TRUSTED® COMPUTING GROUP

SPECIFICATION

DICE Certificate Profiles

Version 1.2 RC1
July 23, 2026

Contact: admin@trustedcomputinggroup.org

PUBLIC REVIEW

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice.

Readers should not design products based on this document.

DISCLAIMERS, NOTICES, AND LICENSE TERMS

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

Contact the Trusted Computing Group at admin@trustedcomputinggroup.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

Contents

1 SCOPE

1.1 Key Words

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

1.2 Statement Type

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

EXAMPLE: Start of informative comment

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

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

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

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

End of informative comment

2 REFERENCES

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

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

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

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

[5] Trusted Computing Group, "DICE Layering Architecture," 2020. [Online]. Available: https://www.trustedcomputinggroup.org.

[6] Internet Engineering Task Force, RFC5280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5280.

[7] Internet Engineering Task Force, RFC9052, "CBOR Object Signing and Encryption (COSE): Structures and Process" August 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9052.

[8] Internet Engineering Task Force, RFC9360, "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", February 2023. Available online: https://www.rfc-editor.org/rfc/rfc9360/.

[9] Internet Engineering Task Force, RFC7515, "JSON Web Signature (JWS)", May 2015. Available online: https://www.rfc-editor.org/rfc/rfc7515/.

[10] IANA Registry, "IANA CoRE Parameters – Content-Formats", Available online: https://www.iana.org/assignments/core-parameters/core-parameters.xhtml - content-formats/.

3 TERMS AND DEFINITIONS

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

3.1 Trusted Computing Terms

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

- Trusted Computing Group Glossary [1]
- NIST Computer Security Resource Center Glossary [2].

3.2 Glossary

TERM	DEFINITION
Digest	The result of a cryptographic hash operation.
Device	A highly integrated platform containing a programmable component with other optional programmable components and peripherals.
DevID, IDevID, LDevID	These terms are defined by the IEEE 802.1AR [3] standard as information that an entity (a person or device) possesses that allow it to make a verifiable claim of identity, i.e., to be authenticated.
Measurement	A digest of code and/or configuration data. It is implementation-specific, and out of scope for this document, whether a measurement is over a region of memory, a firmware or software image, or some combination thereof.

3.3 DICE Architecture Terminology Conventions

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

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

3.4 Abbreviations

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

ABBREVIATION	DESCRIPTION
CA	Certificate Authority
CDI	Compound Device Identifier
DICE	Device Identifier Composition Engine
ECA	Embedded Certificate Authority
OID	Object Identifier
PCR	Platform Configuration Register
RoT	Root-of-Trust
TCB	Trusted Computing Base
TCI	TCB Component Identity
UDS	Unique Device Secret

4 INTRODUCTION

This document assumes the reader is familiar with attestation architecture (see [1], [2]) and the TCG DICE Hardware Requirements for a Device Identifier Composition Engine [4], i.e., DICE. The DICE specification describes the composition of an identifier that represents the combination of hardware and software that begins a device boot sequence.

The DICE hardware requirements specification describes the construction of the Compound Device Identifier (CDI). The DICE Layering Architecture specification [5] describes how this construction also applies to transitions between components of a device's Trusted Computing Base (TCB). A device's TCB consists of all security relevant components that have been loaded at a given point in the boot sequence. A TCB component comprises hardware, firmware, software and/or configuration. A measurement of a TCB component is a TCB Component Identifier (TCI). An example of a TCI value is a digest of component firmware. In some cases where a component does not consist of measurable firmware or software, a hardware product identifier (or equivalent) MAY be used.

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

TCB components also use their DICE Compound Device Identifier (CDI) as the input to a key generation function. For example, this could be an asymmetric key generation function producing a key pair that is enrolled as an 802.11AR device identity credential, known as IDevID or LDevID [3]. Keys derived from CDI values could be enrolled with an application-specific Certificate Authority and used to perform attestation, authentication, and certification.

This specification defines conventions and profiles for Embedded Certificate Authority (ECA) certificates and attestation certificates, including IDevID and LDevID credentials, when used in a layered architecture.

5 DICE ARCHITECTURE CONVENTIONS FOR X.509 CERTIFICATES

5.1 DICE Certificate Extensions

The following sections describe conventions for certificates associated with a DICE-derived key pair. The certificate format is presumed to be X.509.v3 [6] and IEEE802.1AR-2018 [3] compatible.

5.1.1 Serial Number Generation

Certificate Serial Numbers distinguish different certificates from one another. As a result, the Serial Number field MUST be unique per CA for each Alias Key certificate. Different embedded CAs MAY issue certificates with the same Serial Number fields.

Start of informative comment

The ASN.1 encoding requires a serial number to be a positive integer.

End of informative comment

5.1.2 Certificate Lifetime

Devices with a secure local clock can use the clock to set the validity period of certificates. Conventionally, devices without a secure clock use generalized time values in the distant future to ensure the validity of the certificates they create. If a secure clock is unavailable, then devices can set the notAfter portion of the certificate's Validity Period to the X509-defined GeneralizedTime value of 99991231235959Z. This value is used to indicate that the certificate has no defined expiration date. Devices can set the notBefore portion to a known date and time in the recent past (e.g., TCB build time).

In practice, devices implementing this specification will either recertify keys on each boot (because the device is also recreating keypairs on each boot) or keys MAY be persisted if device firmware remains unchanged. In either scenario, it is assumed that rotation of certificates coincides with update of the device firmware.

5.1.3 Subject and Subject Alternative Name

Subject names SHOULD identify the component environment where the private key resides. Subject MAY contain representations of the device serial number, firmware identifiers, or DICE layering coordinates.

The attestation Verifier is recommended to not obtain attestation claims from Subject or SubjectAltName fields.

When a DICE layer functions as an ECA, the root ECA Subject name MUST be device-unique.

Subject names MUST be formatted for use in byte array comparisons. For example, parsing and comparing sub-fields might be possible but not mandatory.

Refer to RFC5280 [6] for rules and guidance on the interaction of the Subject (Section 4.1.2.6 [6]) and SubjectAltName (Section 4.2.1.6 [6]) fields when both are present in a certificate.

5.1.4 Issuer Name

The use of Issuer Name fields in all certificate types at layer n+1 MUST match the Subject Name of the issuing certificate at layer n. Issuer MAY contain representations of the device serial number, firmware identifiers, or DICE layering coordinates. Refer to RFC5280 for rules and guidance on the interaction of the Issuer and IssuerAltName fields when both are present in a certificate.

The attestation Verifier is recommended to not obtain attestation claims from Issuer or IssuerAltName fields.

5.1.5 Policy OIDs for Key Usage

The certificate can include the following OIDs. They are used by certificate issuers in Extended Key Usage to identify the purpose of the TCB layer. Certificate verification processes use the OIDs to correctly validate the usage or to improve verification processing efficiency.

The certificate profiles section (section 5.1.6) describes expected OID usage.

The following extensions use an OIDs arc from the TCG namespace.

```
TCG Branch:   tcg OBJECT IDENTIFIER ::= {2 23 tcg(133)}
DICE Branch:  tcg-dice OBJECT IDENTIFIER ::= { tcg platformClass(5) dice(4) }
              tcg-dice-kp OBJECT IDENTIFIER ::= { tcg-dice kp(100) }
```

5.1.5.1 Initial Identity OID

The identityInit policy OID identifies a key that can be used to authenticate an initial identity (e.g., IDevID) for a device, DICE layer or DICE component:

```
tcg-dice-kp-identityInit OBJECT IDENTIFIER ::= {tcg-dice-kp identity-init(6)}
```

The key MUST be bound to the device during manufacturing.

The key MUST be certified by the manufacturer. If the key is certified using an embedded CA:

- The embedded CA key MUST be certified by the manufacturer.
- The embedded CA key MUST be bound to the device during manufacturing.

No policyQualifier is defined for this OID.

5.1.5.2 Local Identity OID

The identityLoc policy OID identifies a key that can be used to authenticate a local identity (e.g., LDevID) for a device, DICE layer or DICE component:

```
tcg-dice-kp-identityLoc OBJECT IDENTIFIER ::= {tcg-dice-kp identity-loc(7)}
```

The key SHOULD be bound to the device post manufacturing.

The key MUST be certified by a local issuer.

If the key is certified using an embedded CA, the embedded CA MUST be certified by the local issuer.

No policyQualifier is defined for this OID.

5.1.5.3 Initial Attestation OID

The attestInit policy OID identifies an attestation key that can be used to attest (sign) evidence that describes a device (e.g., PCR, SW hash, product name), DICE layer or DICE component:

```
tcg-dice-kp-attestInit OBJECT IDENTIFIER ::= {tcg-dice-kp attest-init(8)}
```

The key MUST be bound to the device during manufacturing.

The key MUST be certified by the manufacturer. If the key is certified using an embedded CA:

- The embedded CA key MUST be certified by the manufacturer.
- The embedded CA key MUST be bound to the device during manufacturing.

No policyQualifier is defined for this OID.

Start of Informative Comment

A manufacturer may certify an Initial Attestation Key using an embedded CA that is itself certified by the manufacturer and bound to the device during manufacturing. In this case, the embedded CA is an intermediate CA in Initial Attestation Key certificate trust chain.

End of Informative Comment

5.1.5.4 Local Attestation OID

The attestLoc policy OID identifies an attestation key that can be used to attest (sign) evidence that describes a device (e.g., PCR, SW hash, product name), DICE layer or DICE component:

```
tcg-dice-kp-attestLoc OBJECT IDENTIFIER ::= {tcg-dice-kp attest-loc(9)}
```

The key SHOULD be bound to the device post manufacturing.

The key MUST be certified by a local issuer.

If the key is certified using an embedded CA, the embedded CA MUST be certified by the local issuer.

No policyQualifier is defined for this OID.

5.1.5.5 Initial Assertion OID

The assertInit policy OID identifies an attestation key that can be used to assert (sign) reference measurements about the device, DICE layer or DICE component:

```
tcg-dice-kp-assertInit OBJECT IDENTIFIER ::= {tcg-dice-kp assert-init(10)}
```

The key MUST be bound to the device during manufacturing.

The key MUST be certified by the manufacturer.

No policyQualifier is defined for this OID.

5.1.5.6 Local Assertion OID

The assertLoc policy OID identifies an attestation key that can be used to assert (sign) reference measurements about the device, DICE layer or DICE component:

```
tcg-dice-kp-assertLoc OBJECT IDENTIFIER ::= {tcg-dice-kp assert-loc(11)}
```

The key SHOULD be bound to the device post manufacturing.

The key MUST be certified by a local issuer.

No policyQualifier is defined for this OID.

5.1.5.7 Embedded Certificate Authority OID

The ECA policy OID identifies an embedded CA that can be used to authorize certificate issuance for keys residing on the current device, DICE layer or DICE component:

```
tcg-dice-kp-eca OBJECT IDENTIFIER ::= {tcg-dice-kp eca(12)}
```

The Issuer signing key MUST be certified by the manufacturer or local issuer.

No policyQualifier is defined for this OID.

5.1.6 Certificate Profiles

DICE layers can have specialized functionality and behavior. Certificate profiles allow the creation of certificates that are specialized for the expected TCB use. A TCB could combine functionality and behavior that is specific to multiple certificate profiles.

In certificates where the subject key was generated using a CDI value, the corresponding TCI value MUST be included in attestation evidence (e.g., as a certificate extension) if present.

5.1.6.1 Initial Device Identifier (IDevID) Certificates

Device manufacturers, OEMs, or other entities in a supply chain could issue IEEE802.1AR IDevID certificates using an external CA. If so, the constraints of Table 1 SHALL apply.

FIELD NAME	CONTENTS
Issuer	MUST identify or chain to the device manufacturer / supply chain entity that issues the certificate. If the Issuer is an embedded CA, then the ECA issuer MUST chain to the manufacturer CA.
Subject	MUST identify the TCB owning the IDevID private key. The Subject name MAY be a class identifier, implying there couldbe other device instances sharing the same name.
Subject Public Key Info	Contains the public key and algorithm identifier that is protected by an immutable TCB layer or a TCB layer that SHALL be modifiable only by the Issuer (as per [4]).
Key Usage	If Subject is an ECA then this field MUST contain keyCertSign and MUST NOT contain cRLSign. Otherwise, this field MUST NOT contain keyCertSign.
Extended Key Usage	This extension MUST contain id-tcg-dice-kp-identityInit, and MAY contain any additional appropriate values for the usage model, e.g., id-kp-clientAuth, id-tcg-dice-kp-eca, or id-tcg-dice-kp-attestInit.
Basic Constraints	If Subject is an ECA then this field MUST contain cA:TRUE and pathLengthConstraint as appropriate. Otherwise, the certificate SHOULD NOT contain BasicConstraints.
Attestation Extensions	A future TCG specification will address attestation extensions.

Table 1: IDevID certificate profile

5.1.6.2 Local Device ID (LDevID) Certificates

The device owner could issue an IEEE802.1AR LDevID certificate using an external CA. If so, the constraints in Table 2 SHALL apply.

FIELD NAME	CONTENTS
Issuer	MUST identify or chain to the owner CA. If the Issuer is an embedded CA, then the ECA issuer MUST chain to the owner CA.
Subject	MUST identify the TCB owning the LDevID private key. The Subject name MAY be a class identifier, implying there could be other device instances sharing the same name.
Subject Public Key Info	Contains the public key and algorithm identifier that is protected by an immutable TCB layer or a TCB layer that SHALL be modifiable only by the Issuer.
Key Usage	If Subject is an ECA then this field MUST contain keyCertSign and MUST NOT contain cRLSign. Otherwise, this field MUST NOT contain keyCertSign.

Extended Key Usage	This extension MUST contain tcg-dice-kp-identityLoc, and MAY contain any additional appropriate values for the usage model, e.g., tcg-dice-kp-eca, tcg-dice-kp-attestLoc, and/or id-kp-clientAuth for clients.
Basic Constraints	If Subject is an ECA then this field MUST contain cA:TRUE and pathLengthConstraint as appropriate. Otherwise, the certificate SHOULD NOT contain BasicConstraints.
Assertions Extensions	A future TCG specification will address attestation extensions.

Table 2: LDevID certificate profile

5.1.6.3 ECA Certificates

ECA certificates can be issued by an external CA or by an embedded CA (e.g. the Subject is an Embedded Sub-CA). If so, the constraints in Table 3 SHALL apply.

FIELD NAME	CONTENTS
Issuer	MUST identify the CA or embedded CA that issues the certificate. The Issuer MUST ensure that the private portion of the Subject Public Key is protected by a TCB. If Issuer is an embedded CA, then the Issuer MUST identify the TCB instance that issues this certificate.
Subject	MUST identify the TCB containing ECA functionality.
Subject Public Key Info	MUST contain the current TCB Layer ECA public key and algorithm identifier.
Key Usage	MUST contain keyCertSign. MUST NOT contain cRLSign, MAY contain other KeyUsage attributes as appropriate.
Extended Key Usage	This extension MUST contain id-tcg-dice-kp-eca, but MAY contain id-tcg-dice-kp-attestInit, id-tcg-dice-kp-attestLoc, id-tcg-dice-kp-identityInit, or id-tcg-dice-kp-identityLoc
Basic Constraints	MUST contain cA:TRUE and pathLengthConstraint as appropriate
Attestation Extensions	A future TCG specification will address attestation extensions.
CRLDistributionPoints Extension	SHOULD be present.

Table 3: ECA certificate profile

5.1.6.4 Attestation Certificates

Attestation certificates can be issued either by an ECA or an external CA. The constraints in Table 4 SHALL apply.

FIELD NAME	CONTENTS
Issuer	MUST contain the name of the embedded CA that issues the Subject Public Key certificate. The Issuer MAY be an ECA (i.e., the previous TCB layer) or an external CA. If the Issuer is an ECA, the Issuer MUST identify the TCB that issues this certificate.
Subject	MUST identify a TCB class or instance.
Subject Public Key Info	MUST contain a current TCB attestation public key and algorithm identifier.
Key Usage	If Subject is an ECA then this field MUST contain keyCertSign and MUST NOT contain cRLSign. Otherwise, this field MUST NOT contain keyCertSign.

Extended Key Usage	MAY contain any appropriate values for the usage model, e.g., id-kp-clientAuth for clients. This extension MUST contain either id-tcg-dice-kp-attestInit or id-tcg-dice-kp-attestLoc.
Basic Constraints	If Subject is an ECA then this field MUST contain cA:TRUE and pathLengthConstraint as appropriate. Otherwise, the certificate SHOULD NOT contain BasicConstraints.
Attestation Extensions	A future TCG specification will address attestation extensions.

Table 4: Attestation Identity certificate profile

5.1.6.5 Other DICE Certificates

This specification does not preclude the use of other types of certificates or credentials. However, the certificate profiles from the previous sections can be used as a conceptual guide for working with a layered DICE architecture.

5.2 DICE Certificate Media Types

Although DICE certificates are [3] and [6] compliant, the inclusion of DICE key purpose OIDs, as well as the inclusion of attestation Evidence (see [2]), indicate that the processing of certificates also requires the processing of attestation Evidence. Various data encapsulation structures (e.g., COSE [7]) allow inclusion of certificates used to validate a digital signature. Because DICE certificates expect special processing, use of a DICE-specific media type or content format identifier is preferable over alternatives.

5.2.1 DICE Certificate Chain Type

DICE-specific media type signals data structure parsers that certificates containing attestation evidence are contained within the payload block. By observing the DICE-specific media type, parsers can route the payload block to a processor that supports DICE certificates and attestation evidence.

The DICE-specific media types are defined as follows:

- “application/dice-cert”
- “application/dice-chain”
- “application/dice-bag”
- “application/dice-chain+cbor”
- “application/dice-chain+json”

The application/dice-cert media type indicates the payload contains a DICE certificate.

```
DiceCertificate ::= Certificate
```

The application/dice-chain media type indicates the payload contains a sequence of one or more DICE certificates where the first certificate is the end-entity (a.k.a., alias) certificate. Additional, intermediate, intermediate alias, or root certificates MAY be included.

```
DiceCertificateChain ::= SEQUENCE OF Certificate
```

The application/dice-bag media type indicates the payload contains a set of one or more DICE certificates where end-entity (a.k.a., alias), intermediate, intermediate alias, root or unaffiliated certificates MAY be included in no specified order.

```
DiceCertificateBag ::= SET OF Certificate
```

The certificate structures use these object identifiers:

```
tcg-dice-Certificate OBJECT IDENTIFIER ::= { tcg-dice 50 1 }
tcg-dice-CertificateChain OBJECT IDENTIFIER ::= { tcg-dice 50 2 }
tcg-dice-CertificateBag   OBJECT IDENTIFIER ::= { tcg-dice 50 3 }
```

Start of Informative Comment

DICE certificates that are issued by an embedded CA (ECA) can be referred to as *alias* or *intermediate alias* certificates. The end entity certificate in a chain containing ECA-issued certificates is called the *alias* certificate. Non-DICE certificates may be included in a DICE certificate chain. For example, a traditional end-entity 802.1AR device ID certificate might double as a DICE ECA certificate with traditional non-DICE intermediate and root certificates.

End of Informative Comment

The application/dice-chain+cbor media type indicates the payload contains one or more DICE certificates encapsulated in a CBOR array. The array MAY contain an ordered or unordered list of certificates.

The application/dice-chain+json media type indicates the payload contains one or more DICE certificates encapsulated in a JSON array. The array MAY contain an ordered or unordered list of certificates.

5.2.2 Optional Parameters

5.2.2.1 Usage

An optional “usage” parameter distinguishes how to interpret array ordering semantics. If “usage=chain” then the CBOR array is a certificate chain with the end entity certificate appearing first (array position 0), followed by any number of intermediate certificates. If “usage=bag” then the CBOR array contains an unordered set of certificates and could contain unaffiliated certificates. By default, where usage is not specified, “usage=chain” is presumed.

5.2.2.2 Profile

The optional “profile” parameter allows API routers to dispatch payloads directly to a profile-specific processor without having to snoop the payload.

The profile parameter contains a text representation of an OID or URI [3].

Start of Informative Comment

The OID representation is fully qualified such as:

```
"application/dice-chain;profile=2.23.133"
```

The URL representation uses tag URI such as:

```
"application/dice-chain;profile=tag:trustedcomputinggroup.org,2025:my-extn"
```

End of Informative Comment

The profile information SHOULD be integrity-protected to prevent denial-of-service attacks.

Start of Informative Comment

Integrity protection could be achieved by including a copy of the profile information in a cryptographically-signed payload.

End of Informative Comment

5.3 DICE Certificate Encapsulation

DICE certificates can be encapsulated in various data structures such as signature blocks, for example COSE [8] and JOSE [9].

5.3.1 CBOR Encapsulation

A DICE certificate chain can be encapsulated in a DICE-chain data structure realized in CBOR as follows:

CDDL format:

```
DICE-chain = bstr / [ 2* certs: bstr ]
```

A DICE certificate bag can be encapsulated in a DICE-bag data structure realized in CBOR as follows:

CDDL format:


```
DICE-bag = bstr / [ 2* certs: bstr ]
```

The COSE signature block defines protected and unprotected headers that can contain certificates.

Start of Informative Comment

For example, a PKIX certificate chain encapsulated in a COSE signature block might be as follows:
CBOR diagnostic format:

```
18([  / COSE_Sign1 tag /
  h'{               / protected header (CBOR map, bstr-wrapped)/
     1: -7,         / alg = ES256 /
     33: [          / x5chain (label 33) /
       h'30820122300D06092A864886...', / end-entity cert(DER)/
       h'3082010A0282010100C3A3B5...', / intermediate cert /
       h'3082010A0282010100AABBCC...'  / another intermediate cert /
     ]
   }',
  { 4: h'4B6565794964' },  / unprotected header (kid = "KeyId")/
  h'546869732069732074686520706C6F6164',  / payload /
  h'...'  / signature /
])
```

A COSE signature block containing a DICE certificate chain uses a different IANA assigned label than 33 (e.g., TBA-272):

```
TBA-272: [          / DICE-chain (label TBA-272) /
       h'30820122300D06092A864886...', / end-entity cert(DER)/
       h'3082010A0282010100C3A3B5...', / intermediate cert /
       h'3082010A0282010100AABBCC...'  / another intermediate cert /
     ]
```

A COSE signature block containing a DICE certificate bag (e.g., TBA-271):

```
TBA-271: [          / DICE-bag (label TBA-271) /
       h'2F71F0F9F171F0F0FFB292A4...', / unaffiliated cert /
       h'3082010A0282010100C3A3B5...', / intermediate cert /
       h'30820122300D06092A864886...', / end-entity cert /
       h'3082010A0282010100AABBCC...'  / another intermediate cert /
     ]
```

End of Informative Comment

5.3.2 JSON Encapsulation

A DICE certificate chain or bag can be encapsulated in a JSON data structure using the JOSE conventions defined in [9]. In this case, the certificates are carried in the JOSE header using the `x5c` parameter.

CDDL format:

```
DICE-chain = { "dicec": base64url(DER-certificate)/[ 2* base64url(DER-certificate)]}
DICE-bag   = { "diceb": base64url(DER-certificate)/[ 2* base64url(DER-certificate)]}
DER-certificate = bytes
```

- Each element of the `x5c` array is a DER-encoded DICE certificate, base64url-encoded for JSON transport.
- When used as a **chain**, the array is ordered: the first element is the end-entity certificate, followed by intermediate certificates.

- When used as a **bag**, the array is unordered: the set of certificates is provided without ordering semantics.
- The JOSE signature and encryption objects define protected and unprotected headers that can contain these certificate arrays, in the same way COSE headers carry `x5chain` or `x5bag`.

Start of Informative Comment

For example, a PKIX certificate chain encapsulated in a JOSE signature block might be as follows:

```
{
  "alg": "ES256",
  "kid": "KeyId",
  "x5c": [
    "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A...", // end-entity cert (base64url DER)
    "MIIBozANBgkqhkiG9w0BAQEFAAOC...",     // intermediate cert
    "MIICtzCCAZ6gAwIChAIn..."             // another intermediate cert
  ]
}
```

A JOSE signature block containing a DICE certificate chain uses dicec header parameter. A JOSE signature block containing a DICE certificate bag uses diceb.

For example, a DICE certificate chain encapsulated in a JOSE signature block might be as follows:

```
{
  "dicec": [
    "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A...", // end-entity DICE cert (base64url DER)
    "MIIBozANBgkqhkiG9w0BAQEFAAOC...",     // intermediate DICE cert
    "MIICtzCCAZ6gAwIChAIn..."             // another intermediate DICE cert
  ]
}
```

For example, a DICE certificate bag encapsulated in a JOSE signature block might be as follows:

```
{
  "diceb": [
    "MIIBozANBgkqhkiG9w0BAQEFAAOC...",     // intermediate DICE cert (base64url DER)
    "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A...", // end-entity DICE cert
    "MIICtzCCAZ6gAwIBAgI..."              // unaffiliated DICE cert
  ]
}
```

End of Informative Comment

6 Security Considerations

Evidence appraisal is at the core of any attestation protocol flow that delivers Evidence to a Verifier. The Attester is usually required to provide Evidence as a condition for gaining access to resources, networks, or services. Attester Evidence therefore needs to be authentic, and integrity-protected so that Verifiers can perform appraisals that accurately represent the security posture of the Attester. Any mistake in the appraisal process could have security implications. For instance, it could lead to the subversion of an access control function, which creates a chance for privilege escalation.

Therefore, the Attester's code and configuration, especially those that collect and protect Evidence, are primary security assets that are assumed to be built and maintained as securely as possible.

The protection of the Attester system is assumed to be relevant throughout its entire lifecycle, from design to operation. This includes the following aspects:

- Minimizing implementation complexity (see also Section 6.1 of [4])
- Using memory-safe programming languages
- Using secure defaults
- Minimizing the attack surface by avoiding unnecessary features that could be exploited by attackers
- Applying the principle of least privilege to the system's users
- Minimizing the potential impact of security breaches by implementing separation of duties in both the software and operational architecture
- Conducting regular, automated audits and reviews of the system, such as ensuring that users' privileges are correctly configured and that any new code has been audited and approved by independent parties
- Failing securely in the event of errors to avoid compromising the security of the system.

The integrity of public and private key material and the secrecy of private key material is assumed to always be protected. This includes key material carried in attestation Evidence and key material used to verify the authority of Evidence (such as public keys that identify trusted Verifiers). For more detailed information on protecting Trust Anchors, refer to Section 12.4 of [1].

The Attester is assumed to use cryptographically-protected, mutually-authenticated secure channels when supplying Evidence to Verifiers to avoid man-in-the-middle attacks. Also consider minimizing the use of intermediaries: each intermediary becomes another party that needs to be trusted and therefore factored in the Attester's TCBs. Refer to Section 12.2 of [1] for information on Conceptual Messages protection.

7 IANA Considerations

7.1 Media Type Assignment

IANA is requested to assign the following media types in the "Media Types" registry [5].

7.1.1 application/dice-cert

This media type indicates the payload is a DER encoded certificate with DICE extensions.

Type name: application

Subtype name: dice-cert

Required parameters: None.

Optional parameters:

- profile — Indicates the profile context.

Encoding considerations: None.

Security considerations: See Section 6.

Interoperability considerations: None.

Published specification: See this specification.

Applications that use this media type: DICE.

Fragment identifier considerations: None

Additional information:

- **Magic number(s):** None
- **File extension(s):** .dice
- **Macintosh file type code(s):** None

Person & email address to contact for further information: iana-requests&trustedcomputinggroup.org

Intended usage: COMMON

Author: N. Smith <ned.smith.ietf@outlook.com>

Change controller: Trusted Computing Group (TCG)

7.1.2 application/dice-chain

This media type indicates the payload is a DER encoded chain (SEQUENCE) of DER encoded certificates, some of which contain DICE extensions.

Type name: application

Subtype name: dice-chain

Required parameters: None.

Optional parameters:

- profile — Indicates the profile context.

Encoding considerations: None.

Security considerations: See Section 6.

Interoperability considerations: None.

Published specification: See this specification.

Applications that use this media type: DICE.

Fragment identifier considerations: None

Additional information:

- **Magic number(s):** None
- **File extension(s):** .dice
- **Macintosh file type code(s):** None

Person & email address to contact for further information: iana-requests&trustedcomputinggroup.org

Intended usage: COMMON

Author: N. Smith <ned.smith.ietf@outlook.com>

Change controller: Trusted Computing Group (TCG)

7.1.3 application/dice-bag

This media type indicates the payload is a DER encoded bag (SET) of DER encoded certificates, some of which contain DICE extensions.

Type name: application

Subtype name: dice-bag

Required parameters: None.

Optional parameters:

- profile — Indicates the profile context.

Encoding considerations: None.

Security considerations: See Section 6.

Interoperability considerations: None.

Published specification: See this specification.

Applications that use this media type: DICE.

Fragment identifier considerations: None

Additional information:

- **Magic number(s):** None
- **File extension(s):** .dice
- **Macintosh file type code(s):** None

Person & email address to contact for further information: iana-requests&trustedcomputinggroup.org

Intended usage: COMMON

Author: N. Smith <ned.smith.ietf@outlook.com>

Change controller: Trusted Computing Group (TCG)

7.1.4 application/dice-chain+cbor

This media type indicates the payload is a CBOR encapsulation of a ordered sequence (chain) or unordered set (bag) of DER encoded PKIX certificates.

Type name: application

Subtype name: dice-chain+cbor

Required parameters: None.

Optional parameters:

- usage=chain – Indicates an ordered sequence of DER encoded PKIX certificates. “chain” is the default if unspecified.
- usage=bag – Indicates an unordered set of DER encoded PKIX certificates.
- profile — Indicates the profile context.

Encoding considerations: None.

Security considerations: See Section 6.

Interoperability considerations: None.

Published specification: See this specification.

Applications that use this media type: DICE.

Fragment identifier considerations: None

Additional information:

- **Magic number(s):** None
- **File extension(s):** .dicecbor
- **Macintosh file type code(s):** None

Person & email address to contact for further information: iana-requests&trustedcomputinggroup.org

Intended usage: COMMON

Author: N. Smith <ned.smith.ietf@outlook.com>

Change controller: Trusted Computing Group (TCG)

7.1.5 application/dice-chain+json

This media type indicates the payload is a JSON encapsulation of a ordered sequence (chain) or unordered set (bag) of DER encoded PKIX certificates.

Type name: application

Subtype name: dice-chain+json

Required parameters: None.

Optional parameters:

- usage=chain – Indicates an ordered sequence of DER encoded PKIX certificates. “chain” is the default if unspecified.
- usage=bag – Indicates an unordered set of DER encoded PKIX certificates.
- profile — Indicates the profile context.

Encoding considerations: None.

Security considerations: See Section 6.

Interoperability considerations: None.

Published specification: See this specification.

Applications that use this media type: DICE.

Fragment identifier considerations: None

Additional information:

- **Magic number(s):** None
- **File extension(s):** .dicejson
- **Macintosh file type code(s):** None

Person & email address to contact for further information: iana-requests&trustedcomputinggroup.org

Intended usage: COMMON

Author: N. Smith <ned.smith.ietf@outlook.com>

Change controller: Trusted Computing Group (TCG)

7.2 CoAP Content-Format ID Assignments

IANA is requested to assign the following Content-Format numbers in the "CoAP Content-Formats" sub-registry, within the "Constrained RESTful Environments (CoRE) Parameters" Registry [10].

Content-Type	Content Coding	ID	Reference
“application/dice-cert”	-	TBA1	This specification
“application/dice-chain”	-	TBA2	This specification
“application/dice-bag”	-	TBA3	This specification
“application/dice-chain+cbor”	-	TBA4	This specification
“application/dice-chain+cbor;usage=bag”	-	TBA5	This specification
“application/dice-chain+json”	-	TBA6	This specification
“application/dice-chain+json;usage=bag”	-	TBA7	This specification

Table 5 - IANA Content Format ID Assignment

8 ASN.1

The key purpose object identifier exports are as follows:

```
TcgDiceKeyPurpose DEFINITIONS AUTOMATIC TAGS ::= BEGIN

EXPORTS ALL;

tcg OBJECT IDENTIFIER ::= { 2 23 tcg(133) }
tcg-dice OBJECT IDENTIFIER ::= { tcg platformClass(5) dice(4) }
tcg-dice-kp OBJECT IDENTIFIER ::= { tcg-dice kp(100) }
tcg-dice-kp-identityInit OBJECT IDENTIFIER ::= { tcg-dice-kp identity-init(6) }
tcg-dice-kp-identityLoc OBJECT IDENTIFIER ::= { tcg-dice-kp identity-loc(7) }
tcg-dice-kp-attestInit OBJECT IDENTIFIER ::= { tcg-dice-kp attest-init(8) }
tcg-dice-kp-attestLoc OBJECT IDENTIFIER ::= { tcg-dice-kp attest-loc(9) }
tcg-dice-kp-assertInit OBJECT IDENTIFIER ::= { tcg-dice-kp assert-init(10) }
tcg-dice-kp-assertLoc OBJECT IDENTIFIER ::= { tcg-dice-kp assert-loc(11) }
tcg-dice-kp-eca OBJECT IDENTIFIER ::= { tcg-dice-kp eca(12) }

END
```

The DICE certificate definition and exports are as follows:

```
DICE-CERTIFICATES { tcg-dice 50 }
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

EXPORTS     DiceCertificate,
            DiceCertificateChain,
            DiceCertificateBag,
            tcg-dice-Certificate,
            tcg-dice-CertificateChain,
            tcg-dice-CertificateBag;

IMPORTS
    Certificate
        FROM PKIX1Explicit88 {iso(1) identified-organization(3) dod(6) internet(1)
 security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18)};

tcg-dice-Certificate OBJECT IDENTIFIER ::= { tcg-dice 50 1 }
tcg-dice-CertificateChain OBJECT IDENTIFIER ::= { tcg-dice 50 2 }
tcg-dice-CertificateBag   OBJECT IDENTIFIER ::= { tcg-dice 50 3 }

DiceCertificate ::= Certificate
DiceCertificateChain ::= SEQUENCE OF Certificate
DiceCertificateBag   ::= SET OF Certificate

END
```

Start of Informative Comment

Note: PKIX1Explicit88 is defined by RFC5912.

End of Informative Comment

