Weave Digital Certificates

Data Structure Specification

Revision 0.1
2017-03-03

Abstract

This document is a technical specification of the format and operational semantics of the *Weave Digital Certificate* data structure, which is sometimes used for authenticating users and devices in the *Weave* system.

Introduction

This document is a specification of the Weave Digital Certificate data structure (hereafter called “Weave certificate”), which is sometimes used as a compact alternative to the standard Public Key Infrastructure (X.509) certificate format [RFC5280] (hereafter called “PKIX certificate”).

There exists a non-surjective injection of Weave certificates into PKIX certificates, i.e., each Weave certificate can be expressed as exactly one PKIX certificate, but only *some* (not all) PKIX certificates can be expressed with exactly one corresponding Weave certificate.

PKIX certificates expressible as Weave certificates are constrained by certain conditions, among them that ASN.1 object identifiers (OID) are referenced by their index into a registry of valid arcs. As a result, only some attribute types for relative distinguished names are valid, only certain cryptographic algorithms are used, and only a limited set of extensions are used.

Also, to compress the structure more efficiently than PKIX certificates, Weave certificates are encoded with the Weave TLV structured data interchange language [WTLV] instead of the ASN.1 Distinguished Encoding Rules (DER) [X690].

The plaintext input to the signature included in a Weave certificate is the Basic Syntax of the corresponding PKIX certificate, not the preceding Weave TLV data in the Weave certificate structure. Accordingly, validating the signature in a Weave certificate entails its conversion to the corresponding PKIX certificate to recover the original plaintext of the Basic Syntax signed by the Certificate Authority (CA).

Special Requirements Language

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are meant to be interpreted in accordance with “Key words for use in RFCs to Indicate Requirement Levels” [RFC2119].

Table Of Contents

Overview

This section introduces the Weave certificate at a conceptual level as a document that identifies the owner of a cryptographic public key, and which can be used in proving that the issuer of the certificate possesses the key.

The structure of Weave certificates and PKIX certificates are similar on cursory overview. Each conceptually comprises a record of the following fields:

- Certificate Text
  - Version Number
  - Serial Number
  - Signature Algorithm ID
  - Issuer Name
  - Validity period
    - Not Before

Not After
Subject name
Subject Public Key Info
Public Key Algorithm
Subject Public Key
Issuer Unique Identifier
Subject Unique Identifier
Extensions
Certificate Signature Algorithm
Certificate Signature

The main difference between Weave and PKIX certificates at the conceptual level is that Weave certificates are encoded in the Weave TLV interchange language, and they use a more constrained definition of the admissible values for each field, e.g., only some of the cryptographic algorithms are permitted, additional limitations apply on the range of valid serial numbers, the valid set of relative distinguished names is limited to a subset of attributes.

Certificate Types

There are four types of Weave certificates, as show in the following table:

Type	**Description**
Device	Certifies the identity of a Weave device.
Service Endpoint	Certifies the identity of a Weave service endpoint.
Authority	Certifies the signing authority of Weave certificates.
Software Publisher	Certifies the identity of a Weave software publisher.

Each of these types defines a distinguished name attribute for use in the Issuer and Subject name fields of PKIX certificates.

Detailed Requirements

This section provides a technical specification of the structure of data comprising a Weave certificate with accompanying requirements for A) their semantic validation, and B) their conversion to and from PKIX certificates. In some cases as noted, the limitations on the semantic interpretation of parts of a Weave certificate follow from the same limitations applied by “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile” [RFC5280].

Weave certificates are encoded in the Weave TLV Format [WTLV]. Accordingly, this document uses the proposed data modeling language in *Using CDDL To Model Weave TLV Structured Data* [CDDL/WEAVE], called CDDL/Weave in this document.
Note: The structured data interchange language used in Weave certificates is the Weave TLV Format, and not the Concise Binary Object Representation (CBOR) [RFC7049].

Weave Packed Certificate Time

The ASN.1 standard representation of date/time information used in declaring the validity period of a PKIX certificate provides a choice of two representations: the `UTCTime` type and the `GeneralizedTime` type. Each of these is capable of representing a Coordinated Universal Time (UTC) epoch by the combination of a Gregorian calendar date and a 24-hour clock time with leap seconds. The main difference between them is that `UTCTime` uses a two-digit year and `GeneralizedTime` uses a four-digit year.

PKIX certificates require the use of the `UTCTime` type for all calendar years between 1950 CE and 2049 CE, and the use of the `GeneralizedTime` type for all other calendar years between 0 CE and 9999 CE. It further specifies that PKIX certificates use the special date/time 9999-12-31 23:59:59 UTC to signify that they have no well-defined expiration date.

In contrast, Weave certificates use the special Weave Packed Certificate Time format, which uses a 32-bit unsigned integer code to represent a useful subset of the range that `GeneralizedTime` can represent. The code representing a date/time `$Y-$M-$D $h:$m:$s` UTC is computed according to the following formula:

```
Let $Y0 = $Y - 2000 in (* adjust range to 0..135 *)
Let $M0 = $M - 1 in (* adjust range to 0..11 *)
Let $D0 = $D - 1 in (* adjust range to 0..30 *)
Let $s0 = if $s > 59 then 59 else $s in (* leap seconds are ignored *)
(((((((($Y0 * 12) + $M0) * 31) + $D0) * 60) + $h) * 60) + $m) * 60) + $s0
```

Zero is a special value in the Weave Packed Certificate Time format, used to signify the Unspecified date, i.e., the date that appears in PKIX certificates as 9999-12-31 23:59:59 UTC.

Note: Not every unsigned 32-bit integer is a valid Weave Packed Certificate Time. Some values when converted to ISO 8601 date/time information would indicate invalid dates, e.g., Feb 29, 2017. Furthermore, Weave Packed Certificate Time values are not valid for years before 2000 CE and after 2133 CE.

Accordingly, the Weave/CDDL type definition for a Weave Packed Certificate Time is a simple 32-bit unsigned integer.

```
WeaveTime = uint .size 4
```

ASN.1 Object Identifiers (OID)

Several important components of PKIX certificates follow the pattern commonly used in ASN.1 data models where sum types are constructed with an ASN.1 object identifier (OID) to identify each variant. For example, the cryptographic algorithm used in the digital signature is identified by its OID.

Weave certificates do not use ASN.1 object identifiers. Instead, each valid OID is mapped within its *reference category* to a Weave TLV context-specific *tag number*. Each OID reference category defines the context of the tag number, and OID values are assigned to reference categories according to the type of fields where they can appear in PKIX certificates.

Accordingly, this document uses the `wtlv-tag-mkx<n>` generic type constructor to specify Weave OID references in the data model of the Weave certificate.

X.501 Distinguished Names

Several important components of PKIX certificates contain fields defined to carry Distinguished Name (DN) values as defined in “Open System Interconnect — The Directory: Models” [X501], e.g., the *Issuer Name* and the *Subject Name*. The ASN.1 format of a DN is a sequence of Relative Distinguished Name (RDN) values, where each RDN is a structure of attributes labeled with OID values.

In a Weave certificate, each DN is encoded as the sequence of RDN elements enclosed in a path container with a context-specific tag of the Weave OID reference for identifying the specific field in the certificate where the DN is used. Any RDN comprising a single attribute is encoded as a Weave TLV element tagged with the Weave OID reference identifying the attribute type, while RDN values comprising more than one attribute are encoded as anonymous structure containers containing the elements representing the attribute-value pairs.

Each RDN attribute is generally encoded in Weave TLV as UTF-8 strings with a context-specific tag comprising the Weave OID reference for the associated DN attribute type.

The following table shows the Weave OID references reserved for DN attribute types used in Weave certificates.

Tag Code	ASN.1 Object Identifier
1	joint_iso_ccitt(2) ds(5) attributeType(4) commonName(3)
2	joint_iso_ccitt(2) ds(5) attributeType(4) surname(4)
3	joint_iso_ccitt(2) ds(5) attributeType(4) serialNumber(5)
4	joint_iso_ccitt(2) ds(5) attributeType(4) countryName(6)
5	joint_iso_ccitt(2) ds(5) attributeType(4) localityName(7)
6	joint_iso_ccitt(2) ds(5) attributeType(4) stateOrProvinceName(8)
7	joint_iso_ccitt(2) ds(5) attributeType(4) organizationName(10)
8	joint_iso_ccitt(2) ds(5) attributeType(4) organizationalUnitName(11)
9	joint_iso_ccitt(2) ds(5) attributeType(4) title(12)
10	joint_iso_ccitt(2) ds(5) attributeType(4) name(41)
11	joint_iso_ccitt(2) ds(5) attributeType(4) givenName(42)
12	joint_iso_ccitt(2) ds(5) attributeType(4) initials(43)
13	joint_iso_ccitt(2) ds(5) attributeType(4) generationQualifier(44)
14	joint_iso_ccitt(2) ds(5) attributeType(4) dnQualifier(46)
15	joint_iso_ccitt(2) ds(5) attributeType(4) pseudonym(65)
16	itu_t(0) data(9) pss(2342) ucl(19200300) pilot(100) pilotAttributeType(1) domainComponent(25)
17	iso(1) organization(3) dod(6) internet(1) private(4) enterprise(1) nest(41387) weave(1) weaveDeviceId(1)
18	iso(1) organization(3) dod(6) internet(1) private(4) enterprise(1) nest(41387) weave(1) weaveServiceEndpointId(1)
19	iso(1) organization(3) dod(6) internet(1) private(4) enterprise(1) nest(41387) weave(1) weaveCAId(1)
20	iso(1) organization(3) dod(6) internet(1) private(4) enterprise(1) nest(41387) weave(1) weaveSoftwarePublisherId(1)

There are four special Weave certificate RDN attribute types, which are listed in the above table with the 1.3.6.1.4.1.41387.1 private arc. These OID values are assigned by Nest Labs, Inc. for use with Weave. The type for each of these attribute types is a 64-bit Extended Unique Identifier [EUI]. They are encoded in Weave TLV as 64-bit unsigned integers, and in the corresponding PKIX certificate, as UTF-8 text strings comprising a 16-digit hexadecimal number.

A further complication is that most of the standard RDN attribute types can be encoded in PKIX certificates as either UTF-8 strings or as ISO 646 strings. In Weave certificates, the most significant bit of the context-specific tag is logical-OR applied to the Weave OID reference index to signal that the corresponding PKIX encoding of the attribute uses the IA5 string format (for ISO 646 characters) instead of UTF-8. The only exception to this rule is the `domainComponent` attribute, i.e., OID 0.9.2342.19200300.100.1.25, which is always encoded in PKIX certificates with IA5 strings.

Accordingly, the following CDDL/Weave preamble is defined for modeling the components of a Weave certificate that comprise X.501 Distinguished Name values:

```
WeaveOID-AttributeType-UTF8 = &(
  commonName:                 wtlv-tag-mkx<1>
  surname:                    wtlv-tag-mkx<2>
  serialNumber:               wtlv-tag-mkx<3>
  countryName:                wtlv-tag-mkx<4>
  localityName:               wtlv-tag-mkx<5>
  stateOrProvinceName:        wtlv-tag-mkx<6>
  organizationName:           wtlv-tag-mkx<7>
  organizationalUnitName:     wtlv-tag-mkx<8>
  title:                      wtlv-tag-mkx<9>
  name:                       wtlv-tag-mkx<10>
  givenName:                  wtlv-tag-mkx<11>
  initials:                   wtlv-tag-mkx<12>
  generationQualifier:        wtlv-tag-mkx<13>
  dnQualifier:                wtlv-tag-mkx<14>
  pseudonym:                  wtlv-tag-mkx<15>
)

WeaveOID-AttributeType-IA5 = &(
  domainComponent:           wtlv-tag-mkx<16>,
  commonNameIA5:             wtlv-tag-mkx<0x81>,
  surnameIA5:                wtlv-tag-mkx<0x82>,
  serialNumberIA5:           wtlv-tag-mkx<0x83>,
  countryNameIA5:            wtlv-tag-mkx<0x84>,
  localityNameIA5:           wtlv-tag-mkx<0x85>,
  stateOrProvinceNameIA5:    wtlv-tag-mkx<0x86>,
  organizationNameIA5:       wtlv-tag-mkx<0x87>,
  organizationalUnitNameIA5: wtlv-tag-mkx<0x88>,
  titleIA5:                  wtlv-tag-mkx<0x89>,
```

```
  nameIA5:                    wtlv-tag-mkx<0x8A>,
  givenNameIA5:               wtlv-tag-mkx<0x8B>,
  initialsIA5:                wtlv-tag-mkx<0x8C>,
  generationQualifierIA5:     wtlv-tag-mkx<0x8D>,
  dnQualifierIA5:             wtlv-tag-mkx<0x8E>,
  pseudonymIA5:               wtlv-tag-mkx<0x8F>,
)

WeaveOID-AttributeType-EUI = &(
  weaveDeviceId:              wtlv-tag-mkx<17>,
  weaveServiceId:             wtlv-tag-mkx<18>,
  weaveCAId:                  wtlv-tag-mkx<19>,
  weaveSoftwarePublisherId:   wtlv-tag-mkx<20>,
)

Weave-Attribute =
  $$Weave-Attribute .within ( wtlv-tag-context => wtlv-anon )

$$Weave-Attribute //= ( WeaveOID-AttributeType-UTF8 => tstr )
$$Weave-Attribute //= ( WeaveOID-AttributeType-IA5 => tstr )
$$Weave-Attribute //= ( WeaveOID-AttributeType-EUI => uint .size 8 )

Weave-RDN = { +Weave-Attribute }
Weave-DN = wtlv-tag-mks<[ +Weave-RDN ]>
```

Data Structure

Weave certificates are encoded in Weave TLV as a structure container with a fully-qualified profile-specific tag for the Weave certificate in the Security profile registered for the Common vendor in [PROFILES].

Accordingly, the following Weave/CDDL excerpt provides definitions used in the modeling of the Weave certificate data structure.

```
Profile-security = (
  Vendor: 0x0000,       ; Vendor identifier for common profiles
  Profile: 0x0004       ; Security profile identifier
)

Tag-Certificate = wtlv-tag-mkq<[Profile-security, 1]>
Weave-Certificate = { Tag-Certificate => Fields-Certificate }
```

The `Fields-Certificate` type is free in the CDDL/Weave excerpt above, and its definition is completed in the Certificate section below by composing all the field definitions into a structure container.

Serial Number

The serial number element of Weave certificates corresponds to the serial number field of PKIX certificates. Weave certificates follow the same limitation on admissible serial numbers as in [RFC5280], i.e., that implementations MUST admit serial numbers up to 20 octets in length, and certificate authorities MUST NOT use serial numbers larger than 20 octets in length.

The context-specific tag number `0x01` identifies the Serial Number field in the Weave certificate structure. The value of the field is a Weave TLV byte sequence between 0 and 20 octets in length.

Accordingly, the CDDL/Weave model for the Serial Number element is shown below:

```
Tag-SerialNumber = wtlv-tag-mkx<0x01>
Field-SerialNumber = ( Tag-SerialNumber => bstr .size 0..20 )
```

Signature Algorithm

Like PKIX certificates, every Weave certificate includes a digital signature in its Signature component. The Signature Algorithm component of the Weave certificate specifies the cryptographic algorithms used for composing and validating the signature embedded in the Signature component of the certificate.

The context-specific tag number 0x02 identifies the Signature Algorithm field in the Weave certificate structure. The value of the field is either a Weave OID reference or an anonymous array of length 2 or more, comprising a Weave OID reference followed by one or more parameters of any anonymous Weave TLV type.

The following Weave OID references are reserved in the context for signature algorithms:

Tag Code	**ASN.1 Object Identifier**
1	`iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) md2WithRSAEncryption(2)`
2	`iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) md5WithRSAEncryption(4)`
3	`iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) sha-1WithRSAEncryption(5)`
4	`iso(1) member-body(2) us(840) ansi-x962(10045) signatures(4) ecdsa-with-SHA1(1)`

5	iso(1) member-body(2) us(840) ansi-x962(10045) signatures(4) ecdsa-with-SHA2(3) ecdsa-with-SHA256(2)

Accordingly, the CDDL/Weave model for the Signature Algorithm element is shown below:

```
Tag-SignatureAlgorithm = wtlv-tag-mkx<0x02>

WeaveOID-SignatureAlgorithm = &(
  md2WithRSAEncryption:       wtlv-tag-mkx<1>,
  md5WithRSAEncryption:       wtlv-tag-mkx<2>,
  sha1WithRSAEncryption:      wtlv-tag-mkx<3>,
  ecdsaWithSHA1:              wtlv-tag-mkx<4>,
  ecdsaWithSHA256:            wtlv-tag-mkx<5>,
)

$Opt-SigAlgo /= WeaveOID-SignatureAlgorithm
$Opt-SigAlgo /= [ oid: WeaveOID-SignatureAlgorithm, +wtlv-anon ]

Field-SignatureAlgorithm = ( Tag-SignatureAlgorithm => $Opt-SigAlgo )
```

Issuer Name

The context-specific tag number 0x03 identifies the Issuer Name field in the Weave certificate structure. The value of the field is a Distinguished Name. Accordingly, the CDDL/Weave model for the Issuer Name element is shown below:

```
Tag-IssuerName = wtlv-tag-mkx<0x03>
Field-IssueName = ( Tag-IssuerName => Weave-DN )
```

Validity

The context-specific tag numbers 0x04 and 0x05 identify the Not Before and Not After fields in the Weave certificate structure, which indicate the period of validity for the certificate. The value of each of these fields is a Weave Packed Certificate Time format integer code.

The Validity field in a PKIX certificate is encoded in ASN.1 as a sequence with exactly two elements, the notBefore and the notAfter elements, encoded as Time type values. In a Weave certificate, these two fields are encoded as separate fields of the top level `Map-CertificateFields` structure.

Accordingly, the CDDL/Weave model for the Issuer Name element is shown below:

```
Tag-NotBefore     = wtlv-tag-mkx<0x04>
Tag-NotAfter      = wtlv-tag-mkx<0x05>

Fields-Validity = (
  Tag-NotBefore   => WeaveTime,
  Tag-NotAfter    => WeaveTime
)
```

Subject Name

The context-specific tag number 0x06 identifies the Subject Name field in the Weave certificate structure. The value of the field is a Distinguished Name. Accordingly, the CDDL/Weave model for the Issuer Name element is shown below:

```
Tag-SubjectName = wtlv-tag-mkx<0x06>
Field-SubjectName = ( Tag-SubjectName => Weave-DN )
```

Public Key Information

The public key information in a Weave certificate is a structure comprising A) the Public Key Algorithm identification, and B) the Public Key material, which has different types depending on the algorithm identified. This information comprises a system of fields in the Weave certificate structure.

The context-specific tag number 0x07 identifies the Public Key Algorithm field in the Weave certificate structure. The value of the field is a Weave OID reference according to the following table:

Tag Code	**ASN.1 Object Identifier**
1	`iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) rsaEncryption(1)`
2	`iso(1) member-body(2) us(840) ansi-x962(10045) keyType(2) ecPublicKey(1)`
3	`iso(1) identified-organization(3) certicom(132) schemes(1) ecdh(12)`
4	`iso(1) identified-organization(3) certicom(132) schemes(1) ecmqv(13)`

If the Public Key Algorithm is one of `ecPublicKey`, `ecdh`, or `ecmqv`, then the context-specific tag number 0x08 further identifies the Elliptical Curve Identifier. These identifiers are Weave OID references according to the following table:

Tag Code	ASN.1 Object Identifier
1	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2pnb163v1(1)
2	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2pnb163v2(2)
3	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2pnb163v3(3)
4	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2pnb176w1(4)
5	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2tnb191v1(5)
6	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2tnb191v2(6)
7	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2tnb191v3(7)
8	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2onb191v4(8)
9	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2onb191v5(9)
10	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2pnb208w1(10)
11	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2tnb239v1(11)
12	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2tnb239v2(12)
13	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2tnb239v3(13)
14	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2onb239v4(14)
15	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2onb239v5(15)
16	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2pnb272w1(16)
17	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2pnb304w1(17)
18	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2tnb359v1(18)
19	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2pnb368w1(19)
20	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) characteristicTwo(0) c2tnb431r1(20)

21	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) prime(1) prime192v1(1)
22	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) prime(1) prime192v2(2)
23	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) prime(1) prime192v3(3)
24	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) prime(1) prime239v1(4)
25	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) prime(1) prime239v2(5)
26	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) prime(1) prime239v3(6)
27	iso(1) member_body(2) us(840) ansi-x962(10045) curves(3) prime(1) prime256v1(7)
28	iso(1) organization(3) certicom(132) curve(0) secp112r1(6)
29	iso(1) organization(3) certicom(132) curve(0) secp112r2(7)
30	iso(1) organization(3) certicom(132) curve(0) secp128r1(28)
31	iso(1) organization(3) certicom(132) curve(0) secp128r2(29)
32	iso(1) organization(3) certicom(132) curve(0) secp160k1(9)
33	iso(1) organization(3) certicom(132) curve(0) secp160r1(8)
34	iso(1) organization(3) certicom(132) curve(0) secp160r2(30)
35	iso(1) organization(3) certicom(132) curve(0) secp192k1(31)
36	iso(1) organization(3) certicom(132) curve(0) secp224k1(32)
37	iso(1) organization(3) certicom(132) curve(0) secp224r1(33)
38	iso(1) organization(3) certicom(132) curve(0) secp256k1(10)
39	iso(1) organization(3) certicom(132) curve(0) secp384r1(34)
40	iso(1) organization(3) certicom(132) curve(0) secp521r1(35)
41	iso(1) organization(3) certicom(132) curve(0) sect113r1(4)
42	iso(1) organization(3) certicom(132) curve(0) sect113r2(5)
43	iso(1) organization(3) certicom(132) curve(0) sect131r1(22)
44	iso(1) organization(3) certicom(132) curve(0) sect131r2(23)
45	iso(1) organization(3) certicom(132) curve(0) sect163k1(1)
46	iso(1) organization(3) certicom(132) curve(0) sect163r1(2)
47	iso(1) organization(3) certicom(132) curve(0) sect163r2(15)
48	iso(1) organization(3) certicom(132) curve(0) sect193r1(24)
49	iso(1) organization(3) certicom(132) curve(0) sect193r2(25)
50	iso(1) organization(3) certicom(132) curve(0) sect233k1(26)
51	iso(1) organization(3) certicom(132) curve(0) sect233r1(27)
52	iso(1) organization(3) certicom(132) curve(0) sect239k1(3)

53	iso(1) organization(3) certicom(132) curve(0) sect283k1(16)
54	iso(1) organization(3) certicom(132) curve(0) sect283r1(17)
55	iso(1) organization(3) certicom(132) curve(0) sect409k1(36)
56	iso(1) organization(3) certicom(132) curve(0) sect409r1(37)
57	iso(1) organization(3) certicom(132) curve(0) sect571k1(38)
58	iso(1) organization(3) certicom(132) curve(0) sect571r1(39)

If the Public Key Algorithm is `rsaEncryption`, then the context-specific tag number 0x09 identifies the RSA Public Key Material in the Weave certificate structure. The type of its value is a structure container with at least two elements, as shown in the following list:

- Context-specific tag number 0x01 identifies the RSA Key Modulus, which is byte sequence of the appropriate length.

- Context-specific tag number 0x02 identifies the RSA Key Exponent, which is an unsigned 64-bit integer.

If the Public Key Algorithm is one of `ecPublicKey`, `ecdh`, or `ecmqv`, then the context-specific tag number 0x0A identifies the Elliptic Curve Public Key Material in the Weave certificate structure. The type of its value is a byte sequence of the appropriate length, including the EC code point.

Accordingly, the Weave/CDDL model for the OID references used in the Public Key Information elements is shown below:

```
; Weave certificate structure fields
Tag-PublicKeyAlgorithm  = wtlv-tag-mkx<0x07>
Tag-RSAPublicKey        = wtlv-tag-mkx<0x09>

Fields-PublicKeyInformation = $$Fields-PublicKeyInformation

; RSA Public Key Information
WeaveOID-RSAPublicKeyAlgorithm = wtlv-tag-mkx<0x01> ; OID reference

Tag-RSAModulus    = wtlv-tag-mkx<0x01>
Tag-RSAExponent   = wtlv-tag-mkx<0x02>

$$Fields-PublicKeyInformation //= (
  Tag-PublicKeyAlgorithm => WeaveOID-RSAPublicKeyAlgorithm,
  Tag-RSAPublicKey => {
    Tag-RSAModulus => bstr,
    Tag-RSAExponent => uint .size 8,
  }
)
```

```
; EC Public Key Information
WeaveOID-ECPublicKeyAlgorithm = &(
  ecPublicKey:    wtlv-tag-mkx<2>,
  ecdh:           wtlv-tag-mkx<3>,
  ecmqv:          wtlv-tag-mkx<4>,
)

WeaveOID-EllipticalCurveId = &(
  c2pnb163v1: wtlv-tag-mkx<1>,
  c2pnb163v2: wtlv-tag-mkx<2>,
  c2pnb163v3: wtlv-tag-mkx<3>,
  c2pnb176w1: wtlv-tag-mkx<4>,
  c2tnb191v1: wtlv-tag-mkx<5>,
  c2tnb191v2: wtlv-tag-mkx<6>,
  c2tnb191v3: wtlv-tag-mkx<7>,
  c2onb191v4: wtlv-tag-mkx<8>,
  c2onb191v5: wtlv-tag-mkx<9>,
  c2pnb208w1: wtlv-tag-mkx<10>,
  c2tnb239v1: wtlv-tag-mkx<11>,
  c2tnb239v2: wtlv-tag-mkx<12>,
  c2tnb239v3: wtlv-tag-mkx<13>,
  c2onb239v4: wtlv-tag-mkx<14>,
  c2onb239v5: wtlv-tag-mkx<15>,
  c2pnb272w1: wtlv-tag-mkx<16>,
  c2pnb304w1: wtlv-tag-mkx<17>,
  c2tnb359v1: wtlv-tag-mkx<18>,
  c2pnb368w1: wtlv-tag-mkx<19>,
  c2tnb431r1: wtlv-tag-mkx<20>,
  prime192v1: wtlv-tag-mkx<21>,
  prime192v2: wtlv-tag-mkx<22>,
  prime192v3: wtlv-tag-mkx<23>,
  prime239v1: wtlv-tag-mkx<24>,
  prime239v2: wtlv-tag-mkx<25>,
  prime239v3: wtlv-tag-mkx<26>,
  prime256v1: wtlv-tag-mkx<27>,
  secp112r1:  wtlv-tag-mkx<28>,
  secp112r2:  wtlv-tag-mkx<29>,
  secp128r1:  wtlv-tag-mkx<30>,
  secp128r2:  wtlv-tag-mkx<31>,
  secp160k1:  wtlv-tag-mkx<32>,
  secp160r1:  wtlv-tag-mkx<33>,
  secp160r2:  wtlv-tag-mkx<34>,
  secp192k1:  wtlv-tag-mkx<35>,
  secp224k1:  wtlv-tag-mkx<36>,
  secp224r1:  wtlv-tag-mkx<37>,
  secp256k1:  wtlv-tag-mkx<38>,
  secp384r1:  wtlv-tag-mkx<39>,
  secp521r1:  wtlv-tag-mkx<40>,
```

```
  sect113r1:  wtlv-tag-mkx<41>,
  sect113r2:  wtlv-tag-mkx<42>,
  sect131r1:  wtlv-tag-mkx<43>,
  sect131r2:  wtlv-tag-mkx<44>,
  sect163k1:  wtlv-tag-mkx<45>,
  sect163r1:  wtlv-tag-mkx<46>,
  sect163r2:  wtlv-tag-mkx<47>,
  sect193r1:  wtlv-tag-mkx<48>,
  sect193r2:  wtlv-tag-mkx<49>,
  sect233k1:  wtlv-tag-mkx<50>,
  sect233r1:  wtlv-tag-mkx<51>,
  sect239k1:  wtlv-tag-mkx<52>,
  sect283k1:  wtlv-tag-mkx<53>,
  sect283r1:  wtlv-tag-mkx<54>,
  sect409k1:  wtlv-tag-mkx<55>,
  sect409r1:  wtlv-tag-mkx<56>,
  sect571k1:  wtlv-tag-mkx<57>,
  sect571r1:  wtlv-tag-mkx<58>,
)

Tag-EllipticalCurveIdentifier = wtlv-tag-mkx<0x08>
Tag-EllipticalCurvePublicKey  = wtlv-tag-mkx<0x0A>

$$Fields-PublicKeyInformation //= (
  Tag-PublicKeyAlgorithm => WeaveOID-ECPublicKeyAlgorithm,
  Tag-EllipticalCurveIdentifier => WeaveOID-EllipticalCurveId,
  Tag-EllipticalCurvePublicKey => bstr,
)
```

Extensions

The context-specific tag numbers from 128 to 255 are reserved for identifying the Extension fields in the Weave certificate structure. The type of each extension field is a structure container with specific fields further described in the subsections below for each extension type.

The following Weave OID references are reserved for specific certificate extensions in the context of the Weave certificate structure:

Tag Code	**ASN.1 Object Identifier**
128	`joint-iso-itu-t(2) ds(5) certificateExtension(29) authorityKeyIdentifier(35)`
129	`joint-iso-itu-t(2) ds(5) certificateExtension(29) subjectKeyIdentifier(14)`

130	joint-iso-itu-t(2) ds(5) certificateExtension(29) keyUsage(15)
131	joint-iso-itu-t(2) ds(5) certificateExtension(29) basicConstraints(19)
132	joint-iso-itu-t(2) ds(5) certificateExtension(29) extendedKeyUsage(37)

Accordingly, the Weave/CDDL preamble for the OID references used identify the Extension fields is shown below:

```
Tag-ExtensionAuthorityKeyIdentifier = wtlv-tag-mkx<128>
Tag-ExtensionSubjectKeyIdentifier   = wtlv-tag-mkx<129>
Tag-ExtensionKeyUsage               = wtlv-tag-mkx<130>
Tag-ExtensionBasicConstraints       = wtlv-tag-mkx<131>
Tag-ExtensionExtendedKeyUsage       = wtlv-tag-mkx<132>

Field-Extension = $$Field-Extension .within wtlv-element
```

Authority Key Identifier Extension

Insert verbiage here.

Accordingly, the Weave/CDDL preamble for the OID references used identify the Authority Key Identifier Extension field is shown below:

```
Tag-AuthorityKeyIdentifier-Critical       = wtlv-tag-mkx<1>
Tag-AuthorityKeyIdentifier-KeyIdentifier  = wtlv-tag-mkx<2>
Tag-AuthorityKeyIdentifier-Issuer         = wtlv-tag-mkx<3>
Tag-AuthorityKeyIdentifier-SerialNumber   = wtlv-tag-mkx<4>

$$Field-Extension //= (
  Tag-ExtensionAuthorityKeyIdentifier => {
    ? Tag-AuthorityKeyIdentifier-Critical       => bool,
    ? Tag-AuthorityKeyIdentifier-KeyIdentifier  => bstr,
    ? Tag-AuthorityKeyIdentifier-Issuer         => Weave-DN,
    ? Tag-AuthorityKeyIdentifier-SerialNumber   => bstr,
  }
)
```

Subject Key Identifier Extension

Insert verbiage here.

Accordingly, the Weave/CDDL preamble for the OID references used identify the Subject Key Identifier Extension field is shown below:

```
Tag-SubjectKeyIdentifier-Critical         = wtlv-tag-mkx<1>
```

```
Tag-SubjectKeyIdentifier-KeyIdentifier    = wtlv-tag-mkx<2>

$$Field-Extension //= (
  Tag-ExtensionSubjectKeyIdentifier => {
    ? Tag-SubjectKeyIdentifier-Critical         => bool,
    ? Tag-SubjectKeyIdentifier-KeyIdentifier    => bstr,
  }
)
```

Key Usage Extension

Insert verbiage here.

Accordingly, the Weave/CDDL preamble for the OID references used identify the Key Usage Extension field is shown below:

```
Tag-KeyUsage-Critical   = wtlv-tag-mkx<1>
Tag-KeyUsage-KeyUsage   = wtlv-tag-mkx<2>

$$Field-Extension //= (
  Tag-ExtensionKeyUsage => {
    ? Tag-KeyUsage-Critical   => bool,
    ? Tag-KeyUsage-KeyUsage   => uint,
  }
)
```

Basic Constraints Extension

Insert verbiage here.

Accordingly, the Weave/CDDL preamble for the OID references used identify the Basic Constraints Extension field is shown below:

```
Tag-BasicConstraints-Critical             = wtlv-tag-mkx<1>
Tag-BasicConstraints-IsCA                 = wtlv-tag-mkx<2>
Tag-BasicConstraints-PathLenConstant      = wtlv-tag-mkx<3>


$$Field-Extension //= (
  Tag-ExtensionBasicConstraints => {
    ? Tag-BasicConstraints-Critical             => bool,
    ? Tag-BasicConstraints-IsCA                 => bool,
    ? Tag-BasicConstraints-PathLenConstant      => uint,
  }
)
```

Extended Key Usage Extension

Insert verbiage here.

Accordingly, the Weave/CDDL preamble for the OID references used identify the Extended Key Usage Extension field is shown below:

```
Tag-ExtendedKeyUsage-Critical = wtlv-tag-mkx<1>
Tag-ExtendedKeyUsage-KeyUsage = wtlv-tag-mkx<2>

$$Field-Extension //= (
  Tag-ExtensionExtendedKeyUsage => {
    ? Tag-ExtendedKeyUsage-Critical       => bool,
    ? Tag-ExtendedKeyUsage-KeyPurposes    => [ *uint ],
  }
)
```

Signature

Each of the two supported families of signature algorithm uses a different context-specific tag number to identify the Signature field in the Weave certificate structure.

RSA Signature

The context-specific tag number 0x0B identifies the Signature field in Weave certificates signed with RSA signature algorithms. The value field is a byte string of appropriate length for the signature algorithm. Accordingly, the CDDL/Weave model for the RSA Signature field is show below:

```
Tag-RSASignature = wtlv-tag-mkx<0x0B>
Field-RSASignature = ( Tag-RSASignature => bstr )
```

EDCA Signature

The context-specific tag number 0x0C identifies the Signature field in Weave certificates signed with EDCA signature algorithms. The value field is a structure container with two elements, one of each of the EDCA ‘r’ and ‘s’ components of a signature. The context-specific tags for these elements are 1 and 2 respectively. Their values are bytes strings of the appropriate length for the specific EDCA signature algorithm.

Accordingly, the CDDL/Weave model for the RSA Signature field is show below:

```
Tag-EDCASignatureValueR = wtlv-tag-mkx<1>
Tag-EDCASignatureValueS = wtlv-tag-mkx<2>

Map-EDCASignatureValues = {
```

```
  Tag-EDCASignatureValueR => bstr,
  Tag-EDCASignatureValueS => bstr
}

Tag-EDCASignature = wtlv-tag-mkx<0x0C>
Field-EDCASignature = ( Tag-EDCASignature => Map-EDCASignatureValues )
```

Unified Signature Choice

Exactly one signature field is appears in the Weave certificate structure. Accordingly, the Weave/CDDL model for the Signature field is a group choice, as shown below:

```
Field-Signature = Field-RSASignature / Field-EDCASignature
```

Certificate

Weave certificates are encoded in Weave TLV as structure container with a fully-qualified profile-specific tag for the Weave certificate in the Security profile registered for the Common vendor in [PROFILES].

At the top level, PKIX certificates comprise a structure of three fields:

1. `tbsCertificate` — a second-level structure comprising the Serial Number, Signature Algorithm, Issuer Name, Validity fields, Subject Name and Public Key Information fields and any additional Extension fields.

2. `signatureAlgorithm` — a copy of the Signature Algorithm field in the tbsCertificate structure above.

3. `signature` — a DER-encoded bit string containing the signature material for the certificate.

By contrast, Weave certificates are a single top-level structure with fields corresponding to all the fields of the `tbsCertificate` structure in the PKIX certificate, followed by an additional field containing the signature. The `signatureAlgorithm` field, which PKIX requires to have the same value as the signature field in the `tbsCertificate` structure, is elided from the Weave certificate to avoid duplication.

Accordingly, the Weave/CDDL model for the Weave Digital Certificate is shown below:

```
Map-Certificate = {
  Field-SerialNumber,
  Field-SignatureAlgorithm,
  Field-IssuerName,
  Fields-Validity,
```

```
  Field-SubjectName,
  Fields-PublicKeyInformation,
  *Field-Extension,
  Field-Signature
}

Profile-security = [
  Vendor: 0x0000,       ; Vendor identifier for common profiles
  Profile: 0x0004       ; Security profile identifier
]

Tag-Certificate = wtlv-tag-mkq<[Profile-security, 1]>
Weave-Certificate = { Tag-Certificate => Map-Certificate }
```

Further Considerations

This section contains informative background about the reasoning behind the requirements specified in the previous section.

References

Normative References

CDDL/WEAVE	Woodyatt J., Using CDDL To Model Weave TLV Structured Data, Technical Specification, v0.1, February 2017
EUI	Guidelines for 64-bit Global Identifier
ID.CDDL	C. Vigano and H. Berkholz, “CBOR data definition language (CDDL): a notational convention to express CBOR data structures”, I-D.greevenbosch-appsawg-cbor-cddl-09, September 2016.
RFC2119	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
RFC5280	D. Cooper et al., “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, RFC 5280, May 2008
X501	ITU-T Recommendation X.501 (10/16): Information technology - Open Systems Interconnection - The Directory: Models
X690	ITU-T Recommendation X.690 (08/15): Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

WTLV	Weave TLV, Specification

Informative References

PROFILES	Weave: Common Profile Identifier Registry, Revision 14, August 2016
RFC7049	C. Bormann and P. Hoffman, “Concise Binary Object Representation (CBOR)”, RFC 7049, October 2013
THIS	Template: Functional Requirements