TCG EK Credential Profile

For TPM Family 2.0; Level 0

Specification Version 2.0
Revision 14
4 November 2014
Published

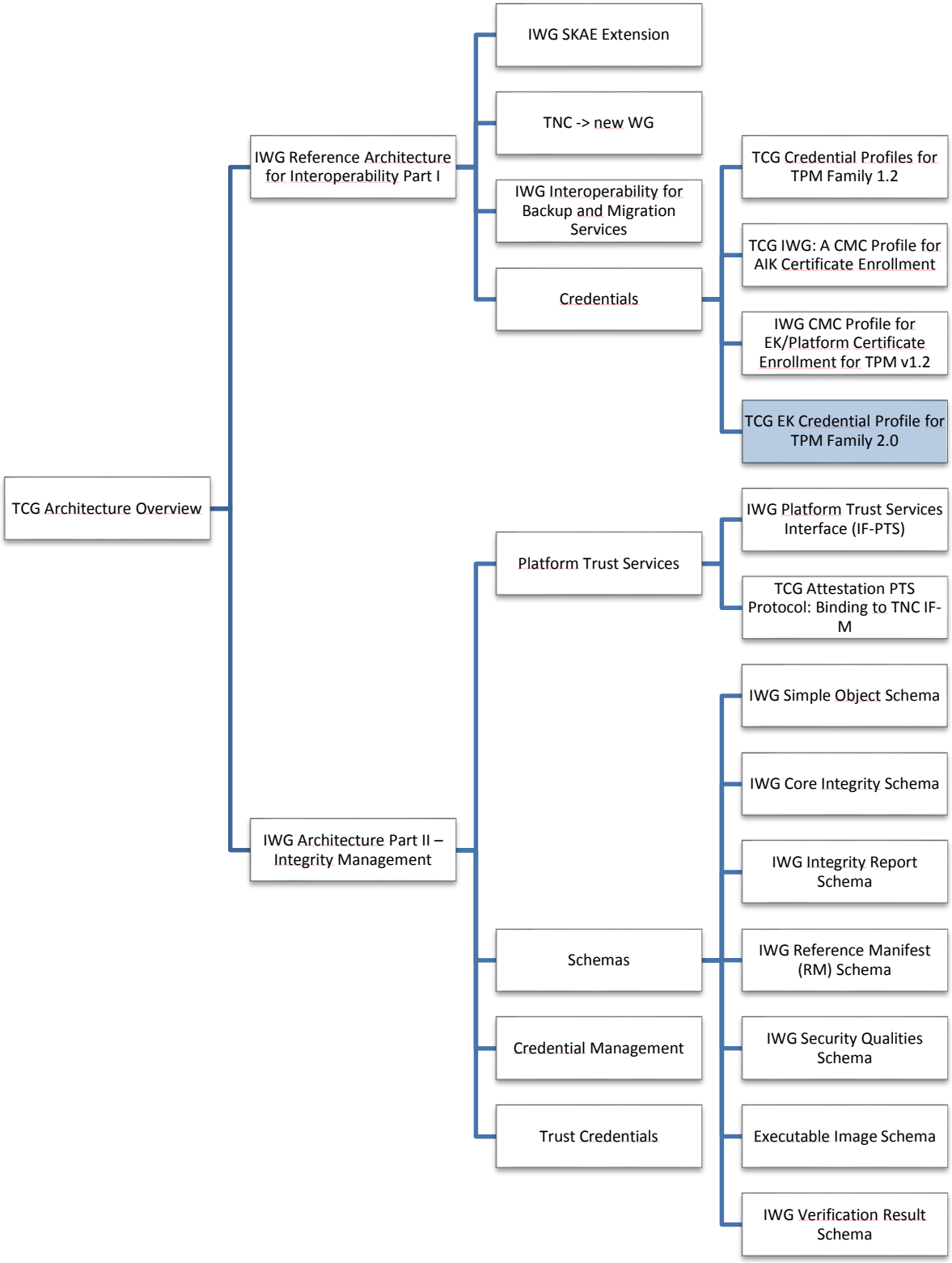
Contact:
admin@trustedcomputinggroup.org

TCG PUBLISHED

TCG

Disclaimers, Notices, and License Terms

IWG Document Roadmap



Acknowledgement

The TCG wishes to thank those who contributed to this specification. This document builds on considerable work done in the various working groups in the TCG.

Special thanks to the members of the IWG group and others contributing to this document:

Name	Member Company
Bob Bell	Cisco Systems
Max Pritikin (Group Chair)	Cisco Systems
Tom Laffey	Hewlett-Packard
Scott Kelly	Hyperthought
Ken Goldman	IBM
Ga-Wai Chin (Editor)	Infineon
Georg Rankl	Infineon
Stefan Kaeser	Infineon
Eduardo Cabre	Intel Corporation
Rahul Verma	Microsoft
Ronald Aigner	Microsoft
Olivier Collart	STMicroelectronics
Carolin Latze (Group Chair)	Swisscom
Gloria Serrao	United States Government
Greg Kazmierczak	Wave Systems

Table of Contents

Tables

1 Introduction

1.1 Purpose

The purpose of this document is to define the TPM 2.0 Endorsement Key (EK) Credential. This specification describes the content of the credential and provides an X.509 instantiation of the credential. A standardized and commonly used format should provide better interoperability between credential providers and users.

1.2 Scope

This document specifies the TPM 2.0 Endorsement Key Credential. It does not apply to TPM 1.2 credentials or credentials of other type.

1.3 Relationship to Other TCG Specifications

A TPM claiming adherence to this specification SHALL be compliant with the TPM 2.0 Library Specification[1]; Family 2.0; Level 00; Revision 00.99 or later.

1.4 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[16].

1.5 Abbreviations

CFB Cipher Feedback mode

CSR Certificate Signing Request

EK Endorsement Key

EPS Endorsement Primary Seed

IDevID Initial Device Identifier

KDF Key Derivation Function

OEM Original Equipment Manufacturer

RDN Relative Distinguished Name

TPM2_ Prefix that indicates a TPM 2.0 command

1.6 Definition of Terms

The TCG Technical Committee Glossary contains a few definitions that are fundamental to this document.

The following operational definitions, however, are specific to this specification.

Certificate – A certificate is an instantiation of a credential using the industry-standard certificate structure from ISO/IEC/ITU-T X.509 version 3. Certificate generation consists of (a) assembling values for the credential fields and (b) signing over the assembled fields.

Credential – A credential is an abstract proof that must be instantiated as a certificate before it can be exchanged between entities.

2 TPM 2.0 EK and EK Credential

2.1 Endorsement Key

The Endorsement Key (EK) is an asymmetric key pair consisting of a public and private key stored in a Shielded Location on the TPM. The public part of the EK can be read from the TPM while the private part MUST never be exposed. The public key of the EK is included in the EK certificate.

In TPM 1.2, the Endorsement Key was defined as an RSA 2048 bit key. Different than in TPM 1.2, TPM 2.0 can have more than one EK. The algorithm flexibility provided by the TPM 2.0 Library Specification[1] allows the TPM to create EKs of any type of asymmetric algorithm implemented in the TPM (see 2.1.2). The following provides a recommendation for the type of Endorsement Key for which an EK certificate is issued that is compliant to section 3.2 of this specification. In this case, the TPM 2.0 EK SHOULD be an

- RSA 2048 bit key or
- ECC NISTP-256 bit key

NOTE The recommendation is provided to enhance interoperability. However, the EK is not limited to this set of algorithms.

Any asymmetric algorithm supported in a platform-specific specification used to implement the TPM MAY be used instead or in addition to the recommended key types.

The Endorsement Key pair MUST either be generated by the TPM using the command TPM2_CreatePrimary or the TPM manufacturer MUST generate the EK in the manufacturing environment and inject it into the TPM. The Endorsement Key is stored as an object in the TPM and is referenced by its object handle.

2.1.1 Object Handle

If the Endorsement Key pair is preinstalled as a persistent object when the TPM is shipped, the handle value MUST be in the reserved handle range for Endorsement primary keys defined by the TCG in *Registry of Reserved TPM 2.0 Handles and Localities[2]*. In any other case, the EK handle value is assigned by the TPM and returned as a response parameter of the command TPM2_CreatePrimary. The EK is assigned a transient object handle until it is made persistent with the command TPM2_EvictControl. When made persistent, the EK is assigned a new object handle whose value is specified by the caller in the persistentHandle command parameter.

It is not mandatory that an Endorsement Key injected by the TPM manufacturer is preinstalled as a persistent object. The key handle MAY be returned to the caller in TPM2_CreatePrimary when the inPublic command parameter matches the public area template of the EK. This is to allow the caller to specify values for insensitive, outsideInfo and creationPCR (see *TPM 2.0 Library Specification, Part 3[1]*) which are set as additional command parameters of TPM2_CreatePrimary and which can otherwise not be assigned.

2.1.2 Primary Key Generation

The Endorsement Key is a Primary Object controlled by the Endorsement Hierarchy. The Endorsement Hierarchy has a Seed, the Endorsement Primary Seed (EPS) which is unique to each TPM. The Primary Seed is a large random value; its size is required to be at least twice the security strength of any algorithm implemented on the TPM. The EPS MUST be generated within the TPM or MUST be generated and injected by the TPM manufacturer in the manufacturing environment. The attribute TPMA_PERMANENT.tpmGeneratedEPS (see *TPM 2.0 Library Specification, Part 2[1]*) MUST be set properly to indicates the source of the Seed. The attribute can be read with TPM2_GetCapability. The Seed cannot be read from the TPM and MUST never be exposed.

The TPM 2.0 Library Specification[1] defines a process to create primary keys based on a Key Derivation Function (KDF) and a Primary Seed. The KDF is a deterministic function that uses key parameters to derive a reproducible key. These parameters determine the type of the key and are

input to the command TPM2_CreatePrimary. When this command is called with the same parameters, the same key is generated as long as the EPS does not change. The key can be made persistent in TPM NV memory using the command TPM2_EvictControl or recreated when needed. This way, any type of key (e.g. symmetric or asymmetric, signing or decryption key) can be created by the TPM.

2.1.3 EK Usage

In TPM 1.2, the Endorsement Key was defined as a decryption key; it could not be used for signing operations. Unlike TPM 1.2, TPM 2.0 provides more flexibility in defining an EK. The properties of the Endorsement Key are determined by its public area template (TPMT_PUBLIC structure, see *TPM 2.0 Library Specification Part 2[1]*). The TPMT_PUBLIC structure includes the base attributes restricted, sign and decrypt that determine the cryptographic operation a key may perform on an object. The TPM 2.0 Library Specification[1] does not impose any restrictions regarding the attributes of the Endorsement Key. As any other key the Endorsement Key can be a created as a decryption or signing key.

However, the EK and its credential may be considered privacy-sensitive if the private part of the EK is used in a cryptographic protocol. In this case, the public EK or the EK certificate may represent a privacy-sensitive cryptographic identifier for a particular platform. In privacy-sensitive environments, the EK SHOULD NOT be used as a signing key and restricted to specific operations (this is described in more detail in section 2.3 Privacy Protection).

On the other hand, there are environments where privacy is not an issue. This specification distinguishes between user device TPMs and non-user device TPMs. Whether the EK MAY sign depends on the type of platform for which the TPM is built.

2.1.3.1 User Device TPM

User device TPMs are TPMs that are associated to a human user, typically PC Client or Mobile platforms. If the EK is certified by a trusted entity, it SHOULD NOT be used for signing operations due to privacy concerns. In this case, the EK SHOULD be defined as a restricted decryption key. User device TPMs SHOULD follow the same privacy constraints as TPM 1.2. For a decryption key, the key usage field in the EK Credential defined in section 3.2.15 MUST be set to keyEncipherment (for an RSA EK) or keyAgreement (for an ECC EK).

2.1.3.2 Non-User Device TPM

Non-user device TPMs on the other hand are TPMs that are associated to an enterprise, rather than a specific user. This can be e.g.

- Network Elements (e.g. routers, switches, wireless access points)
- Servers, Virtual Servers, Virtual Devices in a cloud infrastructure
- Embedded Devices (e.g. printers)

For such platforms privacy is not a central concern and unique identification is of critical importance. These platforms MAY use a certified EK for signing operations. This is intended to facilitate establishment of further TPM keys, like Device Identification keys, without the need for an Attestation Key. This allows for simpler infrastructure implementations. In this case, no restrictions other than defined in the TPM 2.0 Library Specification[1] apply to the settings of the base attributes. The EK MAY be defined as a general-purpose key if both, signing and decryption should be supported. The key usage field in the EK Credential defined in section 3.2.15 MUST be set appropriately to indicate the usage of the EK. If decryption is supported, keyEncipherment (for an RSA EK) or keyAgreement (for an ECC EK) MUST be set; if signing is supported, digitalSignature MUST be set.

One use case for a signing EK is to sign the Certificate Signing Request (CSR) for an Initial Device Identifier (IDevID) key. The IDevID key is a TPM-generated key that is used as an initial identity for secure device authentication (see IEEE 802.1AR[9]). The CSR can be signed with the command TPM2_Sign. The hash calculated over the certification request information is passed to the TPM in

the digest command parameter; the inScheme command parameter specifies the signing scheme. The issuer of the IDevID certificate could verify the CSR signature to ensure that the chip requesting the IDevID certificate is privileged to receive it. Therefore the issuer could have a list of EK certificates of all valid TPMs a product manufacturer has purchased. Alternatively, a CSR could also be signed by an Attestation Key.

2.1.4 EK Lifetime

In TPM 2.0, the lifetime of an Endorsement Key is tied to the Endorsement Primary Seed (EPS). As long as the EPS is not changed, EKs can be recreated with their public area templates. The command TPM2_ChangeEPS replaces the Endorsement Primary Seed with a new random value and makes it impossible to recreate any EKs derived from the previous Seed. This will invalidate all certificates associated with the EKs.

Platform-specific specifications determine whether the command TPM2_ChangeEPS is required to be implemented. Some platforms might want to change the EPS, e.g. during platform refurbishment to erase existing EKs or after a field upgrade from a firmware that had a severe security flaw (in order to revoke all EKs associated with the old firmware).

On the other hand, there are platforms that need a permanent EPS because invalidating the Endorsement Keys would disable the platform to prove that it is a genuine trusted platform. In non-user device TPMs (see 2.1.3.2), for instance, the EPS is required to be permanent because the EK represents the trust anchor for the device identity.

The TPM 2.0 Library Specification[1] provides a means to prevent the EKs from being replaced. The command to change the EPS requires Platform Authorization, so the OEM can decide if the EPS ever changes. The use of Platform Authorization can be disallowed by turning off the Platform hierarchy (by setting the phEnable flag to CLEAR); this disables any functionality in the TPM that would require platformAuth or platformPolicy. Furthermore, TPM2_ChangeEPS can be added to the list of commands that require assertion of Physical Presence. Alternatively, platforms could prevent EKs from being erased by not exposing this functionality to the user.

2.1.5 Default EK Public Area Template

The following section defines the default values of an RSA and ECC EK public area template. These default values are used to generate the RSA and ECC Endorsement Key corresponding to the EK certificates installed by the TPM or Platform manufacturer. The hash calculated over the public area template is one of the command parameters to the KDF that is used to create the EK. The TPM or Platform manufacturer MAY create a proprietary EK public area template that is different from the defaults below (this procedure is further described in section 2.2.1). The values provided in the tables below would be typically used by PC Client platforms. Platform-specific working groups MAY define a proprietary EK public area template if required. This might be necessary e.g. for Non-User Device TPMs that require an EK for signing purposes. However, the EK provided by the manufacturer MUST be defined as a non-duplicable key. This is ensured by setting the following two attributes:

- fixedTPM = 1
- fixedParent = 1

When privacy is an issue, great care should be taken when selecting the attributes and the authorization for the key (see section 2.3 privacy protection).

2.1.5.1 RSA Template

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_RSA
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA256
objectAttributes	TPMA_OBJECT	fixedTPM = 1 stClear = 0 fixedParent = 1 sensitiveDataOrigin = 1 userWithAuth = 0 adminWithPolicy = 1 noDA = 0 encryptedDuplication = 0 restricted = 1 decrypt = 1 sign = 0
authPolicy	TPM2B_DIGEST	
size	UINT16	32
buffer	BYTE	0x83, 0x71, 0x97, 0x67, 0x44, 0x84, 0xB3, 0xF8, 0x1A, 0x90, 0xCC, 0x8D, 0x46, 0xA5, 0xD7, 0x24, 0xFD, 0x52, 0xD7, 0x6E, 0x06, 0x52, 0x0B, 0x64, 0xF2, 0xA1, 0xDA, 0x1B, 0x33, 0x14, 0x69, 0xAA TPM2_PolicySecret(TPM_RH_ENDORSEMENT), see 2.1.5.3
parameters	TPMS_RSA_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_AES
symmetric->keyBits	TPMI_AES_KEY_BITS	128
symmetric->mode	TPMI_SYM_MODE	TPM_ALG_CFB
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ASYM_SCHEME	TPM_ALG_NULL
scheme->details		NULL
keyBits	TPMI_RSA_KEY_BITS	2048
exponent	UINT32	0
unique	TPM2B_PUBLIC_KEY_RSA	
size	UINT16	256
buffer	BYTE	All 0

Table 1: Default RSA EK Public Area Template (TPMT_PUBLIC)

The default RSA EK public area template in Table 1 specifies an RSA 2048 bit key. Its objectAttributes indicate that the Endorsement Key is a restricted non-duplicable decryption key and its authorization is only allowed with authPolicy. The policy requires endorsementAuth to authorize the EK. This policy is created with TPM2_PolicySecret where the authHandle parameter of the command, indicating the entity providing the authorization value, references the Endorsement hierarchy. The symmetric key is used to protect the child keys of the EK and is defined as an AES 128 bit key using CFB mode. SHA256 is used to calculate the Name of the EK. The buffer reserved for the public key of the EK is set to all zeros.

2.1.5.2 ECC Template

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_ECC
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA256
objectAttributes	TPMA_OBJECT	fixedTPM = 1 stClear = 0 fixedParent = 1 sensitiveDataOrigin = 1 userWithAuth = 0 adminWithPolicy = 1 noDA = 0 encryptedDuplication = 0 restricted = 1 decrypt = 1 sign = 0
authPolicy	TPM2B_DIGEST	
size	UINT16	32
buffer	BYTE	0x83, 0x71, 0x97, 0x67, 0x44, 0x84, 0xB3, 0xF8, 0x1A, 0x90, 0xCC, 0x8D, 0x46, 0xA5, 0xD7, 0x24, 0xFD, 0x52, 0xD7, 0x6E, 0x06, 0x52, 0x0B, 0x64, 0xF2, 0xA1, 0xDA, 0x1B, 0x33, 0x14, 0x69, 0xAA TPM2_PolicySecret(TPM_RH_ENDORSEMENT), see 2.1.5.3
parameters	TPMS_ECC_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_AES
symmetric->keyBits	TPMI_AES_KEY_BITS	128
symmetric->mode	TPMI_SYM_MODE	TPM_ALG_CFB
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ECC_SCHEME	TPM_ALG_NULL
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	
x->size	UINT16	32
x->buffer	BYTE	All 0
y->size	UINT16	32
y->buffer	BYTE	All 0

Table 2: Default ECC EK Public Area Template (TPMT_PUBLIC)

The default ECC EK public area template in Table 2 specifies an ECC NISTP-256 bit key. Its objectAttributes indicate that the Endorsement Key is a restricted non-duplicable decryption key and its authorization is only allowed with authPolicy. The policy requires endorsementAuth to authorize the EK. This policy is created with TPM2_PolicySecret where the authHandle parameter of the command, indicating the entity providing the authorization value, references the Endorsement hierarchy. The symmetric key is used to protect the child keys of the EK and is defined as an AES 128 bit key using CFB mode. SHA256 is used to calculate the Name of the EK. The buffer reserved for the public key of the EK is set to all zeros.

2.1.5.3 authPolicy

The equations below are copied from the TPM 2.0 Library Specification Part 3[1] and indicate how the authPolicy value, used in the default RSA and ECC EK public area templates (see Table 1 and 2), is calculated using the TPM2_PolicySecret command. If there is an inconsistency between the equations below and the equations defined in the TPM 2.0 Library Specification, the definitions in the Library Specification take precedence.

TPM2_PolicySecret uses the PolicyUpdate function:

PolicyUpdate(TPM_CC_PolicySecret, authObject→Name, policyRef)

This is equivalent to:

$policyDigest_{new} := H_{policyAlg}(policyDigest_{old} \,||\, TPM\_CC\_PolicySecret \,||\, authObject \rightarrow Name)$
$policyDigest_{new+1} := H_{policyAlg}(policyDigest_{new} \,||\, policyRef.buffer)$

With:

policyAlg = SHA256
$policyDigest_{old}$ = 0x0...0 (32 bytes)
TPM_CC_PolicySecret = 0x00000151
authObject→Name is TPM_RH_ENDORSEMENT (=0x4000000B)
policyRef.buffer = not available

The authPolicy in the default EK public area template is calculated as follows:

$policyDigest_{new} := H_{SHA256}(0x0...0 \,||\, 0x00000151 \,||\, 0x4000000B)$
$policyDigest_{new+1} := H_{SHA256}(policyDigest_{new})$

2.2 Endorsement Key Credential

The Endorsement Key Credential is an X.509 v3 certificate that contains the public EK, as well as various assertions regarding the security qualities and provenance of the TPM. The definition of the certificate fields are specified in section 3.2. The EK Credential is usually issued by a TPM or Platform manufacturer during manufacturing process. An entity SHALL NOT create an EK Credential for a TPM unless the entity is satisfied that the public key referenced in the EK Credential was either:

- returned in response to a TPM2_CreatePrimary command by an implementation of Protected Capabilities and Shielded Locations that meets the TPM 2.0 Library Specification[1] or
- generated outside the TPM and inserted by a process defined in the Target of Evaluation (TOE) of the security target in use to evaluate the TPM.

There might be use cases where it is useful to issue an EK Credential after manufacturing (e.g. if the EPS was changed or the TPM is shipped without EK). In this case, the entity issuing the credential would create a new Endorsement Key with TPM2_CreatePrimary. This procedure would require support for certificate enrollment. Support for an enrollment protocol is optional and MAY be done using a proprietary method of the TPM or Platform manufacturer or a method standardized by TCG. One example implementation (at time of writing, only available for TPM 1.2) is described in the IWG document *CMC Profile for EK/Platform Certificate Enrollment[6]*.

A primary use case of an EK Credential is to assist Attestation CAs to issue credentials for restricted signing keys (Attestation Keys). The EK Credential can be used to provide evidence that the Attestation Keys are resident on the same TPM.

In TPM 2.0, multiple EKs can be derived from a single Seed (as described in section 2.1.2.) As a result, the TPM can have more than one EK Credential. However, the TPM might not be provisioned with all the credentials because of NV space restrictions; the credentials could be stored encrypted off the TPM. If an EK Credential is stored on the TPM, it is stored as an NV Index; in this case, it is referenced by its NV Index handle (see 2.2.1). The authorization to modify or access (read, write, delete) the credential is determined by its attributes. The attributes of the Index are defined by platform-specific workgroups, as well as the authorization for the index. Definitions specific to PC Client can be found in *PC Client Specific Platform TPM Profile for TPM 2.0[5],* section Non-volatile Storage.

2.2.1 NV Index Handles

The NV Index handles related to the EK Credential have fixed values that are defined by the TCG in the *Registry of Reserved TPM 2.0 Handles and Localities[2]*. The values are assigned by individual workgroups and might therefore differ for different platforms. The TCG Registry defines three types of handles which are described below. Each handle is associated with a separate NV Index in the TPM.

- EK Certificate
  This is the NV Index handle of an RSA or ECC EK Credential. If an EK certificate is present, this MUST be used.
- EK Nonce
  This is the NV Index handle of a nonce value (encoded as a sized array) that is included in the unique field of the TPMT_PUBLIC template. The nonce is used to provide additional entropy in the key creation process of the EK. This is optional and only required if non-default values are used. (The default values are defined in Table 1 and 2, section 2.1.5.)
- EK Template
  This is the NV Index handle of a TPMT_PUBLIC template. The EK Template allows the credential provider to implement a proprietary template. This is optional and only required if

non-default values are used. (The default values are defined in Table 1 and 2, section 2.1.5.)

The following steps describe the procedure to create the Endorsement Key corresponding to an EK Credential:

1. Read the EK Nonce from the TPM NV memory, if present.
2. Read the EK Template from the TPM NV memory, if present.
3. If an EK Template was read from the TPM NV memory, use it as the public area template, otherwise use the default public area template defined in Table 1 and 2, section 2.1.5.
4. If the EK Nonce was read from the TPM NV memory, insert it into the unique field of the public area template, ignoring the length field of the EK Nonce, and padding the rest of the unique field with zeros. Otherwise, leave the unique field from step 3 as it is.
5. Create the EK using the public area template so prepared in the command TPM2_CreatePrimary.

NOTE A platform-specific working group is allowed to define a proprietary default EK public area template that can be used instead of the templates defined in Table 1 and 2.

2.2.2 Assertions

In general, an EK Credential asserts that the holder of the private EK is a TPM conforming to TCG specifications. Since the EK Credential is a public key credential, then by definition the signature of the issuer binds the public key material and the subject of the credential, which is a particular TPM device.

More specifically, an EK Credential asserts:

- Mandatory TPM specification compliance: The TPM device correctly implements the Protected Capabilities and Shielded Locations according to a particular version of the TPM specification set, especially the protection of the private Endorsement Key (EK). The TPM specification is described by family, level, and revision. The TPM device MUST be fully described by the following three data items: TPM manufacturer, TPM part number, and TPM firmware version. The values are manufacturer-specific.
- Optional TPM security assertions: The EK Credential MAY include assertions that it meets various evaluation conformance criteria or that it was manufactured or initialized under certain specified conditions.

To meet the assertion requirements listed above, an EK Credential MUST contain the following information fields:

- EK public key
- TPM spec version
- TPM manufacturer, TPM part number, and TPM firmware version

The assertions in the EK Credential provide information about the TPM which may be useful for an Attestation CA. The Attestation CA can verify whether the TPM conforms to a required TPM specification version and validate other properties of the TPM implementation. Based on this information, the CA may evaluate whether or not the TPM should receive an Attestation Key Credential. However, this specification does not provide any guidelines for Attestation CAs how to evaluate the content of an EK certificate.

2.2.3 EK Credential Lifetime

An EK Credential contains fields that express the validity period of the credential. The validity period is at the discretion of the manufacturer. The credential is not expected to expire during the normal life expectancy of the platform in which it resides. The lifetime can vary widely between different types of platforms (e.g. while a typical validity period for a PC Client platform is 5-10 years,

non-user device TPMs are expected to operate indefinitely into the future in which case the value 99991231235959Z should be used as expiration date). The credential lifetime can also depend on the lifetime of the TPM device and the algorithm type of the Endorsement Key. The time frame during which the security strength of the EK is acceptable SHOULD be taken into account by the manufacturer when determining the credential lifetime (e.g. see SP800-57[10]).

However, an EK Credential can become useless before expiration of the validity period if the associated EK is irrevocably erased from the TPM. This is the case if the EPS is replaced (see 2.1.4 EK Lifetime).

In TPM 1.2, the EK Credential was defined as an NV index that had the D bit set (see [7]). This way, the credential could not be deleted after the TPM has been locked. TPM 2.0 provides equivalent functionality to define a permanent NV Index. In TPM 2.0, the NV Index attributes TPMA_NV_PLATFORMCREATE and TPMA_NV_POLICY_DELETE (see TPM *2.0 Library Specification, Part 2[1]*) determine the authorization required to delete an NV Index.

TPMA_NV_PLATFORMCREATE indicates whether the NV Index was defined by the platform. If SET, the index may only be undefined with Platform Authorization and not with Owner Authorization. TPMA_NV_PLATFORMCREATE SHOULD be SET for an EK Credential to prevent the credential from being deleted if the Owner is cleared. Platform-defined NV indices in addition can SET TPMA_NV_POLICY_DELETE.

If TPMA_NV_POLICY_DELETE is SET, the Index may not be deleted unless the authPolicy of the NV Index is satisfied using the command TPM2_NV_UndefineSpaceSpecial. A platform that requires a permanent EK Credential would not create a policy that allows the EK Credential to be removed. On the other hand, a platform that wants to clear the EK Credential, e.g. during platform refurbishment, could create a policy that includes the command TPM2_PolicyCommandCode where the command code is set to TPM_CC_NV_UndefineSpaceSpecial.

The settings of the NV Index attributes are determined by Platform-specific specifications. Definitions specific to PC Client can be found in *PC Client Specific Platform TPM Profile for TPM 2.0[5],* section Non-volatile Storage.

2.3 Privacy Protection

In TPM 2.0, privacy-sensitive operations are controlled by the Privacy Administrator. The Privacy Administrator controls the Endorsement hierarchy and sets the hierarchy authorization and policy (endorsementAuth and endorsementPolicy). The Privacy Administrator and the Owner are often the same entity.

Because an Endorsement Key is unique to a TPM and usually has a long lifetime it could be used to identify a user or a platform. Therefore, the EK and its certificate may be privacy-sensitive. The following applies, if protection of privacy is important:

- The use of the EK SHOULD be limited, through policy, so that authorization from the Privacy Administrator is always required. This can be enforced by using an authPolicy that requires endorsementAuth or endorsementPolicy, authorization with authValue should be disabled. This prevents the EK from being used without permission of the Privacy Administrator.
- The usage of the EK SHOULD be limited by its object attributes, so the EK can only be authorized for specific commands. The EK SHOULD be defined as a non-duplicable restricted decryption key. This prevents the EK from being used for signing operations.
- The EK and EK certificate SHOULD NOT be considered public, and SHOULD be available only to those entities which are trusted by the Privacy Administrator.

The availability of the EK can be controlled with the flag ehEnable. The purpose of the flag is to enable and disable the Endorsement hierarchy. When the Endorsement hierarchy is disabled (ehEnable CLEAR) objects defined under that hierarchy are inaccessible, endorsementAuth and endorsementPolicy cannot be used for authorization. It is not possible to use the EK in any command or read the public EK with TPM2_ReadPublic. The ehEnable flag may be cleared with the command TPM2_HierarchyControl using Endorsement Authorization or Platform Authorization.

Protection for the EK Credential can be provided by the flag phEnableNV if TPMA_NV_PLATFORMCREATE is SET in the NV Index attributes of the EK Credential. This attribute indicates whether the index was defined by the platform. When phEnableNV is CLEAR, NV space defined by the platform firmware is not accessible, including the EK Credential. This flag can only be cleared by Platform Authorization.

3 X.509 ASN.1 Definitions

This section contains the format for the EK Credential instantiated as an X.509 certificate. All fields are defined in ASN.1 and encoded using DER[18]. The appropriate OIDs are defined in section 5.

Version 3 of the X.509 certificate structure is used for compatibility with existing PKI tools and services. TCG credential profiles do not utilize all aspects of X.509 defined fields and some fields are overloaded with TCG specific interpretations. The following sections define TCG interpretations for X.509 certificates.

TCG defines a number of new attribute value types to hold TCG-specific values. When present in a public key certificate they are carried in the subject alternative name or subject directory attributes extension.

This specification is a profile of RFC 5280[11] which is itself a profile of the ISO/IEC/ITU-T X.509 specifications for public key certificates. All syntax and semantics are inherited from those specifications unless explicitly documented otherwise below.

3.1 TCG Attributes

3.1.1 TPM Security Assertions

This attribute describes security-related assertions about the TPM.

Each attribute begins with a version number which identifies the version of the assertion syntax. Future versions of this profile may add new assertions by appending new fields at the end of the ASN.1 SEQUENCE and increasing the version number to identify which version of the assertion syntax is encoded.

The **fieldUpgradable** BOOLEAN indicates whether the TPM is capable of having its firmware upgraded after manufacturing.

The **ekGenerationType** indicates how the Endorsement Key in the TPM was created. It may be internally generated within the TPM, generated externally and then inserted under a controlled environment during manufacturing. The revocable variants indicate whether the EK Credential can be revoked or not.

In the **CommonCriteriaMeasures**, the profile and target for the evaluation can be described by either an OID, a URI to a document describing the value, or both. If both are present, they must represent consistent values. The URI values are included in an **URIReference** which describes the URI to the document and a cryptographic hash value which identifies a specific version of the document.

URIMAX is a constant used to provide an upper bound on the length of a URI included in the certificate. This upper bound may be helpful to consumers of the extension and also helps limit the overall size of the certificate. In order to provide a reasonable upper bound for ASN.1 parsers, URIMAX SHOULD NOT exceed a value of 1024. This value was selected as it matches the length limit for <A> anchors in HTML as specified by the SGML declaration (LITLEN) for HTML[17].

STRMAX is a constant defining the upper bound on the length of a string type. Like the URIMAX this is to aid ASN.1 parsers and help limit the upper bound on the length of the certificate. Based on the expected sizes of the strings in the ASN.1 in this document an upper bound of 256 was selected. STRMAX SHOULD NOT exceed a value of 256.

```
Version ::= INTEGER { v1(0) }


tPMSecurityAssertions ATTRIBUTE ::= {
    WITH SYNTAX TPMSecurityAssertions
```

```
            ID tcg-at-tpmSecurityAssertions }

        TPMSecurityAssertions ::= SEQUENCE {
            version Version DEFAULT v1,
            fieldUpgradable BOOLEAN DEFAULT FALSE,
            ekGenerationType [0] IMPLICIT EKGenerationType OPTIONAL,
            ekGenerationLocation [1] IMPLICIT EKGenerationLocation OPTIONAL,
            ekCertificateGenerationLocation [2] IMPLICIT
                EKCertificateGenerationLocation OPTIONAL,
            ccInfo [3] IMPLICIT CommonCriteriaMeasures OPTIONAL,
            fipsLevel [4] IMPLICIT FIPSLevel OPTIONAL,
            iso9000Certified [5] IMPLICIT BOOLEAN DEFAULT FALSE,
            iso9000Uri IA5STRING (SIZE (1..URIMAX) OPTIONAL }

        EKGenerationType ::= ENUMERATED {
            internal (0),
            injected (1),
            internalRevocable(2),
            injectedRevocable(3) }

        EKGenerationLocation ::= ENUMERATED {
            tpmManufacturer (0),
            platformManufacturer (1),
            ekCertSigner (2) }

        EKCertificateGenerationLocation ::= ENUMERATED {
            tpmManufacturer (0),
            platformManufacturer (1),
            ekCertSigner (2) }

        -- common criteria evaluation

        CommonCriteriaMeasures ::= SEQUENCE {
            version IA5STRING (SIZE (1..STRMAX)), -- "2.2" or "3.1"; future syntax defined
by CC
            assurancelevel EvaluationAssuranceLevel,
            evaluationStatus EvalutionStatus,
            plus BOOLEAN DEFAULT FALSE,
            strengthOfFunction [0] IMPLICIT StrengthOfFunction OPTIONAL,
            profileOid [1] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
            profileUri [2] IMPLICIT URIReference
         OPTIONAL,
            targetOid [3] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
```

```
    targetUri [4] IMPLICIT URIReference OPTIONAL }

EvaluationAssuranceLevel ::= ENUMERATED {
    level1 (1),
    level2 (2),
    level3 (3),
    level4 (4),
    level5 (5),
    level6 (6),
    level7 (7) }

StrengthOfFunction ::= ENUMERATED {
    basic (0),
    medium (1),
    high (2) }

-- Reference to external document containing information relevant to this subject.
-- The hashAlgorithm and hashValue MUST both exist in each reference if either
-- appear at all.
URIReference ::= SEQUENCE {
    uniformResourceIdentifier IA5String (SIZE (1..URIMAX),
    hashAlgorithm AlgorithmIdentifier OPTIONAL,
    hashValue BIT STRING OPTIONAL }

EvaluationStatus ::= ENUMERATED {
    designedToMeet (0),
    evaluationInProgress (1),
    evaluationCompleted (2) }

-- fips evaluation

FIPSLevel ::= SEQUENCE {
    version IA5STRING (SIZE (1..STRMAX)), -- "140-1" or "140-2"
    level SecurityLevel,
    plus BOOLEAN DEFAULT FALSE }

SecurityLevel ::= ENUMERATED {
    level1 (1),
    level2 (2),
    level3 (3),
    level4 (4) }
```

3.1.2 TPM Device Attributes

The following definitions define the syntax of the relative distinguished names (RDNs) used in the subject alternative name extension to identify the type of the TPM.

The value of the **TPMManufacturer** attribute MUST be the ASCII representation of the hexadecimal value of the 4 byte vendor identifier defined in the TCG *Vendor ID Registry[3]*. Each byte is represented individually as a two digit unsigned hexadecimal number using the characters 0-9 and A-F. The result is concatenated together to form an 8 character name which is appended after the lower-case ASCII characters “id:”.

For example, the vendorId 0x12 0x34 0x56 0xEF would be encoded as “id:123456EF”.

Likewise, the value of the **TPMVersion** attribute MUST be the ASCII representation of the hexadecimal value of the 4 bytes derived from the major and minor firmware version of the TPM. Each byte is represented individually as a two digit unsigned hexadecimal number using the characters 0-9 and A-F. The result is concatenated together to form a 8 character name which is appended after the lower-case ASCII characters “id:”.

For example, a revMajor of 0x0002 and revMinor of 0x0008 would be encoded as “id:00020008”.

The value of the **TPMModel** attribute is a UTF 8 string that represents the TPM part number. The values are manufacturer-specific.

```
TPMManufacturer ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-tpmManufacturer }

TPMModel ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-tpmModel }

TPMVersion ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-tpmVersion }
```

3.1.3 TPM Specification Attributes

The following definitions define the syntax of the TPM specification attributes.

The **TPMSpecification** attribute identifies the TPM family, level and revision of the TPM specification with which a TPM implementation is compliant. The family value of “2.0” with level 0 and revision 99 identifies a TPM compliant with a public TPM 2.0 specification version 0.99 published by TCG. The family value is encoded in a UTF 8 string but the current defined standard values fall within the ASCII character set.

```
tPMSpecification ATTRIBUTE ::= {
    WITH SYNTAX TPMSpecification
    ID tcg-at-tpmSpecification }

TPMSpecification ::= SEQUENCE {
    family UTF8String (SIZE (1..STRMAX)),
    level INTEGER,
    revision INTEGER }
```

3.2 EK Certificate

This section contains the format for a TPM 2.0 EK Credential conforming to this specification. An X.509 EK certificate is an instantiation of the TPM EK Credential defined in section 2.2.

The “Field Status” column in the table below specifies the presence of the certificate fields. The value “Standard” means the field is an inherent component of the standard certificate syntax and is not optional. The value “MUST”, “SHOULD” or “MAY” is used to indicate the presence of the certificate extensions. The content is described in the “Value” column. Values marked with “(optional)” are added for completeness and are meant to be optional.

NOTE This specification does not preclude the use of other certificate extensions. However, any extensions marked as critical will cause interoperability problems when existing clients do not know how to parse the extension and reject it as specified in RFC 5280[11], section 4.2. (This has historically been a challenge when introducing new critical extensions.)

Field Name	**RFC 5280 Type**	**Value**	**Field Status**
Version	INTEGER	V3 (encoded as value 2)	Standard
Serial Number	INTEGER	Positive integer	Standard
Signature Algorithm	AlgorithmIdentifier	sha256WithRSAEncryption or ecdsa-with-SHA256	Standard
Issuer	Name	Name of issuing CA	Standard
Validity	notBefore notAfter	Beginning and end of validity period	Standard
Subject	Name	Unique name assigned by the manufacturer or empty	Standard
Subject Public Key Info	SubjectPublicKeyInfo	RSA 2048 bit key rsaEncryption or ECC NISTP-256 bit key id-ecPublicKey	Standard
Extensions			
Certificate Policies	CertificatePolicies	CertPolicyId CPSuri (optional) UserNotice (optional)	MUST non-critical
Subject Alternative Name	GeneralName directoryName	TPM manufacturer TPM part number TPM firmware version TPM serial nr (optional)	MUST critical/ non-critical (dep. on subject)
Basic Constraints	BasicConstraints	CA=FALSE	MUST critical

Field Name	RFC 5280 Type	Value	Field Status
Subject Directory Attributes	SubjectDirectoryAttributes	TPMSpecification Family Level Revision TPMSecurityAssertions (optional)	MUST non-critical
Authority Key Id	AuthorityKeyIdentifier	Key identifier Issuer name and serial number (optional)	MUST non-critical
Authority Info Access	AuthorityInfoAccessSyntax	id-ad-caIssuers URI to issuing CA id-ad-ocsp (optional) URI to OCSP responder	SHOULD non-critical
CRL Distribution	CRLDistributionPoints	URI to CRL	MAY non-critical
Key Usage	KeyUsage	keyEncipherment or keyAgreement or digitalSignature	MUST critical
Extended Key Usage	ExtKeyUsageSyntax	tcg-kp-EKCertificate	SHOULD non-critical
Subject Key Id	SubjectKeyIdentifier	Key identifier	MAY

Table 3: EK Certificate Fields

3.2.1 Version

This field describes the version of the X.509 certificate. Since EK certificates contain mandatory extensions the version number MUST be set to 3 (which is encoded as the value 2 in ASN.1).

3.2.2 Serial Number

The serial number MUST be a positive integer which is uniquely assigned to each EK certificate by the issuer. The combination of an issuer's DN and the serial number MUST uniquely describe a single certificate.

3.2.3 Signature Algorithm

This field identifies the algorithm used by the EK certificate issuer to sign the certificate. The certificate SHOULD be signed using an RSA 2048 bit key or ECC NISTP-256 bit key.

When using an RSA key the EK certificate SHOULD be signed using the algorithm sha256WithRSAEncryption which has the OID value defined in RFC 5754[12] as shown below. The AlgorithmIdentifier parameters field MUST be the ASN.1 type NULL.

```
sha256WithRSAEncryption OBJECT IDENTIFIER  ::=  {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 11 }
```

When using an ECC key the EK certificate SHOULD be signed using the algorithm ecdsa-with-SHA256 which has the OID value defined in RFC 5754[12] as shown below. The AlgorithmIdentifier parameters field MUST be the ASN.1 type NULL.

```
ecdsa-with-SHA256 OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
    us(840)ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 2 }
```

If an ECC curve different than the recommended NISTP-256 is used as the signing key, the security strength of the signing algorithm SHOULD be adapted to be equivalent to the security strength of the signing key (e.g. ecdsa-with-SHA384 for NISTP-384 or ecdsa-with-SHA512 for NISTP-521).

NOTE If a certificate issuer wishes to sign a certificate of one type with a key of another type (i.e. sign an RSA certificate with ECDSA or an ECC certificate with RSA), they should take care of existing patents in order to avoid licensing issues.

3.2.4 Issuer

This field contains the distinguished name of the certificate issuer which is the entity that vouches that the TPM is genuine and complies with the TPM 2.0 Library Specification[1].

3.2.5 Validity

The period when the certificate is valid is represented by two date values named notBefore and notAfter. Issuers SHOULD assign notBefore to the current time when the EK certificate is issued and notAfter to the last date that the certificate will be considered valid. Both notBefore and notAfter MUST use the appropriate time format as indicated by RFC 5280[11]. (See also section 2.2.3 EK Credential Lifetime)

3.2.6 Subject

The subject field MUST contain an X.500 distinguished name (DN) that uniquely identifies the TPM or, if unique identification through the subject field is not required, MUST be empty.

If the subject name field is empty, the subject alternative name extension MUST be critical in accordance with RFC 5280[11], otherwise it SHOULD be non-critical.

3.2.7 Subject Public Key Info

This describes the public Endorsement Key algorithm and key value. The public key SHOULD be an RSA 2048 bit key or ECC NISTP-256 bit key.

For an RSA public key the algorithm rsaEncryption which has the OID value defined in RFC 3279[13] as shown below MUST be used. The AlgorithmIdentifier parameters field MUST be the ASN.1 type NULL.

```
rsaEncryption OBJECT IDENTIFIER  ::=  {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }
```

The RSA public key MUST be encoded using the ASN.1 type RSAPublicKey as defined in RFC 3279[13].

```
RSAPublicKey ::= SEQUENCE {
    modulus            INTEGER,    -- n
    publicExponent     INTEGER  }  -- e
```

For an ECC public key the algorithm id-ecPublicKey which has the OID value defined in RFC 5480[14] as shown below MUST be used. The ECParameters field is required, the nameCurve field SHOULD contain the OID secp256r1 (NISTP-256 curve).

```
id-ecPublicKey OBJECT IDENTIFIER  ::=  {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) keyType(2) 1 }
```

```
ECParameters ::= CHOICE {
    namedCurve         OBJECT IDENTIFIER
    -- implicitCurve   NULL
    -- specifiedCurve  SpecifiedECDomain
    }

secp256r1 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) curves(3) prime(1) 7 }
```

If an ECC curve different than the recommended NISTP-256 is used as the subject public key, the appropriate OID defined for that ECC curve MUST be used.

The ECC public key MUST be encoded as an ECC Point. The uncompressed Format SHOULD be used.

```
ECPoint ::= OCTET STRING
```

3.2.8 Certificate Policies

This indicates the policy terms under which the certificate was issued. PolicyIdentifier MUST have at least one object identifier. Policy qualifiers are optional. The cPSuri policy qualifier MAY contain the value of an HTTP URL at which a plain language version of the issuer's certificate policy may be obtained. The userNotice policy qualifer MAY contain an explicitText. This extension SHOULD be non-critical.

3.2.9 Subject Alternative Name

This contains the alternative name of the entity associated with this certificate. The issuer MUST include TPM manufacturer, TPM part number and TPM firmware version, using the directoryName-form within the GeneralName structure. The ASN.1 encoding is specified in section 3.1.2 TPM Device Attributes. In accordance with RFC 5280[11], this extension MUST be critical if subject is empty and SHOULD be non-critical if subject is non-empty.

- The TPM manufacturer identifies the manufacturer of the TPM. This value MUST be the vendor ID defined in the TCG *Vendor ID Registry[3]* . It MUST match the value reported by the command TPM2_GetCapability(property = TPM_PT_MANUFACTURER).
- The TPM part number is encoded as a string and is manufacturer-specific. A manufacturer MUST provide a way to the user to retrieve the part number physically or logically. This information could be e.g. provided as part of the vendor string in the command TPM2_GetCapability(property = TPM_PT_VENDOR_STRING_x; x=1…4).
- The TPM firmware version is a manufacturer-specific implementation version of the TPM. This value SHOULD match the version reported by the command TPM2_GetCapability (property = TPM_PT_FIRMWARE_VERSION_1).

NOTE This representation of subject alternative name is maintained to provide consistency with TPM 1.2.

In addition, TPM 2.0 allows the inclusion of an optional attribute that contains the TPM serial number. The issuer MAY include HardwareModuleName as defined in RFC 4108[15] using the otherName-form within the GeneralName structure. HardwareModuleName is intended to facilitate establishment of Initial Device Identifier (IDevID) Credentials that are compliant to the IWG document *TPM Keys for Platform Identity[7]* (at time of writing, only available for TPM 1.2) which include the same HardwareModuleName attribute. This enables a platform manufacture to replicate the certified TPM serial number from the EK Credential to the IDevID Credential. Alternatively, the unique serial number could also be stored in the subject field.

- The TPM serial number is a manufacturer-specific unique chip identification number. If present, the manufacturer SHOULD implement an NV index that allows the user to retrieve the serial number from the TPM.

NOTE The TPM serial number usually includes detailed production parameters. Since that might be revealing information that the manufacturer doesn't want to disclose the hash of the TPM serial number could be used instead.

The HardwareModuleName attribute as defined in RFC 4108[15] is shown below. The hwType field MUST contain the TCG registered OID (2.23.133.1.2) that represents the hwType for TPM 2.0. The hwSerialNum field MUST contain the TPM serial number.

```
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 }
```

3.2.10 Basic Constraints

This indicates whether the subject is a CA. "CA" MUST be set to FALSE. This extension MUST be critical.

3.2.11 Subject Directory Attributes

The extension includes miscellaneous properties and security assertions about the entity. This extension MUST be non-critical.

The following attribute MUST be included in a subject directory attributes extension in the EK certificate:

- The "TPM Specification" attribute which identifies the family, level and revision of the TCG TPM specification to which the TPM was designed. The ASN.1 encoding is specified in section 3.1.3 TPM Specification Attributes.

The following attribute MAY be included in a subject directory attributes extension in the EK certificate:

- The "TPM Security Assertions" attribute which describes various assertions about the security properties of the TPM and the conditions under which the Endorsement Key was generated. The ASN.1 encoding is specified in section 3.1.1 TPM Security Assertions.

3.2.12 Authority Key Identifier

This identifies the subject public key of the certificate issuer and hence facilitates the validation of the certificate path. The certificate MUST contain an authority key identifier that matches the subject key identifier of the CA certificate. The issuer name and the serial number are optional. This extension MUST be non-critical.

3.2.13 Authority Information Access

This provides additional information about the issuer. Authority Information Access SHOULD contain the accessMethod OID id-ad-caIssuers and MAY additionally contain the OID id-ad-ocsp. This extension MUST be non-critical.

If id-ad-caIssuers appears as accessMethod, then the accessLocation value SHOULD point to the URL where the certificate of the issuing CA can be retrieved.

If id-ac-ocsp appears as accessMethod, then the accessLocation value SHOULD point to the access value of the OCSP responder (HTTP URI). The relying party can access the certificate status for this certificate by sending a properly formatted OCSPRequest to the URI. If both a CDP and OCSP AIA extension are present in the certificate, then the relying parties SHOULD use OCSP as the primary validation mechanism.

3.2.14 CRL Distribution

This extension is optional and provides the location of the subject's revocation information. The relying party can access the CRL for this certificate from this URI. If both a CDP and OCSP AIA extension are present in the certificate, then relying parties SHOULD use OCSP as the primary validation mechanism. This extension MUST be non-critical.

3.2.15 Key Usage

This extension indicates the intended purpose of the subject public key. This extension MUST be critical.

If the EK has the decrypt attribute set, the keyEncipherment bit MUST be set for an RSA EK certificate; the keyAgreement bit MUST be set for an ECC EK certificate.

If the EK has the sign attribute set, the digitalSignature bit MUST be set.

3.2.16 Extended Key Usage

This extension indicates the intended purpose of the subject public key. Extended key usage SHOULD contain the OID tcg-kp-EKCertificate defined in section 5 of this document as shown below. The OID is used to unambiguously identify the certificate as an EK certificate. This extension MUST be non-critical.

```
tcg-kp-EKCertificate OBJECT IDENTIFIER ::= {
    joint-iso-itu-t(2) international-organizations(23) tcg(133) kp(8) 1}
```

NOTE If the issuing CA is used exclusively to issue EK certificates, it is recommended to equally include the OID tcg-kp-EKCertificate in the issuing CA certificate. This ensures that the use of the CA is limited to that particular purpose. If the issuing CA issues certificates for multiple known purposes, then the set of relevant EKU OIDs could be included in the issuing CA certificate.

3.2.17 Subject Key Identifier

This identifies the public key of the certificate. This extension MAY be included in EK certificates.

4 Changes from Previous Versions

This chapter provides a summary of significant changes from the TCG Credential Profile specification V1.2[4].

- Former versions of the Credential Profile specification defined three different credential types. The Endorsement Key (EK) Credential, the Attestation Identity Key (AIK) Credential, and the Platform Endorsement (Platform) Credential. This document specifies only the Endorsement Key Credential for TPM 2.0.
- In TPM 1.2, the Endorsement Key was defined as a single, RSA 2048 bit key. Further, it was required to be a decryption key. TPM 2.0 can have more than one EK, and supports different asymmetric key types. An EK can be created as a decryption or signing key.
- Due to the increased number of possible EKs, TPM 2.0 allows EK Credentials to be stored in an external device. In addition to the handle for the EK certificate, there might be auxiliary handles for a nonce and a public area template (that indicate how the key associated to the certificate is generated).

Changes related to certificate fields:

- Signature algorithm: TPM 1.2 used sha-1WithRSAEncrytion. This is changed to sha256WithRSAEncryption for TPM 2.0 since the use of SHA-1 for the generation of digital signatures is considered as too weak (see [10]).
- Subject: In TPM 1.2, subject was required to be empty. In TPM 2.0, it can either contain a distinguished name or be empty. This change accommodates the request for a field that could contain a unique identifier.
- Subject public key info: TPM 1.2 used OID id-RSAES-OAEP where the octect string for the “pSourceFunc” parameter was “TCPA”. This is changed to OID rsaEncryption for TPM 2.0 since the uncommon OID used in TPM 1.2 could not be processed by most of the standard libraries.
- Certificate policies: In TPM 1.2, the CPS qualifier (URL point at plain text version of the certificate policy) and the UserNotice qualifier (text with “TCPA Trusted Platform Module Endorsement”) were mandatory. In TPM 2.0, both policy qualifiers are changed to be optional because the RFC 5280[11] recommends that the extension consist of only an OID to promote interoperability. The extension is changed to be non-critical for TPM 2.0.
- Subject alternative name: This extension was required to be critical in TPM 1.2. In TPM 2.0, it is either critical or non-critical dependent on the presence of the subject field which is in accordance to RFC 5280[11].
- Subject alternative name – TPM firmware version: The attribute is increased to contain four bytes to match the version number implemented in TPM 2.0.
- Subject alternative name – TPM serial number: TPM 2.0 allows the inclusion of the HardwareModuleName attribute which includes the TPM serial number.
- Subject directory attributes – supported algorithms: The attribute is removed for TPM 2.0. While TPM 1.2 was constrained to RSA and SHA-1, TPM 2.0 is able support many different algorithms. That would result in a too complex list and is therefore dropped.
- Authority information access: In TPM 1.2, accessMethod was only allowed to be id-ad-ocsp. This is changed to allow either id-ad-ocsp or id-ad-caIssuers for TPM 2.0. The URL provided with id-ad-caIssuers is useful to retrieve the issuing CA certificate.
- Key usage: Empty in TPM 1.2. Since in TPM 2.0 the subject public key info is changed to contain the generic OID rsaEncryption, the key usage extension is used to indicate the purpose of the key.
- Extended key usage: Empty in TPM 1.2. TPM 2.0 recommends using tcg-kp-EKCertificate. This OID unambiguously identifies the certificate as an EK certificate.
- The certificate field definitions were expanded to cover ECC.

5 X.509 ASN.1 Structures and OIDs

TCG has registered an object identifier (OID) namespace as an “international body” in the ISO registration hierarchy. This leads to shorter OIDs and gives TCG the ability to manage its own namespace. The OID namespace is inherited from TCPA. These definitions are intended to be used within the context of an X.509 v3 certificate specifically leveraging the profile described in RFC 5280[11].

```
-- TCG specific OIDs
tcg OBJECT IDENTIFIER ::= {
    joint-iso-itu-t(2) international-organizations(23) tcg(133) }

tcg-tcpaSpecVersion OBJECT IDENTIFIER ::= {tcg 1}
tcg-attribute OBJECT IDENTIFIER ::= {tcg 2}
tcg-protocol OBJECT IDENTIFIER ::= {tcg 3}
tcg-algorithm OBJECT IDENTIFIER ::= {tcg 4}
tcg-ce OBJECT IDENTIFIER ::= {tcg 6}
tcg-kp OBJECT IDENTIFIER ::= {tcg 8}

-- TCG Spec Version OIDs
tcg-sv-tpm12 OBJECT IDENTIFIER ::= { tcg-tcpaSpecVersion 1}
tcg-sv-tpm20 OBJECT IDENTIFIER ::= { tcg-tcpaSpecVersion 2}

-- TCG Attribute OIDs
tcg-at-tpmManufacturer OBJECT IDENTIFIER ::= {tcg-attribute 1}
tcg-at-tpmModel OBJECT IDENTIFIER ::= {tcg-attribute 2}
tcg-at-tpmVersion OBJECT IDENTIFIER ::= {tcg-attribute 3}
tcg-at-platformManufacturer OBJECT IDENTIFIER ::= {tcg-attribute 4}
tcg-at-platformModel OBJECT IDENTIFIER ::= {tcg-attribute 5}
tcg-at-platformVersion OBJECT IDENTIFIER ::= {tcg-attribute 6}

tcg-at-securityQualities OBJECT IDENTIFIER ::= {tcg-attribute 10}
tcg-at-tpmProtectionProfile OBJECT IDENTIFIER ::= {tcg-attribute 11}
tcg-at-tpmSecurityTarget OBJECT IDENTIFIER ::= {tcg-attribute 12}
tcg-at-tbbProtectionProfile OBJECT IDENTIFIER ::= {tcg-attribute 13}
tcg-at-tbbSecurityTarget OBJECT IDENTIFIER ::= {tcg-attribute 14}
tcg-at-tpmIdLabel OBJECT IDENTIFIER ::= {tcg-attribute 15}
tcg-at-tpmSpecification OBJECT IDENTIFIER ::= {tcg-attribute 16}
tcg-at-tcgPlatformSpecification OBJECT IDENTIFIER ::= {tcg-attribute 17}
tcg-at-tpmSecurityAssertions OBJECT IDENTIFIER ::= {tcg-attribute 18}
tcg-at-tbbSecurityAssertions OBJECT IDENTIFIER ::= {tcg-attribute 19}

-- TCG Algorithm OIDs
tcg-algorithm-null OBJECT IDENTIFIER ::= {tcg-algorithm 1}

-- TCG Key Purposes OIDs
tcg-kp-EKCertificate OBJECT IDENTIFIER ::= {tcg-kp 1}
tcg-kp-PlatformCertificate OBJECT IDENTIFIER ::= {tcg-kp 2}
tcg-kp-AIKCertificate OBJECT IDENTIFIER ::= {tcg-kp 3}

-- TCG Certificate Extensions
tcg-ce-relevantCredentials OBJECT IDENTIFIER ::= {tcg-ce 2}
tcg-ce-relevantManifests OBJECT IDENTIFIER ::= {tcg-ce 3}
tcg-ce-virtualPlatformAttestationService OBJECT IDENTIFIER ::= {tcg-ce 4}
tcg-ce-migrationControllerAttestationService OBJECT IDENTIFIER ::= (tcg-ce 5}
tcg-ce-migrationControllerRegistrationService OBJECT IDENTIFIER ::= (tcg-ce 6}
tcg-ce-virtualPlatformBackupService OBJECT IDENTIFIER ::= (tcg-ce 7}

-- TCG Protocol OIDs
tcg-prt-tpmIdProtocol OBJECT IDENTIFIER ::= {tcg-protocol 1}

-- tcg specification attributes for tpm and platform
```

```
tPMSpecification ATTRIBUTE ::= {
    WITH SYNTAX TPMSpecification
    ID tcg-at-tpmSpecification }

TPMSpecification ::= SEQUENCE {
    family UTF8String (SIZE (1..STRMAX)),
    level INTEGER,
    revision INTEGER }

tCGPlatformSpecification ATTRIBUTE ::= {
    WITH SYNTAX TCGPlatformSpecification
    ID tcg-at-tcgPlatformSpecification }

TCGSpecificationVersion ::= SEQUENCE {
    majorVersion INTEGER,
    minorVersion INTEGER,
    revision INTEGER }

TCGPlatformSpecification ::= SEQUENCE {
    Version TCGSpecificationVersion,
    platformClass OCTET STRING SIZE(4) }

-- tcpa tpm specification attribute (deprecated)

tCPASpecVersion ATTRIBUTE ::= {
    WITH SYNTAX TCPASpecVersion
    ID tcg-tcpaSpecVersion }

TCPASpecVersion ::= SEQUENCE {

    major INTEGER,

    minor INTEGER }


-- manufacturer implementation model and version attributes

TPMManufacturer ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-tpmManufacturer }

TPMModel ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-tpmModel }

TPMVersion ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-tpmVersion }

PlatformManufacturer ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-platformManufacturer }

PlatformModel ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-platformModel }

PlatformVersion ATTRIBUTE ::= {
    WITH SYNTAX UTF8String (SIZE (1..STRMAX))
    ID tcg-at-platformVersion }


-- tpm and platform tbb security assertions

Version ::= INTEGER { v1(0) }
```

```
tPMSecurityAssertions ATTRIBUTE ::= {
    WITH SYNTAX TPMSecurityAssertions
    ID tcg-at-tpmSecurityAssertions
}

TPMSecurityAssertions ::= SEQUENCE {
    version Version DEFAULT v1,
    fieldUpgradable BOOLEAN DEFAULT FALSE,
    ekGenerationType [0] IMPLICIT EKGenerationType OPTIONAL,
    ekGenerationLocation [1] IMPLICIT EKGenerationLocation OPTIONAL,
    ekCertificateGenerationLocation [2] IMPLICIT
        EKCertificateGenerationLocation OPTIONAL,
    ccInfo [3] IMPLICIT CommonCriteriaMeasures OPTIONAL,
    fipsLevel [4] IMPLICIT FIPSLevel OPTIONAL,
    iso9000Certified [5] IMPLICIT BOOLEAN DEFAULT FALSE,
    iso9000Uri IA5STRING (SIZE (1..URIMAX)) OPTIONAL }

tBBSecurityAssertions ATTRIBUTE ::= {
    WITH SYNTAX TBBSecurityAssertions
    ID tcg-at-tbbSecurityAssertions }

TBBSecurityAssertions ::= SEQUENCE {
    version Version DEFAULT v1,
    ccInfo [0] IMPLICIT CommonCriteriaMeasures OPTIONAL,
    fipsLevel [1] IMPLICIT FIPSLevel OPTIONAL,
    rtmType [2] IMPLICIT MeasurementRootType OPTIONAL,
    iso9000Certified BOOLEAN DEFAULT FALSE,
    iso9000Uri IA5STRING (SIZE (1..URIMAX)) OPTIONAL }

EKGenerationType ::= ENUMERATED {
    internal (0),
    injected (1),
    internalRevocable(2),
    injectedRevocable(3) }

EKGenerationLocation ::= ENUMERATED {
    tpmManufacturer (0),
    platformManufacturer (1),
    ekCertSigner (2) }

EKCertificateGenerationLocation ::= ENUMERATED {
    tpmManufacturer (0),
    platformManufacturer (1),
    ekCertSigner (2) }

-- V1.1 of this specification adds hybrid and physical.
-- Hybrid means the measurement root is capable of static AND dynamic
-- Physical means that the root is anchored by a physical TPM
-- Virtual means the TPM is virtualized (possibly running in a VMM)

-- TPMs or RTMs might leverage other lower layer RTMs to virtualize the
-- the capabilities of the platform.
MeasurementRootType ::= ENUMERATED {

    static (0),

    dynamic (1),

    nonHost (2),

    hybrid (3),

    physical (4),

    virtual (5) }
```

```
        -- common criteria evaluation

        CommonCriteriaMeasures ::= SEQUENCE {
            version IA5STRING (SIZE (1..STRMAX)), -- "2.2" or "3.1"; future syntax defined
by CC
            assurancelevel EvaluationAssuranceLevel,
            evaluationStatus EvalutionStatus,
            plus BOOLEAN DEFAULT FALSE,
            strengthOfFunction [0] IMPLICIT StrengthOfFunction OPTIONAL,
            profileOid [1] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
            profileUri [2] IMPLICIT URIReference OPTIONAL,
            targetOid [3] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
            targetUri [4] IMPLICIT URIReference OPTIONAL }


        EvaluationAssuranceLevel ::= ENUMERATED {
            level1 (1),
            level2 (2),
            level3 (3),
            level4 (4),
            level5 (5),
            level6 (6),
            level7 (7) }

        StrengthOfFunction ::= ENUMERATED {
            basic (0),
            medium (1),
            high (2) }

        URIReference ::= SEQUENCE {
            uniformResourceIdentifier IA5String (SIZE (1..URIMAX)),
            hashAlgorithm AlgorithmIdentifier OPTIONAL,
            hashValue BIT STRING OPTIONAL }


        EvaluationStatus ::= ENUMERATED {
            designedToMeet (0),
            evaluationInProgress (1),
            evaluationCompleted (2) }


        -- fips evaluation

        FIPSLevel ::= SEQUENCE {
            version IA5STRING (SIZE (1..STRMAX)), -- "140-1" or "140-2"
            level SecurityLevel,
            plus BOOLEAN DEFAULT FALSE }


        SecurityLevel ::= ENUMERATED {
            level1 (1),
            level2 (2),
            level3 (3),
            level4 (4) }


        -- aik certificate label from tpm owner


        TPMIdLabel OTHER-NAME ::= {UTF8String IDENTIFIED BY {tcg-at-tpmIdLabel} }


        -- the following are deprecated but may be present for compatibility with TCPA

        TPMProtectionProfile ATTRIBUTE ::= {
            WITH SYNTAX ProtectionProfile
            ID tcg-at-tpmProtectionProfile }


        TPMSecurityTarget ATTRIBUTE ::= {
            WITH SYNTAX SecurityTarget
            ID tcg-at-tpmSecurityTarget }
```

```
TBBProtectionProfile ATTRIBUTE ::= {
    WITH SYNTAX ProtectionProfile
    ID tcg-at-tbbProtectionProfile }


TBBSecurityTarget ATTRIBUTE ::= {
    WITH SYNTAX SecurityTarget
    ID tcg-at-tbbSecurityTarget }


ProtectionProfile ::= OBJECT IDENTIFIER

SecurityTarget ::= OBJECT IDENTIFIER


-- V1.1 addition for enabling references to other credentials or
-- XML-based Reference Manifests.  These data objects are included
-- in X.509 extensions using the new tcg-ce-[relevantCredentials,
-- relevantManifests] OIDs.

HashAlgAndValue ::= SEQUENCE {
   hashAlg         AlgorithmIdentifier,
   hashValue       OCTET STRING }


HashedSubjectInfoURI  ::=  SEQUENCE {
   documentURI IA5String (SIZE (1..URIMAX)),
   documentAccessInfo OBJECT IDENTIFIER OPTIONAL,
   documentHashInfo HashAlgAndValue OPTIONAL }

SubjectInfoURIList  ::=
   SEQUENCE SIZE (1..REFMAX) OF HashedSubjectInfoURI


TCGRelevantCredentials::=

   SEQUENCE SIZE (1..REFMAX) OF HashedSubjectInfoURI

TCGRelevantManifests::=

   SEQUENCE SIZE (1..REFMAX) OF HashedSubjectInfoURI


-- V1.2 addition of virtualization oriented credential extensions.  This extension
indicates how a remote challenger can contact the (deep) attestation service below
the current credential holder in order to attest the layer below.  Using this model
allows the credential of each virtualization layer to reference the attestation
service for the layer below it.    A remote challenger could traverse the layer
hierarchy using this extension until reaching the physical trusted platform rooted
attestation.    The following URI is optionally included in a certificate for a
virtual machine associated with the tcg-ce-virtualPlatformAttestationService
extension OID.          These URI are associated with the tcg-ce-
[virtualPlatformAttestationService,          migrationControllerAttestationService,
migrationControllerRegistrationService,     virtualPlatformBackupService]     OIDs
respectively:

VirtualPlatformAttestationServiceURI ::= IA5String (SIZE (1..URIMAX)

MigrationControllerAttestationServiceURI ::= IA5String (SIZE (1..URIMAX)

MigrationControllerRegistrationServiceURI ::= IA5String (SIZE (1..URIMAX)

VirtualPlatformBackupServiceURI ::= SEQUENCE {
   restoreAllowed BOOLEAN DEFAULT FALSE,
   backupServiceURI    IA5String }
```

6 References

For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] TPM 2.0 Library Specification:
http://www.trustedcomputinggroup.org/resources/tpm_library_specification

[2] Registry of Reserved TPM 2.0 Handles and Localities:
http://www.trustedcomputinggroup.org/resources/registry

[3] Vendor ID Registry:
http://www.trustedcomputinggroup.org/resources/vendor_id_registry

[4] TCG Credential Profile Version 1.0, Version 1.1 and Version 1.2:
http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_tcg_credential_profiles_specification

[5] PC Client Specific Platform TPM Profile for TPM 2.0:
http://www.trustedcomputinggroup.org/developers/pc_client

[6] CMC Profile for EK/Platform Certificate Enrollment for TPM 1.2:
http://www.trustedcomputinggroup.org/developers/infrastructure

[7] TPM Keys for Platform Identity for TPM 1.2:
http://www.trustedcomputinggroup.org/developers/infrastructure

[8] TPM 1.2 Main Specification:
http://www.trustedcomputinggroup.org/resources/tpm_main_specification

[9] IEEE Standard for Local and metropolitan area networks, Secure Device Identity, 2009

[10] NIST Special Publication 800-57, Recommendation for Key Management – Part 1: General

[11] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, http://www.ietf.org/rfc/rfc5280.txt

[12] Using SHA2 Algorithms with Cryptographic Message Syntax, RFC 5754,
http://www.ietf.org/rfc/rfc5754.txt

[13] Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 3279, http://www.ietf.org/rfc/rfc3279.txt

[14] Elliptic Curve Cryptography Subject Public Key Information, RFC 5480,
http://www.ietf.org/rfc/rfc5480.txt

[15] Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages, RFC 4108,
http://www.ietf.org/rfc/rfc4108.txt

[16] Key words for use in RFCs to Indicate Requirement Levels, RFC 2119,
http://www.ietf.org/rfc/rfc2119.txt

[17] Hypertext Markup Language – 2.0, RFC 1866, http://www.ietf.org/rfc/rfc1866.txt

[18] ITU-T X.690: Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

A. Certificate Examples

A.1 Example 1 (user device TPM, e.g. PC-Client)

The following section provides an example for a standard, user device TPM (e.g. PC-Client) Endorsement Key certificate. The ASN.1 encoding for the subject alternative name and subject directory attributes extension is provided below. The values used in this example are for illustrative purpose and must be replaced with manufacturer-specific data.

Subject alternative name:
TPMManufacturer = id:54534700 (TCG)
TPMModel = ABCDEF123456 (part number)
TPMVersion = id:00010023 (firmware version)

```
// SEQUENCE
30 49
    // SET
    31 16
        // SEQUENCE
        30 14
            // OBJECT IDENTIFER tcg-at-tpmManufacturer (2.23.133.2.1)
            06 05 67 81 05 02 01
            // UTF8 STRING id:54434700 (TCG)
            0C 0B 69 64 3A 35 34 34 33 34 37 30 30
    // SET
    31 17
        // SEQUENCE
        30 15
            // OBJECT IDENTIFER tcg-at-tpmModel (2.23.133.2.2)
            06 05 67 81 05 02 02
            // UTF8 STRING ABCDEF123456
            0C 0C 41 42 43 44 45 46 31 32 33 34 35 36
    // SET
    31 16
        // SEQUENCE
        30 14
            // OBJECT IDENTIFER tcg-at-tpmVersion (2.23.133.2.3)
            06 05 67 81 05 02 03
            // UTF8 STRING id:00010023
            0C 0B 69 64 3A 30 30 30 31 30 30 32 33
```

Subject directory attributes:
TPMSpecification
Family = id:322E3000 (2.0)
Level = 0
Revision = 99
TPMSecurityAssertions (not included here since optional)

```
// SEQUENCE
30 1E
    // OBJECT IDENTIFIER tcg-at-tpmSpecification (2.23.133.2.16)
    06 05 67 81 05 02 10
        // SET
        31 15
            // SEQUENCE
            30 13
                // UTF8 STRING (2.0)
                0C 03 32 2E 30
                // INTEGER (0)
                02 01 00
                // INTEGER (99)
                02 01 63
```

The encoding of the extensions above is extracted from the following example certificate. The example certificate provided below is for illustrative propose only, all example values must be replaced with manufacturer-specific data. For simplicity some optional configurations (e.g. optional data within an extension) are omitted. The manufacturer's certificate is not required to look exactly the same as the example certificate. For better testing the certificate is provided in PEM format. When read from the TPM the certificate is encoded in DER[18].

```
-----BEGIN CERTIFICATE-----
MIID7zCCAtegAwIBAgIBATANBgkqhkiG9w0BAQsFADAUMRIwEAYDVQQDDAlFeGFt
cGxlQ0EwHhcNMTQwMTE1MTU0MDUwWhcNMTUwMTE1MTU0MDUwWjAAMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAncvm0aOBK05rdNInYXzJGV5SFteVUFpt
XFxg4evROvlulB3BzUmFGQYFDcItVnJX2fAvf0UJLtLBVBQggb5ylL6bRpj72cS3
oyNbs0CGmix9Z1QDjkZZFvIsD1GcKO0tvsCvsEItH8Cm0fq8WcGFijWLdRD5eulP
55pq1bAHAvIo4+VLMJVBG71xrKGZeHPjKoq6seYjh7AGy+hk2vmFzpZ8Ghdgqv+K
02IZ7FEdzuylHW8U3qsxBHysMut4inj6AiVf467OOs5meHiifIK9MGkovMrfY9iX
uUVUs/KXpE1sgeoX9BLvx1BPcODosr5K+z5i71OtIXy4CXrPvcGzRwIDAQABo4IB
XjCCAVowQAYIKwYBBQUHAQEENDAyMDAGCCsGAQUFBzAChiRodHRwOi8vd3d3LmV4
YW1wbGUuY29tL0V4YW1wbGVDQS5jcnQwDgYDVR0PAQH/BAQDAgAgMFkGA1UdEQEB
/wRPME2kSzBJMRYwFAYFZ4EFAgEMC2lkOjU0NDM0NzAwMRcwFQYFZ4EFAgIMDEFC
Q0RFRjEyMzQ1NjEWMBQGBWeBBQIDDAtpZDowMDAxMDAyMzAMBgNVHRMBAf8EAjAA
MDUGA1UdHwQuMCwwKqAooCaGJGh0dHA6Ly93d3cuZXhhbXBsZS5jb20vRXhhbXBs
ZUNBLmNybDAQBgNVHSAECTAHMAUGAyoDBDAfBgNVHSMEGDAWgBQ0d2ckTESv554q
4LJMaVeVJLM92jAQBgNVHSUECTAHBgVngQUIATAhBgNVHQkEGjAYMBYGBWeBBQIQ
MQ0wCwwDMi4wAgEAAgFjMA0GCSqGSIb3DQEBCwUAA4IBAQAba2btJ/+4z02MWpNp
99AFGpEu3yIaJqI6NeHvC6fxxe/9lWlHKISR+CnpAh03/MKT8TP2/cUSi0jjkQNh
MtueUNofE79fYXtHXHU7wzzUFWNwCmhTuHDYl3jmD0fJ9yA2CuUHT6q3UV+PwXN+
EHE1hQwC8QtNC/5A7wY1e5dBLdgwSSIgTc4lSsbNcZ9d+m7mWEWpumSYU0czTDEN
Hmdu/VJuDN/RCOAyBb+hc19LAucGmnFYOhxWHfd9zbXZA1ldFUxrpPuVfKx+Eo8f
rMsB2oZKMwSYUAWotqolhLe2wdBMRjdmVz44kIhuFB7y4BpQjlB1+xAzX9Hb31CG
eoS2
-----END CERTIFICATE-----
```

A.2 Example 2 (non-user device TPM, e.g. DevID)

This second example certificate additionally includes the optional HardwareModuleName attribute in the subject alternative name extension which includes the TPM serial number. The particular use case for such an EK certificate is for non-user devices (e.g. switches, routers, and wireless access points) that require an Device Identifier (DevID) Credential conforming to the IWG *TPM Keys for Platform Identity[7]* specification. The values used in this example are for illustrative purpose and must be replaced with manufacturer-specific data.

Subject alternative name:
HW type = TPM 2.0
TPM serial number = the string “tpmserialnumber”

```
// OBJECT IDENTIFIER id-on-hardwareModuleName (1.3.6.1.5.5.7.8.4)
06 08 2B 06 01 05 05 07 08 04
    // CONTEXT SPECIFIC (0)
    A0 1A
        // SEQUENCE
        30 18
            // OBJECT IDENTIFIER TPM2.0 (2.23.133.1.2)
            06 05 67 81 05 01 00
            // OCTECT STRING tpmserialnumber
            04 0F 74 70 6D 73 65 72 69 61 6C 6E 75 6D 62 65 72
```

The encoding of the extension above is extracted from the following example certificate. Same as for the example certificate provided in A.1, it is provided for illustrative propose only, all example values must be replaced with manufacturer-specific data. For simplicity some optional configurations (e.g. optional data within an extension) are omitted. The manufacturer’s certificate is not required to look exactly the same as the example certificate. For better testing the certificate is provided in PEM format. When read from the TPM the certificate is encoded in DER[18].

```
-----BEGIN CERTIFICATE-----
MIIEGDCCAwCgAwIBAgIBATANBgkqhkiG9w0BAQsFADAUMRIwEAYDVQQDDAlFeGFt
cGxlQ0EwHhcNMTQwMTE1MTU0MDUwWhcNMTUwMTE1MTU0MDUwWjAAMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAncvm0aOBK05rdNInYXzJGV5SFteVUFpt
XFxg4evROvlulB3BzUmFGQYFDcItVnJX2fAvf0UJLtLBVBQggb5ylL6bRpj72cS3
oyNbs0CGmix9Z1QDjkZZFvIsD1GcKO0tvsCvsEItH8Cm0fq8WcGFijWLdRD5eulP
55pq1bAHAvIo4+VLMJVBG71xrKGZeHPjKoq6seYjh7AGy+hk2vmFzpZ8Ghdgqv+K
02IZ7FEdzuylHW8U3qsxBHysMut4inj6AiVf467OOs5meHiifIK9MGkovMrfY9iX
uUVUs/KXpE1sgeoX9BLvx1BPcODosr5K+z5i71OtIXy4CXrPvcGzRwIDAQABo4IB
hzCCAYMwQAYIKwYBBQUHAQEENDAyMDAGCCsGAQUFBzAChiRodHRwOi8vd3d3LmV4
YW1wbGUuY29tL0V4YW1wbGVDQS5jcnQwDgYDVR0PAQH/BAQDAgAgMIGBBgNVHREB
Af8EdzB1pEswSTEWMBQGBWeBBQIBDAtpZDo1NDQzNDcwMDEXMBUGBWeBBQICDAxB
QkNERUYxMjM0NTYxFjAUBgVngQUCAwwLaWQ6MDAwMTAwMjOgJgYIKwYBBQUHCASg
GjAYBgVngQUBAgQPdHBtc2VyaWFsbnVtYmVyMAwGA1UdEwEB/wQCMAAwNQYDVR0f
BC4wLDAqoCigJoYkaHR0cDovL3d3dy5leGFtcGxlLmNvbS9FeGFtcGxlQ0EuY3Js
MBAGA1UdIAQJMAcwBQYDKgMEMB8GA1UdIwQYMBaAFDR3ZyRMRK/nnirgskxpV5Uk
sz3aMBAGA1UdJQQJMAcGBWeBBQgBMCEGA1UdCQQaMBgwFgYFZ4EFAhAxDTALDAMy
LjACAQACAWMwDQYJKoZIhvcNAQELBQADggEBABtrZu0n/7jPTYxak2n30AUakS7f
Ihomojo14e8Lp/HF7/2VaUcohJH4KekCHTf8wpPxM/b9xRKLSOORA2Ey255Q2h8T
v19he0dcdTvDPNQVY3AKaFO4cNiXeOYPR8n3IDYK5QdPqrdRX4/Bc34QcTWFDALx
C00L/kDvBjV7l0Et2DBJIiBNziVKxs1xn136buZYRam6ZJhTRzNMMQ0eZ279Um4M
39EI4DIFv6FzX0sC5waacVg6HFYd933NtdkDWV0VTGuk+5V8rH4Sjx+sywHahkoz
BJhQBai2qiWEt7bB0ExGN2ZXPjiQiG4UHvLgGlCOUHX7EDNf0dvfUIZ6hLY=
-----END CERTIFICATE-----
```