ietf.org/html/rfc4211.

[9] M. Pritikin, P. Yee and D. Harkins, "RFC-7030: “Enrollment Over Secure Transport”, October, 2013," October 2013. [Online]. Available: https://tools.ietf.org/rfc/rfc7030.txt. [Accessed 3 March 2020].

[10] Trusted Computing Group, "TCG Platform Attribute Credential Profile, Specification Version 1.0, Revision 12," 16 January 2018. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCG-Platform-Attribute-Credential-Profile-Version-1.0.pdf. [Accessed December 2019].

[11] R.Housley, "RFC-4108: Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages," August 2005. [Online]. Available: https://tools.ietf.org/rfc/rfc4108.txt. [Accessed 3 March 2020].

[12] T. G. D. Pinkas, "RFC 4043," 2005. [Online]. Available: https://tools.ietf.org/html/rfc4043.html. [Accessed August 2020].

[13] US NIST, "SP800-57 Part 1 Revision 4, “Recommendation for Key Management”," January 2016. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-4/final. [Accessed 3 March 2020].

[14] US NIST, "Computer Security Objects Register," [Online]. Available: https://csrc.nist.gov/projects/computer-security-objects-register/algorithm-registration#Modules. [Accessed 3 March 2020].

[15] GmSSL Project, "OIDs," [Online]. Available: http://gmssl.org/docs/oid.html. [Accessed 3 March 2020].

[16] M. Gartner, "Online Object Identifier DER Encoder," [Online]. Available: https://misc.daniel-marschall.de/asn.1/oid-converter/online.php. [Accessed 3 March 2929].

13 Annex A: TCG-CSR Structures

Start of informative comment

The TCG CSR structures are defined as C structures and are intended to be an interface between systems implemented using different tool chains on potentially different CPU architectures.

There are two versions:

An IDevID structure that includes data required to couple an IAK to the TPM as well as other data normally required for issuance of an IDevID. This structure is used any time an enrollment procedure uses the TPM EK certificate in the enrollment process.

An LDevID structure that omits the IDevID information and instead includes an OEM issued IAK certificate.

At the option of the CA, other structures MAY be used.

End of informative comment

13.1 TCG-CSR-IDEVID

Start of informative comment

The following C typedef structures are used to describe an example CSR structure.

The first byte (0x01) of the version is one, indicating an IDevID CSR.

Note that this structure is of indeterminate size. When a size is shown as 0, the item referenced is absent and no related bytes are included in the payload area.

// The TCS-CSR-IDEVID uses Big Endian byte ordering. All sizes are in bytes.

```
typedef struct TCG_CSR_t
{
    uint8_t structVer[4];   // Version 1.0 = 0x01000100
    uint8_t contents[4];    // Size of csrContents
    uint8_t sigSz[4];       // Size, in bytes, of signature
    TCG_IDEVID_CONTENT_t csrContents;
    uint8_t signature[0];   // DER encoded signature, including algorithm ID
} TCG_CSR_IDEVID;

typedef struct TCG_IDEVID_CONTENT_t
{
    uint8_t structVer[4];   // Version 1.0 = 0x00000100
    uint8_t hashAlgoId[4];  // TCG algorithm identifier for CSR hash
    uint8_t hashSz[4];      // Size, in bytes, of hash used
// Hash of all that follows is placed here
    uint8_t prodModelSz[4]; // Size of unterminated product model string
    uint8_t prodSerialSz[4];// Size of unterminated product serial number string
    uint8_t prodCaDataSz[4];// Size of CA-specific required data structure
    uint8_t bootEvntLogSz[4];// Size of boot event log
    uint8_t ekCertSZ[4];    // TPM EK cert size
    uint8_t attestPubSZ[4]; // Attestation key public size
    uint8_t atCreateTktSZ[4];           // TPM2_CertifyCreation ticket size
    uint8_t atCertifyInfoSZ[4];         // TPM2_Certify info size
    uint8_t atCertifyInfoSignatureSZ[4]; // TPM2_CertifyInfo Signature size
    uint8_t signingPubSZ[4];            // Signing key public size
    uint8_t sgnCertifyInfoSZ[4];        // TPM2_Certify info size
    uint8_t sgnCertifyInfoSignatureSZ[4];// TPM2_CertifyInfo Signature size
    uint8_t padSz[4];
// Payload bytes begin here.
```



```
// All payloads are included as byte arrays (no delimiters)

// Payload is followed by padSz bytes of random data here to make
// the structure size a multiple of 16 bytes.
   // prodModel[]
   // prodSerial[]
   // prodCaData[]
   // bootEvntLog[]
   // ekCert[]
   // attestPub[]
   // atCreateTkt[]
   // atCertifyInfo[]
   // atCertifyInfoSig[]
   // signingPub[]
   // sgnCertifyInfo[]
   // sgnCertifyInfoSig[]
   // pad[]
} TCG_CSR_IDEVID_CONTENT;
```

The public keys and signature are represented in ASN.1 DER encoded format. There are several online pages that may assist with algorithm encoding, such as the US NIST Computer Security Objects Register [13] and the GmSSL Project OIDs page [14] There is an OID encoding tool available online [15].

End of informative comment

13.2 TCG-CSR-LDEVID

Start of informative comment

The following C typedef structures are used to describe an LDevID CSR relying on OEM issued certificates.

The first byte (0x02) of the version has the value of two, indicating an LDevID CSR.

Note that this structure is of indeterminate size. When a size is shown as 0, the item referenced is absent and no related bytes are included in the payload area.

```
// The TCS-CSR-LDEVID uses Big Endian byte ordering. All sizes are in bytes.
typedef struct TCG_CSR_t
{
    uint8_t structVer[4];   // Version 1.0 = 0x02000100
    uint8_t contents[4];    // Size of csrContents
    uint8_t sigSz[4];       // Size, in bytes, of signature
    TCG_ LDEVID_CONTENT_t csrContents;
    uint8_t signature[0];   // DER encoded signature
} TCG_CSR_LDEVID;

typedef struct TCG_LDEVID_CONTENT_t
{
    uint8_t structVer[4];   // Version 1.0 = 0x02001000
    uint8_t hashAlgoId[4];  // TCG algorithm identifier for CSR hash
    uint8_t hashSz[4];      // Size, in bytes, of hash used
// Hash of all that follows is placed here
    uint8_t ekCertSZ[4];    // TPM EK cert size
    uint8_t iakCertSZ[4];   // IAK cert size
    uint8_t platCertSZ[4];  // Platform cert size
    uint8_t pubkeySZ[4];    // New public key size
    uint8_t atCertifyInfoSZ[4];          // TPM2_Certify info size
    uint8_t atCertifyInfoSignatureSZ[4]; // TPM2_CertifyInfo Signature size
    uint8_t padSz[4];
```



```
// Payload bytes begin here.
// All payloads are included as byte arrays

   // ekCert[]
   // iakCert[]
   // platCert[]
   // pubkey[]
   // atCertifyInfo[]
   // atCertifyInfoSig[]

// pad[] -- Payload is followed by padSz bytes of random data here
// to make the structure size a multiple of 16 bytes.
} TCG_CSR_LDEVID_CONTENT;
```

Public keys and signature are represented in ASN.1 DER encoded format. There are several online pages that may assist with algorithm encoding, such as the US NIST Computer Security Objects Register [12] and the GmSSL Project OIDs page [14]. There is an OID encoding tool available online [15].

End of informative comment

14 Annex B: IEEE 802.1AR PICS

Start of informative comment

Implementations that are conformant to this specification provide functionality conformant to IEEE 802.1AR-2018 as specified in the PICS Proforma section of that specification. Table 24 is a subset of 802.1AR specified capabilities and options supported by compliance with this specification.

Major capabilities and options:

Item	Feature	Status O: Optional M: Mandatory X:
RSA	Support the RSA-2048/SHA-256 signature suite	O.1
ECDSA256	Support the ECDSA P-256/SHA-256 signature suite	O.1
ECDSA384	Support the ECDSA P-384/SHA-384 signature suite	O.1
IDEVID	Contain an Initial Device Identifier (IDevID) for each of the supported signature suites	M
IDSECRET	Contain an IDevID secret for each IDevID	M
IDCERT	Contain an IDevID certificate for each IDevID	M
IDCHAIN	Contain an IDevID certificate chain for each IDevID	M
ICERTF	Include all mandatory fields in IDevID certificates	M
ICHAINF	Include all mandatory fields in IDevID chain certificates	M
RNG	Generate random numbers for key generation or use by other protocol entities	SI-9:M O
RNGSS	Use an RNG with the required security strength	M
RNGST	Prevent external access to the RNG internal state	RNG:M
XSIGN	Permit remote access to signing operations	X
XCERTF	Include any prohibited field in an IDevID certificate or chain certificate	X
LDEVID	Support the use of at least one LDevID certificate	O

Table 24 - IEEE 802.1AR PICS

Item	Feature	Status	Additional Information
NAME	Have a subject name that has not been and will not be issued to any other device by the CA issuing the certificate.	M	
DNAME	Have an X.500 DN in the subject field	M	
ALTNAME	Include the subjectAltName in each IDevID/IAK certificate	O	

		TCG:M	
HWNAME	Include the HardwareModuleName in the subjectAltName of each IDevID/IAK certificate	O TCG:M	
CERTF1	Contain the subjectKeyIdentifier	O	RECOMMENDED
CERTF2	Certificate and intermediate certificates contain version, serialNumber, signature, issuer, validity, subjectPublicKeyInfo, signatureAlgorithm and signatureValue, and authorityKeyIdentifier as specified in Clause 8.	M	

Table 25 - Certificate Fields and Extensions

End of informative comment

