Network Working Group
Internet-Draft
Updates: 6698 (if approved)
Intended status: Standards Track
Expires: 12 November 2026

J. Preu Mattsson
G. Selander
Ericsson AB
S. Raza
University of Glasgow
J. Hglund
RISE AB
M. Furuhed
IN Groupe
L. Liao
NIO

11 May 2026

CBOR Encoded X.509 Certificates (C509 Certificates)
draft-ietf-cose-cbor-encoded-cert-19

Abstract

This document specifies a CBOR encoding of X.509 certificates. The resulting certificates are called C509 certificates. The CBOR encoding supports a large subset of RFC 5280 and common certificate profiles, and it is extensible.

Two types of C509 certificates are defined. One type is an invertible CBOR re-encoding of DER-encoded X.509 certificates with the signature field copied from the DER encoding. The other type is identical except that the signature is computed over the CBOR encoding instead of the DER encoding, thereby avoiding the use of ASN.1. Both types of certificates have the same semantics as X.509 while providing comparable size reduction.

This document also specifies CBOR-encoded data structures for certification requests and certification request templates, new COSE headers, as well as a TLS certificate type and a file format for C509. This document updates RFC 6698 by extending the TLSA selectors registry to include C509 certificates.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-cose-cbor-encoded-cert/.

Discussion of this document takes place on the CBOR Object Signing and Encryption Working Group mailing list (mailto:cose@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/cose/. Subscribe at https://www.ietf.org/mailman/listinfo/cose/.

Source for this draft and an issue tracker can be found at https://github.com/cose-wg/CBOR-certificates.

Status of This Memo

This Internet-Draft will expire on 12 November 2026.

Copyright Notice

Table of Contents

1. Introduction

One of the challenges with deploying a Public Key Infrastructure (PKI) for the Internet of Things (IoT) is the size and parsing of X.509 public key certificates [RFC5280], since those are not optimized for constrained environments [RFC7228]. Large certificate chains are also problematic in non-constrained protocols such as EAP-TLS [RFC9190] [RFC9191] where authenticators typically drop an EAP session after only 4050 round-trips, QUIC [RFC9000] where the latency increases significantly unless the server sends less than three times as many bytes as received prior to validating the client address, and Resource Public Key Infrastructure (RPKI) [RFC6487] where a single certificate can be very large. More compact certificate representations are, therefore, desirable in many use cases.

X.509 certificates are defined using Abstract Syntax Notation One (ASN.1) and encoded using the Distinguished Encoding Rules (DER) [X.690]. This document specifies an alternative encoding of X.509 certificates using the Concise Binary Object Representation (CBOR) [RFC8949], initially proposed in [X.509-IoT]. The use of a more compact encoding reduces certificate size, which has known performance benefits in terms of decreased communication overhead, power consumption, latency, and storage requirements. The re-encoding of X.509 is called C509, and the resulting certificates are termed C509 certificates. C509 is not a general CBOR encoding for ASN.1 data structures.

CBOR is a data format designed for small code size and small message size in systems with very limited memory, processor power, and instruction sets. CBOR builds on the JSON data model but extends it by, for example, encoding binary data directly without base64 conversion. In addition to the binary CBOR encoding, CBOR also has a diagnostic notation that is human-readable and editable, which simplifies development and debugging. The Concise Data Definition Language (CDDL) [RFC8610] provides a way to express structures for protocol messages and APIs that use CBOR. [RFC8610] also extends the diagnostic notation. For a complete specification and examples, see [RFC8949], [RFC8610], and [RFC8742]. A tool for becoming familiar with CBOR, available at the time of publication, is the CBOR playground [CborMe].

The C509 encoding supports a large subset of [RFC5280] and all certificates profiled for [RFC7925], IEEE 802.1AR (DevID) [IEEE-802.1AR], CAB Baseline [CAB-TLS], [CAB-Code], RPKI [RFC6487],

Wi-SUN [Wi-SUN], and eUICC [GSMA-eUICC]. C509 is designed for small code size and compact encoding of certificates in constrained environments including certificates profiled specifically for IoT deployments, but can be applied to certificate-based authentication in general, for example, using TLS [RFC8446], QUIC [RFC9000], DTLS [RFC9147], COSE [RFC9052] and EDHOC [RFC9528]. This document does not specify a certificate profile.

At the time of publication, there are several C509 implementations targeting, for example, in-vehicle and vehicle-to-cloud communication, Uncrewed Aircraft Systems (UAS), and Global Navigation Satellite System (GNSS) deployments. When used to re-encode DER-encoded X.509 certificates, the CBOR encoding can reduce the size of [RFC7925]-profiled certificates by more than 50%; see Appendix A.

C509 is designed to be extensible to additional X.509 features, for example, support for new algorithms, including new Post-Quantum (PQ) algorithms, which can be registered in the IANA registry as they are specified; see Section 8.14.

This document defines two types of C509 using the same CBOR encoding and differing only in what is being signed:

1. An invertible CBOR re-encoding of DER-encoded X.509 certificates [RFC5280], which can be reversed to obtain the original DER-encoded X.509 certificate, which can be verified using legacy certificate software. Due to the widespread deployment of X.509, it is necessary to allow backward compatibility.

2. Natively signed C509 certificates, where the signature is calculated over the CBOR encoding instead of the DER encoding. This removes the need for ASN.1 and DER parsing and the associated complexity, but such certificates are not backward compatible with implementations requiring DER-encoded X.509. Natively signed C509 certificates can be used in devices that are only required to authenticate to servers compatible with natively signed C509 certificates. This is not a major restriction in many IoT deployments in which the parties that issue and verify certificates are part of a restricted ecosystem.

This document also specifies C509 Certification Requests; see Section 4. It further specifies COSE headers for use with C509 certificates in COSE; see Section 8.16; a TLS certificate type for use with C509 certificates in TLS and QUIC, with or without additional TLS certificate compression; see Section 8.20; and a C509 file format. By extending the TLSA selectors registry to include C509 certificates, this document updates [RFC6698].

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification makes use of the terminology in [RFC2986], [RFC5280], [RFC7228], [RFC8610], and [RFC8949]. When referring to CBOR, this specification always refers to Deterministically Encoded CBOR as specified in Sections 4.2.1 and 4.2.2 of [RFC8949].

3. C509 Certificate

This section specifies the content and encoding of C509 certificates, with the objective of producing a compact representation that supports a large part of [RFC5280] and all of [RFC7925], [IEEE-802.1AR], RPKI [RFC6487], GSMA eUICC [GSMA-eUICC], and CAB Baseline [CAB-TLS] [CAB-Code].

In the CBOR encoding, static fields are elided, elliptic curve points and time values are compressed, OIDs are replaced with short integers or complemented with CBOR OID encoding [RFC9090], and redundant encoding is removed. Combining these techniques significantly reduces certificate size, which is not achievable with general-purpose compression algorithms; see Figure 5.

A C509 certificate can be either a CBOR re-encoding of a DER-encoded X.509 certificate, in which case the signature is calculated on the DER-encoded ASN.1 data in the X.509 certificate, or a natively signed C509 certificate, in which case the signature is calculated directly on the CBOR-encoded data. In both cases, the certificate content adheres to the restrictions given in [RFC5280]. The re-encoding is known to work with DER-encoded certificates, but it might also work with other canonical encodings. The re-encoding does not work for BER-encoded certificates.

In the encoding described below, the elements in arrays are always encoded in the same order as elements of the corresponding SEQUENCE or SET in the DER encoding.

3.1. Message Fields

This section describes the X.509 fields and their CBOR encodings and uses them in the definition of C509 certificates; see Figure 1. While many of [RFC5280] encodings are supported, there are a few instances marked "not supported" for which no alternative is provided and, therefore, no C509 encoding can be generated.

The following Concise Data Definition Language (CDDL) defines the CBOR array C509Certificate and the CBOR Sequence [RFC8742] TBSCertificate. The member names therefore have documentary value only. Applications that do not require a CBOR item MAY represent C509 certificates using the CBOR sequence ~C509Certificate (unwrapped C509Certificate). Examples are given in the appendices; see, for example, Appendix A.1.

```
C509Certificate = [
   TBSCertificate,
   issuerSignatureValue : any,
]

; The elements of the following group are used in a CBOR Sequence:
TBSCertificate = (
   c509CertificateType: int,
   certificateSerialNumber: CertificateSerialNumber,
   issuerSignatureAlgorithm: AlgorithmIdentifier,
   issuer: Name / null,
   validityNotBefore: ~time,
   validityNotAfter: ~time / null,
   subject: Name,
   subjectPublicKeyAlgorithm: AlgorithmIdentifier,
   subjectPublicKey: Defined,
   extensions: Extensions,
)

CertificateSerialNumber = ~biguint

Name = [ * RDNAttribute ] / SpecialText

RDNAttribute = (
    ( attributeType: int, attributeValue: SpecialText ) //
    ( attributeType: ~oid, attributeValue: bytes )
)

AlgorithmIdentifier = int / ~oid /
                      [ algorithm: ~oid, parameters: bytes ]

Extensions = [ * Extension ] / int

Extension = (
    ( extensionID: int, extensionValue: Defined ) //
    ( extensionID: ~oid, extensionValue: bytes / [ bytes ] )
)

SpecialText = text / bytes / tag

Defined = any .ne undefined

tag = #6
```

Figure 1: CDDL for C509Certificate.

C509 certificates are defined in terms of DER-encoded X.509 certificates [RFC5280] as detailed in the following subsections.

3.1.1. version

The ’version’ field is encoded in the ’c509CertificateType’ CBOR int. The field ’c509CertificateType’ also indicates the type of the C509 certificate. Two types are defined in this document: natively signed C509 certificates, following X.509 v3 (c509CertificateType = 2); and CBOR re-encoded X.509 v3 DER certificate (c509CertificateType = 3), see Section 8.2. The number of elements in TBSCertificate is fixed and determined by the type. Additional types may be added in the future.

3.1.2. certificateSerialNumber

The ’certificateSerialNumber’ INTEGER value field is encoded as the unwrapped CBOR unsigned bignum (~biguint) ’CertificateSerialNumber’. Any leading 0x00 byte (to indicate that the number is not negative) is therefore omitted.

3.1.3. signature

The ’signature’ field, containing the signature algorithm including parameters, is encoded as a CBOR int (see Section 8.14) or as an array with an unwrapped CBOR OID tag [RFC9090] optionally followed by the parameters encoded as a CBOR byte string.

3.1.4. issuer

In the general case, the sequence of ’RDNAttribute’ is encoded as a CBOR array consisting of RDNAttribute elements. RelativeDistinguishedName with more than one AttributeTypeAndValue is not supported. Each RDNAttribute is CBOR-encoded as (type, value), either as an (int, SpecialText) pair or as a (~oid, bytes) tuple.

In the former case, the absolute value of the int encodes the attribute type (see Figure 10) and the sign is used to represent the character string type in the X.509 certificate; positive for utf8String, negative for printableString. Attribute values which are always of type IA5String are unambiguously represented using a non-negative int. Examples include emailAddress and domainComponent (see [RFC5280]). In CBOR, all text strings are UTF-8 encoded and in natively signed C509 certificates all CBOR ints SHALL be non-negative. Text strings SHALL still adhere to any [RFC5280] restrictions. The value of the attributes serialNumber and countryName SHALL contain only characters from the 74-character ASCII subset permitted by PrintableString. Additionally, the value of the countryName attribute SHALL have length 2. CBOR encoding is allowed for IA5String (if this is the only allowed type, e.g., emailAddress), printableString and utf8String, whereas the string types teletexString, universalString, and bmpString are not supported.

The text strings are further optimized as follows:

* If the text string has an even length of at least 2 and contains only the symbols '0'-'9' or 'a'-'f', it is encoded as a CBOR byte string.

* If the text string contains an EUI-64 of the form "HH-HH-HH-HH-HH-HH-HH-HH" where each 'H' is one of the symbols '0'-'9' or 'A'-'F', it is encoded as a CBOR tagged MAC address using the CBOR tag 48, see Section 2.4 of [RFC9542]. If of the form "HH-HH-HH-FF-FE-HH-HH-HH", it is encoded as a 48-bit MAC address, otherwise as a 64-bit MAC address. See example in Appendix A.1.

* Otherwise, it is encoded as a CBOR text string.

The final encoding of the attribute value may therefore be text, bytes, or tag, i.e., SpecialText. If Name contains a single 'common name' attribute with attributeType = +1, it is for compactness encoded as just the SpecialText containing the single attribute value.

In natively signed C509 certificates, bytes and tag 48 do not correspond to any predefined text string encoding and may also be used for other attribute types.

If the 'issuer' field is identical to the 'subject' field, e.g., in case of self-signed certificates, then the 'issuer' field MUST be encoded as the CBOR simple value null (0xf6).

3.1.5. validity

The 'notBefore' and 'notAfter' fields are encoded as unwrapped CBOR epoch-based date/time (~time) where the tag content is an unsigned integer. In POSIX time [POSIX], leap seconds are ignored, with a leap second having the same POSIX time as the second before it. Compression of X.509 certificates with the time 23:59:60 UTC is therefore not supported. Note that RFC 5280 mandates encoding of dates through the year 2049 as UTCTime, and later dates as GeneralizedTime. The value "99991231235959Z" (no expiration date) is encoded as the CBOR simple value null.

3.1.6. subject

The 'subject' field is encoded exactly like issuer, except that the CBOR simple value null is not a valid value.

3.1.7. subjectPublicKeyInfo

The 'AlgorithmIdentifier' field including parameters is encoded as the CBOR int 'subjectPublicKeyAlgorithm' (see Section 8.15) or as an array with an unwrapped CBOR OID tag [RFC9090] optionally followed by the parameters encoded as a CBOR byte string.

In general, the 'subjectPublicKey' BIT STRING value field is encoded as a CBOR byte string, but may be encoded as a CBOR item of any type except undefined (see Section 4.4). This specification assumes the BIT STRING has zero unused bits, and the unused bits byte is omitted. For rsaEncryption and id-ecPublicKey, the encoding of subjectPublicKey is further optimized as described in Section 3.2.

3.1.8. issuerUniqueID

Not supported.

3.1.9. subjectUniqueID

Not supported.

3.1.10. extensions

The 'extensions' field is encoded either as a CBOR array or as a CBOR int. An omitted 'extensions' field is encoded as an empty CBOR array.

Each 'extensionID' in the CBOR array is encoded either as a CBOR int (see Section 8.8) or as an unwrapped CBOR OID tag [RFC9090].

* If 'extensionID' is encoded as a CBOR int, it is followed by a CBOR item of any type except undefined (see Section 4.4), and the sign of the int is used to encode if the extension is critical: Critical extensions are encoded with a negative sign and non-critical extensions are encoded with a positive sign. If the CBOR array contains exactly two ints and the absolute value of the first int is 2 (corresponding to keyUsage, see Section 3.3), the CBOR array is omitted and the 'extensions' field is encoded as a single CBOR int with the absolute value of the second int and the sign of the first int.

* If extensionID is encoded as an unwrapped CBOR OID tag, it is followed by the DER-encoded extnValue encoded in the following way:

  - if the extension is non-critical, the extnValue OCTET STRING value field is encoded as a CBOR byte string;

  - if the extension is critical, the extnValue OCTET STRING value field is encoded as a CBOR byte string and further wrapped in a CBOR array consisting of only this element.

The processing of critical and non-critical extensions is specified in Section 4.2 of [RFC5280].

The currently defined extension values for which there is CBOR int encoded 'extensionID' are specified in Section 3.3. The extensions mandated to be supported by [RFC7925] and [IEEE-802.1AR] are given special treatment.

More details about extensions in Section 3.3.

3.1.11. signatureAlgorithm

The 'signatureAlgorithm' field is always the same as the 'signature' field and therefore omitted from the CBOR encoding.

3.1.12. signatureValue

In general, the 'signatureValue' BIT STRING value field is encoded as the CBOR byte string issuerSignatureValue. This specification assumes that the BIT STRING has zero unused bits, and the unused bits byte is omitted. For natively signed C509 certificates, the signatureValue is calculated over the CBOR sequence TBSCertificate. For ECDSA, the encoding of issuerSignatureValue is further optimized as described in Section 3.2.

3.2. Encoding of subjectPublicKey and issuerSignatureValue

3.2.1. Encoding of subjectPublicKey

For RSA public keys (rsaEncryption), the SEQUENCE and INTEGER type and length fields are omitted, and the two INTEGER value fields (modulus, exponent) are encoded as an array of two unwrapped CBOR unsigned bignum (~biguint), i.e., [modulus : ~biguint, exponent : ~biguint]. If the exponent is 65537, the array and the exponent are omitted and subjectPublicKey consists of only the modulus encoded as an unwrapped CBOR unsigned bignum (~biguint).

For elliptic curve public keys in Weierstrass form (id-ecPublicKey), keys may be point-compressed as defined in Section 2.3.3 of [SECG]. Native C509 certificates with Weierstrass form keys use the octets 0x02, 0x03, and 0x04 as defined in [SECG]. If a DER-encoded certificate with an uncompressed public key of type id-ecPublicKey is CBOR-encoded with point compression, then the octet 0xfe is used instead of 0x02 to represent an even y-coordinate, and the octet 0xfd is used instead of 0x03 to represent an odd y-coordinate.

3.2.2. Encoding of issuerSignatureValue

ECDSA signatures are encoded in the same way as in Section 2.1 of [RFC9053]. For example, for P-256, the number of bytes for each integer is 32. The resulting byte string is encoded as a CBOR byte string.

3.3. Encoding of Extensions

The ’extensions’ field is encoded as specified in Section 3.1.10 with further details provided in this section.

For some extensions, the CBOR int encoded extensionID is only supported for commonly used values of the extension. In case of extension values for which the CBOR int encoded extensionID is not supported, the extension MUST be encoded using the unwrapped CBOR OID tag encoded extensionID.

A note on extensionID naming: in existing OID databases, the same OID can be found in versions with and without an ’id-pe’ or ’id-ce’ prefix. We have excluded the prefix for the commonly used extensions defined in [RFC5280] and included them for extensions defined elsewhere.

CBOR encoding of the following extension values is fully supported:

* Subject Key Identifier (subjectKeyIdentifier). In natively signed certificates, KeyIdentifier can, for example, be composed of the leftmost 160 bits of the SHA-256 hash of the CBOR encoded subjectPublicKey. Other methods of generating unique numbers can be used. The extensionValue is encoded as follows:

```
KeyIdentifier = bytes
SubjectKeyIdentifier = KeyIdentifier
```

* Key Usage (keyUsage). The 'KeyUsage' BIT STRING is interpreted as an unsigned integer in network byte order and encoded as a CBOR int. See Section 3.1.10 for special encoding in case keyUsage is the only extension present.

```
KeyUsage = uint
```

* Policy Mappings (policyMappings). extensionValue is encoded as follows:

```
PolicyMappings = [
  + (issuerDomainPolicy: int/~oid, subjectDomainPolicy: int/~oid)
]
```

* Basic Constraints (basicConstraints). If 'cA' = false then extensionValue = -2, if 'cA' = true and 'pathLenConstraint' is not present then extensionValue = -1, and if 'cA' = true and 'pathLenConstraint' is present then extensionValue = pathLenConstraint.

```
BasicConstraints = int
```

* Policy Constraints (policyConstraints). extensionValue is encoded as follows:

```
PolicyConstraints = [
  requireExplicitPolicy: uint / null,
  inhibitPolicyMapping: uint / null,
]
```

* Extended Key Usage (extKeyUsage). extensionValue is encoded as an array of CBOR ints (see Section 8.12), or unwrapped CBOR OID tags [RFC9090], where each int or OID encodes a key usage purpose. If the array contains a single KeyPurposeId, the array is omitted.

```
KeyPurposeId = int / ~oid
ExtKeyUsageSyntax = [ 2* KeyPurposeId ] / KeyPurposeId
```

* Inhibit anyPolicy (inhibitAnyPolicy). extensionValue is encoded as follows:

  InhibitAnyPolicy = uint

* IPAddrBlocks (id-pe-ipAddrBlocks). The X.509 extension IPAddrBlocks is specified in [RFC3779]. The ASN.1 BIT STRING value of IPAddress is converted to a byte sequence defined as:

  unusedBits || value

  where unusedBits is a single octet indicating the number of unused bits in the final octet of the BIT STRING, and value is the sequence of octets containing the BIT STRING value. This byte sequence preserves the exact information contained in the ASN.1 BIT STRING.

  For each IPAddressFamily, the representation is selected as follows:

  - If inherit is present, null SHALL be used.

  - Otherwise, if the byte sequence of any IPAddress (including addressPrefix, and the min and max fields of addressRange) exceeds 8 octets in length, the IPAddressChoice representation SHALL be used.

  - Otherwise, the IntIPAddressChoice representation SHALL be used.

  For IntIPAddressChoice, IntAddressPrefix and the min and max values of IntAddressRange SHALL be encoded as big-endian integers representing the following byte sequence:

  (unusedBits + 1) || value

  The first byte is encoded as (unusedBits + 1) instead of unusedBits in order to guarantee a non-zero value. With the exception of the first IPAddress, each subsequent IPAddress SHALL be encoded as a CBOR integer representing the difference from the previous IPAddress.

  As specified in [RFC3779], the IPAddressFamily element contains an Address Family Identifier (AFI) and, optionally, a Subsequent Address Family Identifier (SAFI). AFIs and SAFIs are defined in [IANA-AFI] and [IANA-SAFI], respectively. The limitations specified in [RFC3779] apply here as well.

```
IntAddressPrefix = int
IntAddressRange  = [ min: int, max: int ]
IntIPAddressOrRange = IntAddressPrefix / IntAddressRange
IntIPAddressChoice  = [ + IntIPAddressOrRange ]

AddressPrefix = bytes
AddressRange  = [ min: bytes, max: bytes ]
IPAddressOrRange = AddressPrefix / AddressRange
IPAddressChoice  = [ + IPAddressOrRange ]

IPAddressFamily = (AFI: uint, SAFI: uint / null,
                   IntIPAddressChoice / IPAddressChoice / null)
IPAddrBlocks = [ + IPAddressFamily ]
```

* IPAddrBlocks v2 (id-pe-ipAddrBlocks-v2). The X.509 extension IPAddrBlocks v2 is specified in [RFC8360]. The extension value is encoded exactly like in the extension "IPAddrBlocks".

* OCSP No Check (id-pkix-ocsp-nocheck). The CBOR encoded extensionValue is the value null.

* TLS Features (id-pe-tlsfeature). The extensionValue is encoded as an array of integers, where each integer represents a TLS extension.

```
TLSFeatures = [* feature: uint]
```

CBOR encoding of the following extension values are partly supported:

* Subject Alternative Name (subjectAltName). If the subject alternative name only contains general names registered in Section 8.13 the extension value can be CBOR encoded. extensionValue is encoded as an array of (int, any) pairs where each pair encodes a general name (see Section 8.13). If subjectAltName contains exactly one dNSName, the array and the int are omitted and extensionValue is the dNSName encoded as a CBOR text string. In addition to the general names defined in [RFC5280], the otherName values with type-id id-on-hardwareModuleName, id-on-SmtpUTF8Mailbox and id-on-MACAddress have been given their own ints; such otherName values are encoded as follows:

  - For id-on-hardwareModuleName, the value is a CBOR array [hwType: ~oid, hwSerialNum: bytes] as specified in [RFC4108].

  - For id-on-SmtpUTF8Mailbox, the value is a CBOR text as specified in [RFC9598].

- For id-on-MACAddress, the value is a CBOR byte string containing 6 octets for EUI-48 and 8 octets for EUI-64 as specified in [I-D.ietf-lamps-macaddress-on].

```
GeneralName = ( GeneralNameType : int, GeneralNameValue : any )
GeneralNames = [ + GeneralName ]
SubjectAltName = GeneralNames / text
```

* Issuer Alternative Name (issuerAltName). extensionValue is encoded exactly like subjectAltName.

```
IssuerAltName  = GeneralNames / text
```

* CRL Distribution Points (cRLDistributionPoints). If all DistributionPoint elements contain the distributionPoint with fullName choice of uniformResourceIdentifier, optional reasons, and optional cRLIssuer with one directoryName, the extension value can be CBOR-encoded. The 'reasons' BIT STRING is interpreted as an unsigned integer in network byte order and encoded as a CBOR uint. If the CRLDistributionPoints consists of only one DistributionPointName, which in turn has only the fullName field of type CBOR text, it is encoded as CBOR text; otherwise it is encoded as a CBOR array.

```
DistributionPointName = [
  fullName:  [ 2 * text ] / text,
  reasons:   uint / null,
  cRLIssuer: Name / null,
]

CRLDistributionPoints = [ + DistributionPointName ] / text
```

* Freshest CRL (freshestCRL). extensionValue is encoded exactly like cRLDistributionPoints.

```
FreshestCRL = CRLDistributionPoints
```

* Authority Information Access (authorityInfoAccess). If all the GeneralNames in authorityInfoAccess are of type uniformResourceIdentifier, the extension value can be CBOR-encoded. Each accessMethod is encoded as a CBOR int (see Section 8.11) or an unwrapped CBOR OID tag [RFC9090]. The uniformResourceIdentifiers are encoded as CBOR text strings.

```
AccessDescription = ( accessMethod: int / ~oid , uri: text )
AuthorityInfoAccessSyntax = [ + AccessDescription ]
```

* Subject Information Access (subjectInfoAccess). Encoded exactly like authorityInfoAccess.

  SubjectInfoAccessSyntax = AuthorityInfoAccessSyntax

* Authority Key Identifier (authorityKeyIdentifier). If the authority key identifier contains all of keyIdentifier, authorityCertIssuer, and authorityCertSerialNumber, or if only keyIdentifier is present, the extension value can be CBOR-encoded. If all three are present, a CBOR array is used. If only keyIdentifier is present, the array is omitted:

```
KeyIdentifierArray = [
  keyIdentifier: KeyIdentifier,
  authorityCertIssuer: GeneralNames,
  authorityCertSerialNumber: CertificateSerialNumber,
]
AuthorityKeyIdentifier = KeyIdentifierArray / KeyIdentifier
```

* Certificate Policies (certificatePolicies). If noticeRef is not used and any explicitText values are encoded as UTF8String, the extension value can be CBOR-encoded. OIDs registered in Section 8.9 are encoded as an int. The policyQualifierId is encoded as a CBOR int (see Section 8.10) or an unwrapped CBOR OID tag [RFC9090].

```
PolicyIdentifier = int / ~oid
PolicyQualifierInfo = (
  policyQualifierId: int / ~oid,
  qualifier: text,
)
CertificatePolicies = [
  + ( PolicyIdentifier, [ * PolicyQualifierInfo ] )
]
```

* Name Constraints (nameConstraints). If the name constraints only contain general names registered in Section 8.13, the extension value can be CBOR-encoded. C509 uses the same additions and restrictions as defined in Section 2.2 of [RFC9549]. Note that the minimum and maximum fields are not used and are therefore omitted. For IPv4 addresses, the iPAddress field MUST contain five octets, and for IPv6 addresses, the field MUST contain 17 octets. In both cases the last octet indicates the number of bits in the prefix. As an example, the address block 192.0.2.0/24 is encoded as C0 00 02 00 18 instead of C0 00 02 00 FF FF FF 00 as in the DER encoding.

```
GeneralSubtrees = [ + GeneralName ]
NameConstraints = [
  permittedSubtrees: GeneralSubtrees / null,
  excludedSubtrees: GeneralSubtrees / null,
]
```

* Subject Directory Attributes (subjectDirectoryAttributes). Encoded as attributes in issuer and subject with the difference that there can be more than one attributeValue.

```
RDNAttributes = (
  ( attributeType: int, attributeValue: [ + SpecialText] ) //
  ( attributeType: ~oid, attributeValue: [+ bytes] )
)
SubjectDirectoryAttributes = [ + RDNAttributes ]
```

* AS Identifiers (id-pe-autonomousSysIds). The X.509 extension AS Identifiers is specified in [RFC3779]. If 'rdi' is not present, the extension value can be CBOR-encoded. Each ASId is encoded as a CBOR uint. With the exception of the first ASId, each subsequent ASId is encoded as the difference from the previous ASId.

```
ASIdOrRange = uint / [min:uint, max:uint]
ASIdentifiers = [ + ASIdOrRange ] / null
```

* AS Identifiers v2 (id-pe-autonomousSysIds-v2). The X.509 extension AS Identifiers v2 is specified in [RFC8360]. The extension value is encoded exactly like in the extension "AS Identifiers".

3.3.1. Example Encoding of Extensions

The examples below use values from Section 8.8, Section 8.12, and Section 8.13:

* A critical basicConstraints ('cA' = true) without pathLenConstraint is encoded as the two CBOR ints -4, -1.

* A non-critical keyUsage with digitalSignature (0), nonRepudiation (1), keyEncipherment (2) and keyAgreement (4) asserted is encoded as the two CBOR ints 2, 23 (2^0 + 2^1 + 2^2 + 2^4 = 23).

* A non-critical extKeyUsage containing id-kp-codeSigning and id-kp-OCSPSigning is encoded as the CBOR int 8 followed by the CBOR array [3, 9].

* A non-critical subjectAltName containing only the dNSName example.com is encoded as the CBOR int 3 followed by the CBOR text string "example.com".

Thus, the extension field of a certificate containing all of the above extensions in the given order would be encoded as the CBOR array [-4, -1, 2, 23, 8, [3, 9], 3, "example.com"].

3.4. C509 COSE Header Parameters

The formatting and processing for c5b, c5c, c5t, and c5u, defined in Table 1 below, are similar to x5bag, x5chain, x5t, x5u defined in [RFC9360] except that the certificates are C509 instead of DER-encoded X.509 and use a COSE_C509 structure instead of COSE_X509.

The COSE_C509 structure used in c5b, c5c, and c5u is defined as:

```
COSE_C509 = C509CertData / [ 2* C509CertData ]
C509CertData = bytes .cborseq C509Certificate
```

C509CertData thus includes the unwrapped CBOR sequence, ~C509Certificate. The byte string encoding includes the length of each certificate, which simplifies parsing. See Appendix A.1.5 for an example.

The COSE_C509 item has media type application/cose-c509-cert, see Section 8.18.1. Different CoAP Content-Formats are defined depending on "usage" = "chain" or not, see Section 8.19. Stored file formats are defined for the cases with/without ("usage" = "chain") with "magic numbers" TBD8/TBD6 using the reserved CBOR tag 55799 and the corresponding Content-Formats TBD15/TBD3, enveloped as described in Section 2.2 of [RFC9277].

The value type of c5t is the COSE_CertHash structure defined in [RFC9360], which contains the hash value of the C509 certificate calculated over ~C509Certificate. Thus, C509CertData contains all data necessary to calculate the thumbprint c5t.

c5u provides an alternative way to identify an untrusted certificate chain by reference with a URI [RFC3986], encoded as a CBOR text string (media type application/cbor and CoAP Content-Format 60). The referenced resource is a COSE_C509 item served with the application/cose-c509-cert media type ("usage" = "chain"), as described above.

As the contents of c5b, c5c, c5t, and c5u are untrusted input, the header parameters can be in either the protected or unprotected header bucket. The trust mechanism MUST process any certificates in the c5b, c5c, and c5u parameters as untrusted input. The presence of a self-signed certificate in the parameter MUST NOT cause the update of the set of trust anchors without appropriate authorization.

Name	Label	Value Type	Description
c5b	24	COSE_C509	An unordered bag of C509 certificates
c5c	25	COSE_C509	An ordered chain of C509 certificates
c5t	22	COSE_CertHash	Hash of a ~C509Certificate
c5u	23	uri	URI pointing to a COSE_C509 containing an ordered chain of certificates

Table 1: C509 COSE Header Parameters

Certificates can also be identified with a ’kid’ header parameter by storing the ’kid’ value and the associated bag or chain in a dictionary.

3.5. C509 COSE Header Algorithm Parameters

This section defines the COSE header parameters used for identifying or transporting the sender’s key for static-static key agreement algorithms corresponding to Section 3 of [RFC9360], see Table 2.

* c5c-sender contains the chain of certificates starting with the sender’s key exchange certificate. The structure is the same as ’c5c’.

* c5t-sender contains the hash value for the sender’s key exchange certificate. The structure is the same as ’c5t’.

* c5u-sender contains a URI for the sender’s key exchange certificate. The structure and processing are the same as ’c5u’.

Name	Algorithm	Label	Type	Description
c5c-sender	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	-30 (suggested)	COSE_C509	An ordered chain of C509 certificates
c5t-sender	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	-31 (suggested)	COSE_CertHash	Hash of a ~C509Certificate
c5u-sender	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	-32 (suggested)	uri	URI pointing to a COSE_C509 containing an ordered chain of certificates

Table 2: Static ECDH Algorithm Values

3.6. Private Key Structures

Certificate management also makes use of data structures including private keys; see, e.g., [RFC7468]. This section defines the following CBOR-encoded structures:

```
C509PrivateKey = [
   C509PrivateKeyType: int,
   subjectPrivateKeyAlgorithm: AlgorithmIdentifier,
   subjectPrivateKey: any,
]
```

The field 'C509PrivateKeyType' indicates the type of the C509 private key. Different types of C509 Private Key Structures can be defined, see Section 8.4. Currently, two types are defined. When C509PrivateKeyType = 0, the subjectPrivateKey is the CBOR byte string encoding of the PrivateKey OCTET STRING value field defined in [RFC5958]. When C509PrivateKeyType = 1, the subjectPrivateKey is a COSE_KEY structure containing a private key as defined in [RFC9052]. Note that COSE_KEY might not be possible to use with all algorithms that have a C509 AlgorithmIdentifier defined.

The C509PrivateKey item is served with the application/cose-c509-privkey media type, see Section 8.18.4, with corresponding CoAP Content-Format defined in Section 8.19. A stored file format is defined with "magic number" TBD12 using the reserved CBOR tag 55799 and the Content-Format TBD10, enveloped as described in Section 2.2 of [RFC9277].

```
C509PEM = [
   C509PrivateKey,
   COSE_C509 / null,
]
```

The C509PEM item is served with the application/cose-c509-pem media type, see Section 8.18.5, with corresponding CoAP Content-Format defined in Section 8.19. A stored file format is defined with "magic number" TBD13 using the reserved CBOR tag 55799 and the Content-Format TBD11, enveloped as described in Section 2.2 of [RFC9277].

3.7. Deterministic Encoding

In some use cases it is desirable to be able to specify a unique C509 representation of a given X.509 certificate.

Although this specification requires the use of Deterministically Encoded CBOR (see Section 2), it is still possible to represent certain X.509 certificate fields in different ways. This is a consequence of the extensibility of the C509 format, in which new encodings can be defined, for example, to optimize extensions for which no special CBOR encoding has previously been defined.

Where both a specific and a generic CBOR encoding are supported, the specific CBOR encoding MUST be used. For example, when a specific CBOR encoding of an extension is defined in Section 3.3 and the C509 Extensions Registry, that specific encoding MUST be used. In particular, when a specific otherName encoding is available, identified by a negative integer value in the C509 General Names Registry, it MUST be used.

Native C509 certificates MUST use only specific CBOR-encoded fields. However, when decoding non-native C509 certificates, the decoder may need to support, for example, the (extensionID: ~oid, extensionValue: bytes / [bytes]) encoding of an extension for which an (extensionID: int, extensionValue: Defined) encoding exists. One reason is that the certificate might have been issued before the specific CBOR extension was registered.

3.8. C509 Name in TLS and DTLS

In TLS and DTLS, the subject of a trusted authority may be sent to the peer to help it select the certificate chain, as in the CertificateAuthoritiesExtension in [RFC8446], in the certificate_authorities field of CertificateRequest in [RFC5246], or in the TrustedAuthorities in [RFC6066]. For such usage in TLS and DTLS, the C509 name is wrapped in a distinguished name [X.501] with exactly one RelativeDistinguishedName, which in turn contains exactly one AttributeTypeAndValue with the attribute C509Name. The attribute value is the raw byte string of the encoded C509 Name as specified in Section 3.1.6.

The attribute for C509 Name has the following structure:

```
id-rdna-c509Name OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 25 TBD30 }

c509Name ATTRIBUTE ::= {
   WITH SYNTAX C509Name
   SINGLE VALUE TRUE
   ID id-rdna-c509Name }

C509Name ::= OCTET STRING
```

4. C509 Certification Request

This section defines the format of a C509 Certification Request based on [RFC2986]. It reuses the encodings of C509 certificates defined in Section 3. A Certification Request is commonly referred to as a Certificate Signing Request (CSR).

The CDDL for the C509 Certification Request is shown in Figure 2. The fields have the same encoding as the corresponding fields of the C509 Certificate, see Section 3.1.

```
C509CertificationRequest = [
   TBSCertificationRequest,
   subjectSignatureValue: any,
]

; The elements of the following group are used in a CBOR Sequence:
TBSCertificationRequest = (
   c509CertificationRequestType: int,
   subjectSignatureAlgorithm: AlgorithmIdentifier,
   subject: Name,
   subjectPublicKeyAlgorithm: AlgorithmIdentifier,
   subjectPublicKey: Defined,
   attributes: CRAttributes,
)

CRAttributes = [ * CRAttribute ]

CRAttribute = (( attributeType: int, attributeValue: Defined ) //
               ( attributeType: ~oid, attributeValue: bytes ))
```

Figure 2: CDDL for C509CertificationRequest.

After verifying subjectSignatureValue, the Certification Authority (CA) MAY transform the C509CertificationRequest into a [RFC2986] CertificationRequestInfo for compatibility with existing procedures and implementations.

The media type of C509CertificationRequest is application/cose-c509-pkcs10, see Section 8.18.2, with corresponding CoAP Content-Format defined in Section 8.19. The "magic number" TBD9 is defined using the reserved CBOR tag 55799 and the Content-Format TBD4, enveloped as described in Section 2.2 of [RFC9277].

4.1. Certification Request Types

Two types of C509 Certification Requests are defined. Both use the same CBOR encoding and differ only in what is being signed; see Section 8.3. A C509 Certification Request is either an invertible CBOR re-encoding of a DER-encoded certification request [RFC2986] or a natively signed request in which the signature is calculated over the CBOR encoding instead of the DER encoding.

* c509CertificationRequestType = 2. This type indicates that the C509 Certification Request is natively signed, i.e., that subjectSignatureValue contains the signature over the CBOR Sequence TBSCertificationRequest; see Figure 2. This encoding removes the need for ASN.1 and DER parsing and for re-encoding by the requesting party.

* c509CertificationRequestType = 3. This type indicates that the C509 Certification Request is a CBOR re-encoded [RFC2986] certification request, as defined in Section 4. This encoding is backward compatible with legacy RFC 2986 certification requests and reduces transport overhead.

The type of certificate issued in response to the request is determined by the application. For C509, the default is c509CertificateType = c509CertificationRequestType.

An implementation MAY only support certain values of c509CertificationRequestType.

4.2. Subject Signature Algorithm

subjectSignatureAlgorithm can be a signature algorithm or a non-signature proof-of-possession algorithm, for example, as defined in [RFC6955]. In the case of [RFC6955], the signature is replaced by a MAC and requires a public Diffie-Hellman key of the verifier to be distributed out of band. Both signature algorithms and non-signature proof-of-possession algorithms are listed in the C509 Signature Algorithms Registry; see Section 8.14. The non-signature proof-of-possession algorithms with SHA-2 and HMAC-SHA2 (see values 14-16 in Section 8.14) require a signature value with syntax DhSigStatic, defined as follows:

```
DhSigStatic = MessageDigest / DhSigStaticType

MessageDigest = bytes

DhSigStaticType = [
  issuer: Name,
  certificateSerialNumber: CertificateSerialNumber,
  hashValue: MessageDigest,
]
```

Note that a key agreement key pair may be used with a signature algorithm in a certification request, see Appendix A.1.3.

4.3. Certification Request Attributes

The 'attributes' field specifies the attributes contained in a certification request. The 'attributes' field with no GeneralAttribute SHALL be encoded as an empty CBOR array.

The remainder of this section specifies CBOR-encoded attributes for Certification Requests.

4.3.1. Extension Request

The X.509 attribute "Extension Request" is defined in [RFC2985]. The 'attributeValue' field has type Extensions as in Section 3.1. An empty CBOR array indicates no extensions.

4.3.2. Challenge Password

The X.509 attribute "Challenge Password" is defined in [RFC2985]. The 'attributeValue' field has type ChallengePassword. A UTF8 String is encoded as CBOR text, and a Printable String is tagged with number 121 (alternative 0 as defined in [IANA-CBOR-TAGS]). All other string types are not supported. For certification request type 2, only UTF8 String is allowed.

```
ChallengePassword = text / #6.121(text)
```

4.3.3. Private Key Possession Statement

The X.509 attribute "Statement of Possession of a Private Key" is defined in [RFC9883]. The 'attributeValue' field has type PrivateKeyPossessionStatement.

```
PrivateKeyPossessionStatement = [
  issuer: Name,
  certificateSerialNumber: CertificateSerialNumber,
  cert: C509Certificate / null,
]
```

4.4. Certification Request Template

Enrollment over Secure Transport (EST, [RFC7030]) defines, and [RFC9908] clarifies, how an EST server can specify what it expects the EST client to include in a subsequent Certification Request. Alternatively to the unstructured mechanism specified in [RFC7030], Appendix B of [RFC8295] describes an approach using a Certification Request Template in response to a GET /csrattrs request by the EST client. The EST server thus returns a Certification Request-like object with various fields filled out, and other fields waiting to be

filled in and a signature to be added by the EST client.

The approach of [RFC8295] is also followed for C509. The C509CertificationRequestTemplate is based on TBSCertificationRequest of the C509CertificationRequest, see Figure 2, but excludes the subjectSignatureValue field from the template since that needs no further specification.

The C509 Certification Request Template is shown in Figure 3.

```
C509CertificationRequestTemplate = [
   c509CertificationRequestTemplateType: int,
   c509CertificationRequestType: [+ int] / undefined,
   subjectSignatureAlgorithm: [+ AlgorithmIdentifier] / undefined,
   subject: NameTemplate / undefined,
   subjectPublicKeyAlgorithm: [+ AlgorithmIdentifier] / undefined,
   subjectPublicKey: undefined,
   extensionsRequest: ExtensionsTemplate / undefined,
]

NameTemplate = [ * RDNAttributeTemplate ]

RDNAttributeTemplate = (
    ( attributeType: uint, minOccurs: uint, maxOccurs: uint,
      attributeValue: SpecialText / undefined ) //
    ( attributeType: ~oid, minOccurs: uint, maxOccurs: uint,
      attributeValue: bytes / undefined )
)

ExtensionsTemplate = [ * ExtensionTemplate ]

ExtensionTemplate = (
    ( extensionID: uint, optional: bool, extensionValue: any ) //
    ( extensionID: ~oid, optional: bool,
    extensionValue: bytes / undefined )
)
```

Figure 3: CDDL for C509CertificationRequestTemplate.

Except as specified in this section, the fields have the same encoding as the corresponding fields of the TBSCertificationRequest, see Figure 2. The specification of the template makes use of the CBOR simple value undefined (0xf7) to indicate fields to fill in. Consistent with this rule, note that the subjectPublicKey field always has the value undefined in the template.

Different types of Certification Request Templates can be defined (see Section 8.5), distinguished by the c509CertificationRequestTemplateType integer. Each type may have its own CDDL structure.

The presence of a Defined (non-undefined) value in a C509CertificationRequestTemplate indicates that the server expects the client to use that value in the certification request. If multiple AlgorithmIdentifier or c509CertificationRequestType values are present, the server expects the client to select one of them for use in the Certification Request. The presence of an undefined value indicates that the client is expected to provide an appropriate value for that field. For example, if the server includes a subjectAltName with a GeneralNameType iPAddress and a GeneralNameValue empty byte string, this means that the client SHOULD fill in a corresponding GeneralNameValue.

For RDNAttributeTemplate, the minOccurs and maxOccurs fields specify the minimal and maximal occurrences of attributes of the given attributeType; maximal shall not be less than minimal, and maximal shall be positive. Negative attributeType is not allowed.

For ExtensionTemplate, the field "optional" specifies whether an extension of the given extensionID is optional. Negative extensionID is not allowed.

The media type of C509CertificationRequestTemplate is application/cose-c509-crtemplate, see Section 8.18.3, with corresponding CoAP Content-Format defined in Section 8.19. The "magic number" TBD18 is defined using the reserved CBOR tag 55799 and the Content-Format TBD19, enveloped as described in Section 2.2 of [RFC9277].

5. C509 Processing and Certificate Issuance

It is straightforward to integrate the C509 format into legacy X.509 processing during certificate issuance. C509 processing can be performed as an isolated function of the CA, or as a separate function trusted by the CA.

The Certification Request format defined in Section 4 follows the PKCS#10 format to enable a direct mapping to the certification request information, see Section 4.1 of [RFC2986]. The CA can make use of a Certification Request Template defined in Section 4.4, for simplified configuration.

When a certification request is received, the CA, or function trusted by the CA, needs to perform some limited C509 processing and verify the proof-of-possession corresponding to the public key, before normal certificate generation can take place.

In the reverse direction, in case c509CertificateType = 3 was requested, a separate C509 processing function can perform the conversion from a generated X.509 certificate to C509 as a bump-in-the-wire. In case c509CertificateType = 2 was requested, the C509 processing needs to be performed before signing the certificate, in which case a tighter integration with the CA may be needed.

6. Operational Considerations

6.1. Legacy Considerations

C509 certificates can be deployed with legacy X.509 certificates and CA infrastructure. An existing CA can continue to use its existing procedures and code for PKCS#10, and DER-encoded X.509 and only implement C509 as a thin processing layer on top. When receiving a C509 Certification Request, the CA transforms it into a DER-encoded CertificationRequestInfo [RFC2986] and uses that with existing processes and code to produce an RFC 5280 DER-encoded X.509 certificate. The DER-encoded X.509 is then transformed into a C509 certificate. At any later point, the C509 certificate can be used to recreate the original X.509 data structure needed to verify the signature.

For protocols like TLS/DTLS 1.2, where certificates are sent unencrypted, the actual encoding and compression can be done at different locations depending on the deployment setting. For example, the mapping between C509 certificate and standard X.509 certificate can take place in a 6LoWPAN border gateway, which allows the server side to stay unmodified. This case gives the advantage of the low overhead of a C509 certificate over constrained wireless links. The conversion to X.509 within a constrained IoT device will incur a computational overhead. However, measured in energy, this is likely to be negligible compared to the reduced communication overhead.

For the setting with constrained server and server-only authentication, the server only needs to be provisioned with the C509 certificate and does not perform the conversion to X.509. This option is viable when client authentication can be asserted by other means.

For protocols like IKEv2, TLS/DTLS 1.3, and EDHOC, where certificates are encrypted, the proposed encoding needs to be done fully end-to-end, through adding the encoding/decoding functionality to the server.

6.2. Expected Certificate Sizes

The CBOR encoding of the sample certificate chains given in Appendix A results in the numbers shown in Figures 4 and 5. COSE_X509 is defined in [RFC9360] and COSE_C509 is defined in Section 8.16. After [RFC7925] profiling, most duplicated information has been removed, and the remaining text strings are minimal in size. Therefore, the further size reduction reached with general compression mechanisms such as Brotli [RFC7932] will be small, mainly corresponding to making the ASN.1 encoding more compact. CBOR encoding can however significantly compress RFC 7925 profiled certificates. In the examples with HTTPS certificate chains (www.ietf.org (ECDSA) and cabforum.org (RSA)) both C509 and Brotli perform well complementing each other. C509 uses dedicated information to compress individual certificates, while Brotli can compress duplicate information in the entire chain. Note that C509 certificates of type 2 and 3 have the same size. For Brotli, the Rust crate Brotli 3.3.0 was used with compression level 11 and window size 22.

In the examples using FN-DSA and ML-DSA certificate chains, the largest portion of the certificate size consists of the public keys and signatures, which are essentially random. As a result, both Brotli and C509 achieve only very limited size reduction. However, C509 still performs slightly better.

Description (number of certs)	COSE_X509	COSE_C509
RFC 7925 profiled IoT Certificate (1)	319	142
RPKI Certificate (1)	20981	11523
ECDSA HTTPS Certificate Chain (2)	1644	1012
RSA HTTPS Certificate Chain (2)	2909	2240
FN-DSA-512 HTTPS Certificate Chain (2)	4417	3897
ML-DSA-65 HTTPS Certificate Chain (2)	11863	11318

Figure 4: Comparing Sizes of Certificate Chains in COSE. Number of bytes (length of certificate chain).

Description (number of certs)	X.509	X.509 + Brotli	C509	C509 + Brotli
RFC 7925 profiled IoT Certificate (1)	325	317	149	158
RPKI Certificate (1)	20987	9109	11529	7020
ECDSA HTTPS Certificate Chain (2)	1651	1181	1019	930
RSA HTTPS Certificate Chain (2)	2656	2195	2071	1913
FN-DSA-512 HTTPS Certificate Chain (2)	4437	4026	3917	3776
ML-DSA-65 HTTPS Certificate Chain (2)	11869	11420	11325	11148

Figure 5: Comparing Sizes of Certificate Chains with TLS 1.3. Number of bytes (length of certificate chain). X.509 and C509 are Certificate messages. X.509 + Brotli and C509 + Brotli are CompressedCertificate messages.

7. Security Considerations

The CBOR encoding of X.509 certificates does not change the security assumptions needed when deploying standard X.509 certificates. The same security procedures applies as for X.509. For example the same certificate path validation as defined in Section 6 of [RFC5280] MUST be performed on C509 certificates before they can be considered trusted. The security considerations of [RFC5280] apply.

The use of natively signed C509 certificates removes the need for ASN.1 encoding, which is a rich source of security vulnerabilities.

Conversion between the certificate formats can be made in constant time to reduce risk of information leakage through side channels.

The mechanism in this document does not reveal any additional information compared to X.509. Because of the difference in size, it will be possible to detect that this profile is used. The gateway solution described in Section 6.1 requires unencrypted certificates which may violate identity protection and is not recommended.

Any issues with decoding or parsing a C509 certificate should be handled exactly as how such errors would be handled for the corresponding X.509 certificate. For example, a non-critical extension MAY be ignored if it is not recognized, see Section 4.2 of [RFC5280].

As stated in Section 3.4, the contents of the COSE Header Parameters c5b, c5c, c5t, c5u is untrusted input that potentially may be verified using existing trust anchors or other trust establishment mechanism out of scope of this document. Similar security considerations as x5bag, x5chain, x5t and x5u apply, see [RFC9360]. Security considerations of the COSE protected and unprotected headers are discussed in [RFC9052].

The specification makes no recommendations on the use of algorithms, paddings, or other security constructs applied in the encoding of a certificate. In particular, an IANA registration does not imply a recommendation. For example, some deprecated algorithms are assigned code points only for backward compatibility to enable CBOR encoding of existing certificates.

8. IANA Considerations

This document creates several new registries in the new registry group "CBOR Encoded X.509 (C509)". For all items, the ’Reference’ field points to this document.

Editor’s note: Add informative reference to the newly created IANA registries and updated existing registries.

8.1. Designated Expert Guidance

Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. Experts should take into account the expected usage of entries when approving point assignment. The length of the encoded value should be weighed against the number of code points left that encode to that size and how constrained the systems it will be used on are. Values in the interval [-24, 23] have a 1-byte encoding, other values in the interval [-256, 255] have a 2-byte encoding, and the remaining values in the interval [-65536,

65535] have a 3-byte encoding.

All assignments according to "IETF Review with Expert Review" are made on an "IETF Review" basis per Section 4.8 of [RFC8126] with "Expert Review" additionally required per Section 4.5 of [RFC8126]. The procedure for early IANA allocation of "standards track code points" defined in [RFC7120] also applies. When such a procedure is used, IANA will ask the designated expert(s) to approve the early allocation before registration. In addition, working group chairs are encouraged to consult the expert(s) early during the process outlined in Section 3.1 of [RFC7120].

8.2. C509 Certificate Types Registry

IANA has created a new registry titled "C509 Certificate Types" under the registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Description, and Reference, where Value is an integer, and the other columns are text strings. It is mandatory to specify content in all columns. For values in the interval [-24, 23], the registration procedure is "IETF Review with Expert Review". For all other values, the registration procedure is "Expert Review". The initial contents of the registry are (see Section 3.1.1):

Value	Description
0	Reserved
1	Reserved
2	Natively Signed C509 Certificate
3	CBOR Re-encoded X.509 v3 Certificate

Figure 6: C509 Certificate Types

8.3. C509 Certification Request Types Registry

IANA has created a new registry titled "C509 Certification Request Types" under the new registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Description, and Reference, where Value is an integer, and the other columns are text strings. All columns are mandatory. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". For all other values the registration procedure is "Expert Review". The initial contents of the registry are:

Value	Description
0	Reserved
1	Reserved
2	Natively Signed C509 Certification Request.
3	CBOR re-encoding of RFC 2986 certification request.

Figure 7: C509 Certification Request Types

8.4. C509 Private Key Types Registry

IANA has created a new registry titled "C509 Private Key Types" in the new registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Comments, subjectPrivateKey, and Reference, where Value is an integer, and the other columns are text strings. The subjectPrivateKey describes the encoding of the subject private key, see Section 3.6. All columns are mandatory. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". For all other values the registration procedure is "Expert Review". The initial contents of the registry are:

Value	Private Key
0	Comments: Asymmetric Key Package (RFC 5958) subjectPrivateKey: PrivateKey OCTET STRING encoded as CBOR byte string
1	Comments: COSE Key Object (RFC 9052) subjectPrivateKey: COSE_Key containing a private key

Figure 8: C509 Private Key Types

8.5. C509 Certification Request Templates Types Registry

IANA has created a new registry titled "C509 Certification Request Templates Types" under the new registry group "CBOR Encoded X.509 (C509)". The columns of the registry are Value, Description, and Reference, where Value is an integer, and the other columns are text strings. All columns are mandatory. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". For all other values the registration procedure is "Expert

Review". The initial contents of the registry are:

Value	Description
0	Simple C509 Certification Request Template

Figure 9: C509 Certification Request Templates Types

8.6. C509 RDN Attributes Registry

IANA has created a new registry titled "C509 RDN Attributes" in the new registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Identifiers, OID, DER, Comments and Reference, where Value is a non-negative integer, and the other columns are text strings. Name and Identifiers are informal descriptions. The fields Name, OID, and DER are mandatory. For RDN Attributes specified only for CBOR encoded certificates where no OID is defined, the OID and DER fields are marked "N/A". If there is an OID defined, the OID is given in dotted decimal representation, and the DER column contains the hex string of the DER-encoded OID [X.690]. If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the RDN Attribute is described. For values in the interval [0, 23] the registration procedure is "IETF Review with Expert Review". Values 32768 are reserved for Private Use. For all other values the registration procedure is "Expert Review".

The initial contents of the registry are:

Value	RDN Attribute	
0	Name:	Email Address
	Identifiers:	emailAddress, e-mailAddress
	OID:	1.2.840.113549.1.9.1
	DER:	06 09 2A 86 48 86 F7 0D 01 09 01
	Comments:	RFC 2985
1	Name:	Common Name
	Identifiers:	commonName, cn
	OID:	2.5.4.3
	DER:	06 03 55 04 03
	Comments:	X.520
2	Name:	Surname
	Identifiers:	surname, sn

	OID:	2.5.4.4
	DER:	06 03 55 04 04
	Comments:	X.520
3	Name:	Serial Number
	Identifiers:	serialNumber
	OID:	2.5.4.5
	DER:	06 03 55 04 05
	Comments:	X.520
4	Name:	Country
	Identifiers:	countryName, c
	OID:	2.5.4.6
	DER:	06 03 55 04 06
	Comments:	X.520
5	Name:	Locality
	Identifiers:	localityName, locality, l
	OID:	2.5.4.7
	DER:	06 03 55 04 07
	Comments:	X.520
6	Name:	State or Province
	Identifiers:	stateOrProvinceName, st
	OID:	2.5.4.8
	DER:	06 03 55 04 08
	Comments:	X.520
7	Name:	Street Address
	Identifiers:	streetAddress, street
	OID:	2.5.4.9
	DER:	06 03 55 04 09
	Comments:	X.520
8	Name:	Organization
	Identifiers:	organizationName, o
	OID:	2.5.4.10
	DER:	06 03 55 04 0A
	Comments:	X.520
9	Name:	Organizational Unit
	Identifiers:	organizationalUnitName, ou
	OID:	2.5.4.11
	DER:	06 03 55 04 0B
	Comments:	X.520
10	Name:	Title
	Identifiers:	title

	OID:	2.5.4.12
	DER:	06 03 55 04 0C
	Comments:	X.520
11	Name:	Business Category
	Identifiers:	businessCategory
	OID:	2.5.4.15
	DER:	06 03 55 04 0F
	Comments:	X.520
12	Name:	Postal Code
	Identifiers:	postalCode
	OID:	2.5.4.17
	DER:	06 03 55 04 11
	Comments:	X.520
13	Name:	Given Name
	Identifiers:	givenName
	OID:	2.5.4.42
	DER:	06 03 55 04 2A
	Comments:	X.520
14	Name:	Initials
	Identifiers:	initials
	OID:	2.5.4.43
	DER:	06 03 55 04 2B
	Comments:	X.520
15	Name:	Generation Qualifier
	Identifiers:	generationQualifier
	OID:	2.5.4.44
	DER:	06 03 55 04 2C
	Comments:	X.520
16	Name:	DN Qualifier
	Identifiers:	dnQualifier
	OID:	2.5.4.46
	DER:	06 03 55 04 2E
	Comments:	X.520
17	Name:	Pseudonym
	Identifiers:	pseudonym
	OID:	2.5.4.65
	DER:	06 03 55 04 41
	Comments:	X.520
18	Name:	Organization Identifier
	Identifiers:	organizationIdentifier

	OID:	2.5.4.97
	DER:	06 03 55 04 61
	Comments:	X.520
19	Name:	Jurisdiction Locality Name
	Identifiers:	jurisdictionLocalityName
	OID:	1.3.6.1.4.1.311.60.2.1.1
	DER:	06 0B 2B 06 01 04 01 82 37 3C 02 01 01
	Comments:	Proprietary Microsoft Attribute
20	Name:	Jurisdiction State or Province
	Identifiers:	jurisdictionStateOrProvinceName
	OID:	1.3.6.1.4.1.311.60.2.1.2
	DER:	06 0B 2B 06 01 04 01 82 37 3C 02 01 02
	Comments:	Proprietary Microsoft Attribute
21	Name:	Jurisdiction Country Name
	Identifiers:	jurisdictionCountryName
	OID:	1.3.6.1.4.1.311.60.2.1.3
	DER:	06 0B 2B 06 01 04 01 82 37 3C 02 01 03
	Comments:	Proprietary Microsoft Attribute
22	Name:	Domain Component
	Identifiers:	domainComponent, dc
	OID:	0.9.2342.19200300.100.1.25
	DER:	06 0A 09 92 26 89 93 F2 2C 64 01 19
	Comments:	RFC 4524
25	Name:	Name
	Identifiers:	name
	OID:	2.5.4.41
	DER:	06 03 55 04 29
	Comments:	X.520
26	Name:	Telephone Number
	Identifiers:	telephoneNumber
	OID:	2.5.4.20
	DER:	06 03 55 04 14
	Comments:	X.520
27	Name:	Directory Management Domain Name
	Identifiers:	dmdName
	OID:	2.5.4.54
	DER:	06 03 55 04 36
	Comments:	X.520
28	Name:	userid
	Identifiers:	uid

	OID:	0.9.2342.19200300.100.1.1
	DER:	06 0A 09 92 26 89 93 F2 2C 64 01 01
	Comments:	RFC 4524
29	Name:	Unstructured Name
	Identifiers:	unstructuredName
	OID:	1.2.840.113549.1.9.2
	DER:	06 09 2A 86 48 86 F7 0D 01 09 02
	Comments:	RFC 2985
30	Name:	Unstructured Address
	Identifiers:	unstructuredAddress
	OID:	1.2.840.113549.1.9.8
	DER:	06 0A 2A 86 48 86 F7 0D 01 09 08
	Comments:	RFC 2985

Figure 10: C509 RDN Attributes

8.7. C509 CR Attributes Registry

IANA has created a new registry titled "C509 CR Attributes" under the registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Identifiers, OID, DER, Comments, attributeValue, and Reference, where Value is an integer, and the other columns are text strings. Name and Identifiers are informal descriptions. The fields Name, OID, and DER are mandatory. For CR Attributes specified only for CBOR encoded certificates where no OID is defined, the OID and DER fields are marked "N/A". If OID is present, the OID is given in dotted decimal representation, and the DER column contains the hex string of the DER-encoded OID [X.690]. If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the CR Attribute is described. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". Values 32768 are reserved for Private Use. For all other values the registration procedure is "Expert Review".

The initial contents of the registry are:

Value	CR Attribute	
0	Name:	Extension Request
	Identifiers:	extensionRequest
	OID:	1.2.840.113549.1.9.14
	DER:	06 09 2A 86 48 86 F7 0D 01 09 0E
	Comments:	RFC 2985
	attributeValue:	Extensions
1	Name:	Challenge Password
	Identifiers:	challengePassword
	OID:	1.2.840.113549.1.9.7
	DER:	06 09 2A 86 48 86 F7 0D 01 09 07
	Comments:	RFC 2985
	attributeValue:	ChallengePassword
2	Name:	Private Key Possession Statement
	Identifiers:	privateKeyPossessionStatement
	OID:	1.3.6.1.4.1.22112.2.1
	DER:	06 0A 2B 06 01 04 01 81 AC 60 02 01
	Comments:	RFC 9883
	attributeValue:	PrivateKeyPossessionStatement

Figure 11: C509 CRAttributes

8.8. C509 Extensions Registry

IANA has created a new registry titled "C509 Extensions" under the new registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Identifiers, OID, DER, Comments, extensionValue, and Reference, where Value is a positive integer, and the other columns are text strings. The fields Name, OID, DER, and extensionValue are mandatory. For all other values the registration procedure is "Expert Review". For Extensions specified only for CBOR encoded certificates where no OID is defined, the OID and DER fields are marked "N/A". If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the Extension is described. For values in the interval [1, 23] the registration procedure is "IETF Review with Expert Review". Values 32768 are reserved for Private Use.

The initial contents of the registry are:

Value	Extension	
1	Name:	Subject Key Identifier
	Identifiers:	subjectKeyIdentifier
	OID:	2.5.29.14
	DER:	06 03 55 1D 0E
	Comments:	RFC 5280
	extensionValue:	SubjectKeyIdentifier
2	Name:	Key Usage
	Identifiers:	keyUsage
	OID:	2.5.29.15
	DER:	06 03 55 1D 0F
	Comments:	RFC 5280
	extensionValue:	KeyUsage
3	Name:	Subject Alternative Name
	Identifiers:	subjectAltName
	OID:	2.5.29.17
	DER:	06 03 55 1D 11
	Comments:	RFC 5280
	extensionValue:	SubjectAltName
4	Name:	Basic Constraints
	Identifiers:	basicConstraints
	OID:	2.5.29.19
	DER:	06 03 55 1D 13
	Comments:	RFC 5280
	extensionValue:	BasicConstraints
5	Name:	CRL Distribution Points
	Identifiers:	cRLDistributionPoints
	OID:	2.5.29.31
	DER:	06 03 55 1D 1F
	Comments:	RFC 5280
	extensionValue:	CRLDistributionPoints
6	Name:	Certificate Policies
	Identifiers:	certificatePolicies
	OID:	2.5.29.32
	DER:	06 03 55 1D 20
	Comments:	RFC 5280
	extensionValue:	CertificatePolicies
7	Name:	Authority Key Identifier
	Identifiers:	authorityKeyIdentifier
	OID:	2.5.29.35

	DER:	06 03 55 1D 23
	Comments:	RFC 5280
	extensionValue:	AuthorityKeyIdentifier
8	Name:	Extended Key Usage
	Identifiers:	extKeyUsage
	OID:	2.5.29.37
	DER:	06 03 55 1D 25
	Comments:	RFC 5280
	extensionValue:	ExtKeyUsageSyntax
9	Name:	Authority Information Access
	Identifiers:	authorityInfoAccess
	OID:	1.3.6.1.5.5.7.1.1
	DER:	06 08 2B 06 01 05 05 07 01 01
	Comments:	RFC 5280
	extensionValue:	AuthorityInfoAccessSyntax
24	Name:	Subject Directory Attributes
	Identifiers:	subjectDirectoryAttributes
	OID:	2.5.29.9
	DER:	06 03 55 1D 09
	Comments:	RFC 5280
	extensionValue:	SubjectDirectoryAttributes
25	Name:	Issuer Alternative Name
	Identifiers:	issuerAltName
	OID:	2.5.29.18
	DER:	06 03 55 1D 12
	Comments:	RFC 5280
	extensionValue:	IssuerAltName
26	Name:	Name Constraints
	Identifiers:	nameConstraints
	OID:	2.5.29.30
	DER:	06 03 55 1D 1E
	Comments:	RFC 9549
	extensionValue:	NameConstraints
27	Name:	Policy Mappings
	Identifiers:	policyMappings
	OID:	2.5.29.33
	DER:	06 03 55 1D 21
	Comments:	RFC 5280
	extensionValue:	PolicyMappings
28	Name:	Policy Constraints
	Identifiers:	policyConstraints

	OID:	2.5.29.36
	DER:	06 03 55 1D 24
	Comments:	RFC 5280
	extensionValue:	PolicyConstraints
29	Name:	Freshest CRL
	Identifiers:	freshestCRL
	OID:	2.5.29.46
	DER:	06 03 55 1D 2E
	Comments:	RFC 5280
	extensionValue:	FreshestCRL
30	Name:	Inhibit anyPolicy
	Identifiers:	inhibitAnyPolicy
	OID:	2.5.29.54
	DER:	06 03 55 1D 36
	Comments:	RFC 5280
	extensionValue:	InhibitAnyPolicy
31	Name:	Subject Information Access
	Identifiers:	subjectInfoAccess
	OID:	1.3.6.1.5.5.7.1.11
	DER:	06 08 2B 06 01 05 05 07 01 0B
	Comments:	RFC 5280
	extensionValue:	SubjectInfoAccessSyntax
32	Name:	IPAddrBlocks
	Identifiers:	id-pe-ipAddrBlocks
	OID:	1.3.6.1.5.5.7.1.7
	DER:	06 08 2B 06 01 05 05 07 01 07
	Comments:	RFC 3779
	extensionValue:	IPAddrBlocks
33	Name:	AS Identifiers
	Identifiers:	id-pe-autonomousSysIds
	OID:	1.3.6.1.5.5.7.1.8
	DER:	06 08 2B 06 01 05 05 07 01 08
	Comments:	RFC 3779
	extensionValue:	ASIdentifiers
34	Name:	IPAddrBlocks v2
	Identifiers:	id-pe-ipAddrBlocks-v2
	OID:	1.3.6.1.5.5.7.1.28
	DER:	06 08 2B 06 01 05 05 07 01 1C
	Comments:	RFC 8360
	extensionValue:	IPAddrBlocks
35	Name:	AS Identifiers v2

		Identifiers:	id-pe-autonomousSysIds-v2
		OID:	1.3.6.1.5.5.7.1.29
		DER:	06 08 2B 06 01 05 05 07 01 1D
		Comments:	RFC 8360
		extensionValue:	ASIdentifiers
36		Name:	OCSP No Check
		Identifiers:	id-pkix-ocsp-nocheck
		OID:	1.3.6.1.5.5.7.48.1.5
		DER:	06 09 2B 06 01 05 05 07 30 01 05
		Comments:	RFC 6960
		extensionValue:	null
38		Name:	TLS Features
		Identifiers:	id-pe-tlsfeature
		OID:	1.3.6.1.5.5.7.1.24
		DER:	06 08 2B 06 01 05 05 07 01 18
		Comments:	RFC 7633
		extensionValue:	TLSFeatures

Figure 12: C509 Extensions

8.9. C509 Certificate Policies Registry

IANA has created a new registry titled "C509 Certificate Policies" under the registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Identifiers, OID, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. The fields Name, OID, and DER are mandatory. For all other values the registration procedure is "Expert Review". For Certificate Policies specified only for CBOR encoded certificates where no OID is defined, the OID and DER fields are marked "N/A". If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the Certificate Policy is described. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". Values 32768 are reserved for Private Use.

The initial contents of the registry are:

Value	Certificate Policy	
0	Name:	Any Policy
	Identifiers:	anyPolicy
	OID:	2.5.29.32.0
	DER:	06 04 55 1D 20 00

	Comments:	RFC 5280
1	Name:	Domain Validation (DV)
	Identifiers:	domain-validated
	OID:	2.23.140.1.2.1
	DER:	06 06 67 81 0C 01 02 01
	Comments:	CA/Browser Forum
2	Name:	Organization Validation (OV)
	Identifiers:	organization-validated
	OID:	2.23.140.1.2.2
	DER:	06 06 67 81 0C 01 02 02
	Comments:	CA/Browser Forum
3	Name:	Individual Validation (IV)
	Identifiers:	individual-validated
	OID:	2.23.140.1.2.3
	DER:	06 06 67 81 0C 01 02 03
	Comments:	CA/Browser Forum
4	Name:	Extended Validation (EV)
	Identifiers:	ev-guidelines
	OID:	2.23.140.1.1
	DER:	06 05 67 81 0C 01 01
	Comments:	CA/Browser Forum
7	Name:	Resource PKI (RPKI)
	Identifiers:	id-cp-ipAddr-asNumber
	OID:	1.3.6.1.5.5.7.14.2
	DER:	06 08 2B 06 01 05 05 07 0E 02
	Comments:	RFC 3779
8	Name:	Resource PKI (RPKI) (Alternative)
	Identifiers:	id-cp-ipAddr-asNumber-v2
	OID:	1.3.6.1.5.5.7.14.3
	DER:	06 08 2B 06 01 05 05 07 0E 03
	Comments:	RFC 8360
24	Name:	Remote SIM Provisioning Role Certificate Issuer
	Identifiers:	id-rspRole-ci
	OID:	2.23.146.1.2.1.0
	DER:	06 07 67 81 12 01 02 01 00
	Comments:	GSMA SGP.22
25	Name:	Remote SIM Provisioning Role eUICC v2
	Identifiers:	id-rspRole-euicc-v2

	OID:	2.23.146.1.2.1.1
	DER:	06 07 67 81 12 01 02 01 01
	Comments:	GSMA SGP.22
26	Name:	Remote SIM Provisioning Role eUICC
	Identifiers:	id-rspRole-euicc
	OID:	2.23.146.1.2.1.0.0.0.0.0
	DER:	06 0B 67 81 12 01 02 01 00 00 00 00 00
	Comments:	GSMA SGP.22
27	Name:	Remote SIM Provisioning Role eUICC Manufacturer v2
	Identifiers:	id-rspRole-eum-v2
	OID:	2.23.146.1.2.1.2
	DER:	06 07 67 81 12 01 02 01 02
	Comments:	GSMA SGP.22
28	Name:	Remote SIM Provisioning Role eUICC Manufacturer
	Identifiers:	id-rspRole-eum
	OID:	2.23.146.1.2.1.0.0.0
	DER:	06 09 67 81 12 01 02 01 00 00 00
	Comments:	GSMA SGP.22
29	Name:	Remote SIM Provisioning Role SM-DP+ TLS v2
	Identifiers:	id-rspRole-dp-tls-v2
	OID:	2.23.146.1.2.1.3
	DER:	06 07 67 81 12 01 02 01 03
	Comments:	GSMA SGP.22
30	Name:	Remote SIM Provisioning Role SM-DP+ TLS
	Identifiers:	id-rspRole-dp-tls
	OID:	2.23.146.1.2.1.0.0.1.0
	DER:	06 0A 67 81 12 01 02 01 00 00 01 00
	Comments:	GSMA SGP.22
31	Name:	Remote SIM Provisioning Role SM-DP+ Authentication v2
	Identifiers:	id-rspRole-dp-auth-v2
	OID:	2.23.146.1.2.1.4
	DER:	06 07 67 81 12 01 02 01 04
	Comments:	GSMA SGP.22
32	Name:	Remote SIM Provisioning Role SM-DP+ Authentication

	Identifiers:	id-rspRole-dp-auth
	OID:	2.23.146.1.2.1.0.0.1.1
	DER:	06 0A 67 81 12 01 02 01 00 00 01 01
	Comments:	GSMA SGP.22
33	Name:	Remote SIM Provisioning Role SM-DP+ Profile Binding v2
	Identifiers:	id-rspRole-dp-pb-v2
	OID:	2.23.146.1.2.1.5
	DER:	06 07 67 81 12 01 02 01 05
	Comments:	GSMA SGP.22
34	Name:	Remote SIM Provisioning Role SM-DP+ Profile Binding
	Identifiers:	id-rspRole-dp-pb
	OID:	2.23.146.1.2.1.0.0.1.2
	DER:	06 0A 67 81 12 01 02 01 00 00 01 02
	Comments:	GSMA SGP.22
35	Name:	Remote SIM Provisioning Role SM-DS TLS v2
	Identifiers:	id-rspRole-ds-tls-v2
	OID:	2.23.146.1.2.1.6
	DER:	06 07 67 81 12 01 02 01 06
	Comments:	GSMA SGP.22
36	Name:	Remote SIM Provisioning Role SM-DS TLS
	Identifiers:	id-rspRole-ds-tls
	OID:	2.23.146.1.2.1.0.0.2.0
	DER:	06 0A 67 81 12 01 02 01 00 00 02 00
	Comments:	GSMA SGP.22
37	Name:	Remote SIM Provisioning Role SM-DS Authentication v2
	Identifiers:	id-rspRole-ds-auth-v2
	OID:	2.23.146.1.2.1.7
	DER:	06 07 67 81 12 01 02 01 07
	Comments:	GSMA SGP.22
38	Name:	Remote SIM Provisioning Role SM-DS Authentication
	Identifiers:	id-rspRole-ds-auth
	OID:	2.23.146.1.2.1.0.0.2.1
	DER:	06 0A 67 81 12 01 02 01 00 00 02 01
	Comments:	GSMA SGP.22

Figure 13: C509 Certificate Policies

8.10. C509 Policies Qualifiers Registry

IANA has created a new registry titled "C509 Policies Qualifiers" under the registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Identifiers, OID, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. The fields Name, OID, and DER are mandatory. If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the Policy Qualifier is described. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". Values 32768 are reserved for Private Use. For all other values the registration procedure is "Expert Review".

The initial contents of the registry are:

Value	Policy Qualifier	
1	Name:	Certification Practice Statement
	Identifiers:	id-qt-cps, cps
	OID:	1.3.6.1.5.5.7.2.1
	DER:	06 08 2B 06 01 05 05 07 02 01
	Comments:	RFC 5280
2	Name:	User Notice
	Identifiers:	id-qt-unotice, unotice
	OID:	1.3.6.1.5.5.7.2.2
	DER:	06 08 2B 06 01 05 05 07 02 02
	Comments:	RFC 5280

Figure 14: C509 Policies Qualifiers

8.11. C509 Information Access Registry

IANA has created a new registry titled "C509 Information Access" under the registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Identifiers, OID, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. The fields Name, OID, and DER are mandatory. If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the Information Access is described. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". For all other values the registration procedure is "Expert Review".

The initial contents of the registry are:

Value	Information Access	
1	Name:	OCSP
	Identifiers:	id-ad-ocsp, id-pkix-ocsp
	OID:	1.3.6.1.5.5.7.48.1
	DER:	06 08 2B 06 01 05 05 07 30 01
	Comments:	RFC 5280
2	Name:	CA Issuers
	Identifiers:	id-ad-caIssuers, caIssuers
	OID:	1.3.6.1.5.5.7.48.2
	DER:	06 08 2B 06 01 05 05 07 30 02
	Comments:	RFC 5280
3	Name:	Time Stamping
	Identifiers:	id-ad-timeStamping, timeStamping
	OID:	1.3.6.1.5.5.7.48.3
	DER:	06 08 2B 06 01 05 05 07 30 03
	Comments:	RFC 3161
5	Name:	CA Repository
	Identifiers:	id-ad-caRepository
	OID:	1.3.6.1.5.5.7.48.5
	DER:	06 08 2B 06 01 05 05 07 30 05
	Comments:	RFC 5280
10	Name:	RPKI Manifest
	Identifiers:	id-ad-rpkiManifest
	OID:	1.3.6.1.5.5.7.48.10
	DER:	06 08 2B 06 01 05 05 07 30 0A
	Comments:	RFC 6487
11	Name:	Signed Object
	Identifiers:	id-ad-signedObject
	OID:	1.3.6.1.5.5.7.48.11
	DER:	06 08 2B 06 01 05 05 07 30 0B
	Comments:	RFC 6487
13	Name:	RPKI Notify
	Identifiers:	id-ad-rpkiNotify
	OID:	1.3.6.1.5.5.7.48.13
	DER:	06 08 2B 06 01 05 05 07 30 0D
	Comments:	RFC 8182

Figure 15: C509 Information Accesses

8.12. C509 Extended Key Usages Registry

IANA has created a new registry titled "C509 Extended Key Usages" under the registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Identifiers, OID, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. The fields Name, OID, and DER are mandatory. For Extended Key Usage specified only for CBOR encoded certificates where no OID is defined, the OID and DER fields are marked "N/A". If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the Extended Key Usage is described. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". Values 32768 are reserved for Private Use. For all other values the registration procedure is "Expert Review".

The initial contents of the registry are:

Value	Extended Key Usage	
0	Name:	Any Extended Key Usage
	Identifiers:	anyExtendedKeyUsage
	OID:	2.5.29.37.0
	DER:	06 04 55 1D 25 00
	Comments:	RFC 5280
1	Name:	TLS Server authentication
	Identifiers:	id-kp-serverAuth
	OID:	1.3.6.1.5.5.7.3.1
	DER:	06 08 2B 06 01 05 05 07 03 01
	Comments:	RFC 5280
2	Name:	TLS Client Authentication
	Identifiers:	id-kp-clientAuth
	OID:	1.3.6.1.5.5.7.3.2
	DER:	06 08 2B 06 01 05 05 07 03 02
	Comments:	RFC 5280
3	Name:	Code Signing
	Identifiers:	id-kp-codeSigning
	OID:	1.3.6.1.5.5.7.3.3
	DER:	06 08 2B 06 01 05 05 07 03 03
	Comments:	RFC 5280
4	Name:	Email protection (S/MIME)

		Identifiers:	id-kp-emailProtection
		OID:	1.3.6.1.5.5.7.3.4
		DER:	06 08 2B 06 01 05 05 07 03 04
		Comments:	RFC 5280
8		Name:	Time Stamping
		Identifiers:	id-kp-timeStamping, timestamping
		OID:	1.3.6.1.5.5.7.3.8
		DER:	06 08 2B 06 01 05 05 07 03 08
		Comments:	RFC 3161
9		Name:	OCSP Signing
		Identifiers:	id-kp-OCSPSigning
		OID:	1.3.6.1.5.5.7.3.9
		DER:	06 08 2B 06 01 05 05 07 03 09
		Comments:	RFC 5280
10		Name:	Kerberos PKINIT Client Auth
		Identifiers:	id-pkinit-KPClientAuth
		OID:	1.3.6.1.5.2.3.4
		DER:	06 07 2B 06 01 05 02 03 04
		Comments:	RFC 4556
11		Name:	Kerberos PKINIT KDC
		Identifiers:	id-pkinit-KPKdc
		OID:	1.3.6.1.5.2.3.5
		DER:	06 07 2B 06 01 05 02 03 05
		Comments:	RFC 4556
12		Name:	SSH Client
		Identifiers:	id-kp-secureShellClient
		OID:	1.3.6.1.5.5.7.3.21
		DER:	06 08 2B 06 01 05 05 07 03 15
		Comments:	RFC 6187
13		Name:	SSH Server
		Identifiers:	id-kp-secureShellServer
		OID:	1.3.6.1.5.5.7.3.22
		DER:	06 08 2B 06 01 05 05 07 03 16
		Comments:	RFC 6187
14		Name:	Bundle Security
		Identifiers:	id-kp-bundleSecurity
		OID:	1.3.6.1.5.5.7.3.35
		DER:	06 08 2B 06 01 05 05 07 03 23
		Comments:	RFC 9174
15		Name:	CMC Certification Authority

	Identifiers:	id-kp-cmcCA	
	OID:	1.3.6.1.5.5.7.3.27	
	DER:	06 08 2B 06 01 05 05 07 03 1B	
	Comments:	RFC 6402	
16	Name:	CMC Registration Authority	
	Identifiers:	id-kp-cmcRA	
	OID:	1.3.6.1.5.5.7.3.28	
	DER:	06 08 2B 06 01 05 05 07 03 1C	
	Comments:	RFC 6402	
17	Name:	CMC Archive Server	
	Identifiers:	id-kp-cmcArchive	
	OID:	1.3.6.1.5.5.7.3.29	
	DER:	06 08 2B 06 01 05 05 07 03 1D	
	Comments:	RFC 6402	
18	Name:	CMC Key Generation Authority	
	Identifiers:	id-kp-cmKGA	
	OID:	1.3.6.1.5.5.7.3.32	
	DER:	06 08 2B 06 01 05 05 07 03 20	
	Comments:	RFC 9480	
20	Name:	Wi-SUN FAN Device	
	Identifiers:	id-kp-wisun-fan-device	
	OID:	1.3.6.1.4.1.45605.1	
	DER:	06 09 2B 06 01 04 01 82 E4 25 01	
	Comments:		

Figure 16: C509 Extended Key Usages

8.13. C509 General Names Registry

IANA has created a new registry titled "C509 General Names" under the registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Comments, GeneralNameValue, and Reference, where Value is an integer, and the other columns are text strings. The fields Name and GeneralNameValue are mandatory. If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the General Name is described. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". For all other values the registration procedure is "Expert Review".

The initial contents of the registry are:

Value	General Name	
-3	Name:	otherName with MACAddress
	Comments:	TBD92(Use RFC I-D-lamps-macaddress-on)
		id-on-MACAddress
		(1.3.6.1.5.5.7.8.12)
		06 08 2B 06 01 05 05 07 08 0C
	GeneralNameValue:	bytes
-2	Name:	otherName with SmtpUTF8Mailbox
	Comments:	RFC 9598
		id-on-SmtpUTF8Mailbox
		(1.3.6.1.5.5.7.8.9)
		06 08 2B 06 01 05 05 07 08 09
	GeneralNameValue:	text
-1	Name:	otherName with hardwareModuleName
	Comments:	RFC 4108
		id-on-hardwareModuleName
		(1.3.6.1.5.5.7.8.4)
		06 08 2B 06 01 05 05 07 08 04
	GeneralNameValue:	[~oid, bytes]
0	Name:	otherName
	Comments:	RFC 5280
	GeneralNameValue:	[~oid, bytes]
1	Name:	rfc822Name
	Comments:	RFC 5280
	GeneralNameValue:	text
2	Name:	dNSName
	Comments:	RFC 5280
	GeneralNameValue:	text
4	Name:	directoryName
	Comments:	RFC 5280
	GeneralNameValue:	Name
6	Name:	uniformResourceIdentifier
	Comments:	RFC 5280
	GeneralNameValue:	text
7	Name:	iPAddress
	Comments:	RFC 5280
	GeneralNameValue:	bytes

8	Name:	registeredID
	Comments	RFC 5280
	GeneralNameValue:	~oid

Figure 17: C509 General Names

8.14. C509 Signature Algorithms Registry

IANA has created a new registry titled "C509 Signature Algorithms" under the registry group "CBOR Encoded X.509 (C509)". The registry includes both signature algorithms and non-signature proof-of-possession algorithms. The fields of the registry are Value, Name, Identifiers, OID, Parameters, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. The fields Name, OID, Parameters, and DER are mandatory. Alignment with the value of public key algorithm must be considered, see instruction in Section 8.15. If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the Signature Algorithm is described. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". For all other values the registration procedure is "Expert Review". The initial contents of the registry are:

Value	Signature Algorithm	
-256	Name:	RSASSA-PKCS1-v1_5 with SHA-1
	Identifiers:	sha1-with-rsa-signature, sha1WithRSAEncryption, sha-1WithRSAEncryption
	OID:	1.2.840.113549.1.1.5
	Parameters:	NULL
	DER:	30 0D 06 09 2A 86 48 86 F7 0D 01 01 05 05 00
	Comments:	
-255	Name:	ECDSA with SHA-1
	Identifiers:	ecdsa-with-SHA1
	OID:	1.2.840.10045.4.1
	Parameters:	Absent
	DER:	30 09 06 07 2A 86 48 CE 3D 04 01
	Comments:	See Section 3.2.2.
0	Name:	ECDSA with SHA-256
	Identifiers:	ecdsa-with-SHA256
	OID:	1.2.840.10045.4.3.2
	Parameters:	Absent

	DER:	30 0A 06 08 2A 86 48 CE 3D 04 03 02
	Comments:	See Section 3.2.2.
1	Name:	ECDSA with SHA-384
	Identifiers:	ecdsa-with-SHA384
	OID:	1.2.840.10045.4.3.3
	Parameters:	Absent
	DER:	30 0A 06 08 2A 86 48 CE 3D 04 03 03
	Comments:	See Section 3.2.2.
2	Name:	ECDSA with SHA-512
	Identifiers:	ecdsa-with-SHA512
	OID:	1.2.840.10045.4.3.4
	Parameters:	Absent
	DER:	30 0A 06 08 2A 86 48 CE 3D 04 03 04
	Comments:	See Section 3.2.2.
3	Name:	ECDSA with SHAKE128
	Identifiers:	id-ecdsa-with-shake128
	OID:	1.3.6.1.5.5.7.6.32
	Parameters:	Absent
	DER:	30 0A 06 08 2B 06 01 05 05 07 06 20
	Comments:	See Section 3.2.2.
4	Name:	ECDSA with SHAKE256
	Identifiers:	id-ecdsa-with-shake256
	OID:	1.3.6.1.5.5.7.6.33
	Parameters:	Absent
	DER:	30 0A 06 08 2B 06 01 05 05 07 06 21
	Comments:	See Section 3.2.2.
5	Name:	Unsigned
	Identifiers:	id-alg-unsigned
	OID:	1.3.6.1.5.5.7.6.36
	Parameters:	Absent
	DER:	30 0A 06 08 2B 06 01 05 05 07 06 24
	Comments:	bytes of size 0
8	Name:	SM2 with SM3
	Identifiers:	sm2-with-sm3
	OID:	1.2.156.10197.1.501
	Parameters:	Absent
	DER:	30 0A 06 08 2A 81 1C CF 55 01 83 75
	Comments:	See Section 3.2.2.
12	Name:	Ed25519
	Identifiers:	id-Ed25519, id-EdDSA25519
	OID:	1.3.101.112

	Parameters:	Absent
	DER:	30 05 06 03 2B 65 70
	Comments:	
13	Name:	Ed448
	Identifiers:	id-Ed448, id-EdDSA448
	OID:	1.3.101.113
	Parameters:	Absent
	DER:	30 05 06 03 2B 65 71
	Comments:	
14	Name:	PoP with SHA-256 and HMAC-SHA256
	Identifiers:	sa-ecdhPop-sha256-hmac-sha256
	OID:	1.3.6.1.5.5.7.6.26
	Parameters:	Absent
	DER:	30 0A 06 08 2B 06 01 05 05 07 06 1A
	Comments:	Proof-of-possession algorithm, indexed with KDF and MAC, see RFC 6955. Requires recipient's public static Diffie-Hellman key
15	Name:	PoP with SHA-384 and HMAC-SHA384
	Identifiers:	sa-ecdhPop-sha384-hmac-sha384
	OID:	1.3.6.1.5.5.7.6.27
	Parameters:	Absent
	DER:	30 0A 06 08 2B 06 01 05 05 07 06 1B
	Comments:	Proof-of-possession algorithm, indexed with KDF and MAC, see RFC 6955. Requires recipient's public static Diffie-Hellman key
16	Name:	PoP with SHA-512 and HMAC-SHA512
	Identifiers:	sa-ecdhPop-sha512-hmac-sha512
	OID:	1.3.6.1.5.5.7.6.28
	Parameters:	Absent
	DER:	30 0A 06 08 2B 06 01 05 05 07 06 1C
	Comments:	Proof-of-possession algorithm, indexed with KDF and MAC, see RFC 6955. Requires recipient's public static Diffie-Hellman key
23	Name:	RSASSA-PKCS1-v1_5 with SHA-256
	Identifiers:	sha256WithRSAEncryption
	OID:	1.2.840.113549.1.1.11
	Parameters:	NULL
	DER:	30 0B 06 09 2A 86 48 86 F7 0D 01 01 0B 05 00
	Comments:	
24	Name:	RSASSA-PKCS1-v1_5 with SHA-384
	Identifiers:	sha384WithRSAEncryption
	OID:	1.2.840.113549.1.1.12

	Parameters:	NULL
	DER:	30 0B 06 09 2A 86 48 86 F7 0D 01 01 0C 05 00
	Comments:	
25	Name:	RSASSA-PKCS1-v1_5 with SHA-512
	Identifiers:	sha512WithRSAEncryption
	OID:	1.2.840.113549.1.1.13
	Parameters:	NULL
	DER:	30 0B 06 09 2A 86 48 86 F7 0D 01 01 0D 05 00
	Comments:	
26	Name:	RSASSA-PSS with SHA-256
	Identifiers:	rsassa-pss, id-RSASSA-PSS
	OID:	1.2.840.113549.1.1.10
	Parameters:	SHA-256, MGF-1 with SHA-256, saltLength = 32
	DER:	30 41 06 09 2A 86 48 86 F7 0D 01 01 0A 30 34 A0 0F 30 0D 06 09 60 86 48 01 65 03 04 02 01 05 00 A1 1C 30 1A 06 09 2A 86 48 86 F7 0D 01 01 08 30 0D 06 09 60 86 48 01 65 03 04 02 01 05 00 A2 03 02 01 20
	Comments:	
27	Name:	RSASSA-PSS with SHA-384
	Identifiers:	rsassa-pss, id-RSASSA-PSS
	OID:	1.2.840.113549.1.1.10
	Parameters:	SHA-384, MGF-1 with SHA-384, saltLength = 48
	DER:	30 41 06 09 2A 86 48 86 F7 0D 01 01 0A 30 34 A0 0F 30 0D 06 09 60 86 48 01 65 03 04 02 02 05 00 A1 1C 30 1A 06 09 2A 86 48 86 F7 0D 01 01 08 30 0D 06 09 60 86 48 01 65 03 04 02 02 05 00 A2 03 02 01 30
	Comments:	
28	Name:	RSASSA-PSS with SHA-512
	Identifiers:	rsassa-pss, id-RSASSA-PSS
	OID:	1.2.840.113549.1.1.10
	Parameters:	SHA-512, MGF-1 with SHA-512, saltLength = 64
	DER:	30 41 06 09 2A 86 48 86 F7 0D 01 01 0A 30 34 A0 0F 30 0D 06 09 60 86 48 01 65 03 04 02 03 05 00 A1 1C 30 1A 06 09 2A 86 48 86 F7 0D 01 01 08 30 0D 06 09 60 86 48 01 65 03 04 02 03 05 00 A2 03 02 01 40
	Comments:	
29	Name:	RSASSA-PSS with SHAKE128
	Identifiers:	id-RSASSA-PSS-SHAKE128
	OID:	1.3.6.1.5.5.7.6.30
	Parameters:	Absent

	DER:	30 0A 06 08 2B 06 01 05 05 07 06 1E
	Comments:	
30	Name:	RSASSA-PSS with SHAKE256
	Identifiers:	id-RSASSA-PSS-SHAKE256
	OID:	1.3.6.1.5.5.7.6.31
	Parameters:	Absent
	DER:	30 0A 06 08 2B 06 01 05 05 07 06 1F
	Comments:	

Figure 18: C509 Signature Algorithms

8.15. C509 Public Key Algorithms Registry

IANA has created a new registry titled "C509 Public Key Algorithms" under the registry group "CBOR Encoded X.509 (C509)". The fields of the registry are Value, Name, Identifiers, OID, Parameters, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. The fields Name, OID, Parameters, and DER are mandatory. If the public key can only be used with one signature algorithm and the OID of the public key algorithm is the same as the signature algorithm, then the value must be chosen equal to the value of signature algorithm, see Section 8.14. If it is not expected to be understood from the other information (e.g. the OID), then the Comments field must contain a reference to where the Public Key Algorithm is described. For values in the interval [-24, 23] the registration procedure is "IETF Review with Expert Review". For all other values the registration procedure is "Expert Review". The initial contents of the registry are:

Value	Public Key Algorithm	
0	Name:	RSA
	Identifiers:	rsaEncryption
	OID:	1.2.840.113549.1.1.1
	Parameters:	NULL
	DER:	30 0D 06 09 2A 86 48 86 F7 0D 01 01 01 05 00
	Comments:	subjectPublicKey encoded as in Section 3.2.1
1	Name:	EC Public Key (Weierstrass) with secp256r1
	Identifiers:	ecPublicKey, id-ecPublicKey
	OID:	1.2.840.10045.2.1
	Parameters:	namedCurve = secp256r1 (1.2.840.10045.3.1.7)
	DER:	30 13 06 07 2A 86 48 CE 3D 02 01 06 08 2A 86 48 CE 3D 03 01 07
	Comments:	subjectPublicKey encoded as in Section 3.2.1

		Also known as P-256, ansip256r1, prime256v1
2	Name:	EC Public Key (Weierstrass) with secp384r1
	Identifiers:	ecPublicKey, id-ecPublicKey
	OID:	1.2.840.10045.2.1
	Parameters:	namedCurve = secp384r1 (1.3.132.0.34)
	DER:	30 10 06 07 2A 86 48 CE 3D 02 01 06 05 2B 81 04 00 22
	Comments:	subjectPublicKey encoded as in Section 3.2.1 Also known as P-384, ansip384r1
3	Name:	EC Public Key (Weierstrass) with secp521r1
	Identifiers:	ecPublicKey, id-ecPublicKey
	OID:	1.2.840.10045.2.1
	Parameters:	namedCurve = secp521r1 (1.3.132.0.35)
	DER:	30 10 06 07 2A 86 48 CE 3D 02 01 06 05 2B 81 04 00 23
	Comments:	subjectPublicKey encoded as in Section 3.2.1 Also known as P-521, ansip521r1
6	Name:	EC Public Key (Weierstrass) with sm2p256v1
	Identifiers:	ecPublicKey, id-ecPublicKey
	OID:	1.2.840.10045.2.1
	Parameters:	namedCurve = sm2p256v1 (1.2.156.10197.1.301)
	DER:	30 13 06 07 2A 86 48 CE 3D 02 01 06 08 2A 81 1C CF 55 01 82 2D
	Comments:	subjectPublicKey encoded as in Section 3.2.1
8	Name:	X25519 (Montgomery)
	Identifiers:	id-X25519
	OID:	1.3.101.110
	Parameters:	Absent
	DER:	30 05 06 03 2B 65 6E
	Comments:	
9	Name:	X448 (Montgomery)
	Identifiers:	id-X448
	OID:	1.3.101.111
	Parameters:	Absent
	DER:	30 05 06 03 2B 65 6F
	Comments:	
12	Name:	Ed25519 (Twisted Edwards)
	Identifiers:	id-Ed25519, id-EdDSA25519
	OID:	1.3.101.112
	Parameters:	Absent

	DER:	30 05 06 03 2B 65 70
	Comments:	
13	Name:	Ed448 (Edwards)
	Identifiers:	id-Ed448, id-EdDSA448
	OID:	1.3.101.113
	Parameters:	Absent
	DER:	30 05 06 03 2B 65 71
	Comments:	
24	Name:	EC Public Key (Weierstrass) with brainpoolP256r1
	Identifiers:	ecPublicKey, id-ecPublicKey
	OID:	1.2.840.10045.2.1
	Parameters:	namedCurve = brainpoolP256r1 (1.3.36.3.3.2.8.1.1.7)
	DER:	30 14 06 07 2A 86 48 CE 3D 02 01 06 09 2B 24 03 03 02 08 01 01 07
	Comments:	subjectPublicKey encoded as in Section 3.2.1
25	Name:	EC Public Key (Weierstrass) with brainpoolP384r1
	Identifiers:	ecPublicKey, id-ecPublicKey
	OID:	1.2.840.10045.2.1
	Parameters:	namedCurve = brainpoolP384r1 (1.3.36.3.3.2.8.1.1.11)
	DER:	30 14 06 07 2A 86 48 CE 3D 02 01 06 09 2B 24 03 03 02 08 01 01 0B
	Comments:	subjectPublicKey encoded as in Section 3.2.1
26	Name:	EC Public Key (Weierstrass) with brainpoolP512r1
	Identifiers:	ecPublicKey, id-ecPublicKey
	OID:	1.2.840.10045.2.1
	Parameters:	namedCurve = brainpoolP512r1 (1.3.36.3.3.2.8.1.1.13)
	DER:	30 14 06 07 2A 86 48 CE 3D 02 01 06 09 2B 24 03 03 02 08 01 01 0D
	Comments:	subjectPublicKey encoded as in Section 3.2.1
27	Name:	EC Public Key (Weierstrass) with FRP256v1
	Identifiers:	ecPublicKey, id-ecPublicKey
	OID:	1.2.840.10045.2.1
	Parameters:	namedCurve = FRP256v1 (1.2.250.1.223.101.256.1)
	DER:	30 15 06 07 2A 86 48 CE 3D 02 01 06 0A 2A 81 7A 01 81 5F 65 82 00 01

| | Comments: subjectPublicKey encoded as in Section 3.2.1 |
+-------+---+

Figure 19: C509 Public Key Algorithms

8.16. COSE Header Parameters Registry

IANA is requested to assign the entries in Table 1 to the "COSE Header Parameters" registry in the registry group "CBOR Object Signing and Encryption (COSE)" with this document as reference.

8.17. COSE Header Algorithm Parameters Registry

IANA is requested to assign the entries in Table 2 to the "COSE Header Algorithm Parameters" registry in the registry group "CBOR Object Signing and Encryption (COSE)" with this document as reference.

8.18. Media Type Application Registry

IANA is requested to assign the following entries into the "application" registry in the registry group "Media Types" with this document as reference.

8.18.1. Media Type application/cose-c509-cert

When the application/cose-c509-cert media type is used, the data is a COSE_C509 structure. If the parameter "usage" is set to "chain", this sequence indicates a certificate chain.

Type name: application

Subtype name: cose-c509-cert

Required parameters: N/A

Optional parameters: usage

* Can be absent to provide no further information about the intended meaning of the order in the CBOR sequence of certificates.

* Can be set to "chain" to indicate that the sequence of data items is to be interpreted as a certificate chain.

Encoding considerations: binary

Security considerations: See the Security Considerations section of [[this document]].

Interoperability considerations: N/A

Published specification: [[this document]]

Applications that use this media type: Applications that employ COSE and use C509 as a certificate type.

Fragment identifier considerations: N/A

Additional information:

* Deprecated alias names for this type: N/A

* Magic number(s): TBD8, TBD6

* File extension(s): .c509

* Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: COSE WG

Change controller: IETF

8.18.2. Media Type application/cose-c509-pkcs10

When the application/cose-c509-pkcs10 media type is used, the data is a C509CertificationRequest structure.

Type name: application

Subtype name: cose-c509-pkcs10

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: See the Security Considerations section of [[this document]].

Interoperability considerations: N/A

Published specification: [[this document]]

Applications that use this media type: Applications that employ COSE and C509 Certification Request.

Fragment identifier considerations: N/A

Additional information:

* Deprecated alias names for this type: N/A

* Magic number(s): TBD9

* File extension(s): .c509

* Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: COSE WG

Change controller: IETF

8.18.3. Media Type application/cose-c509-crtemplate

When the application/cose-c509-crtemplate media type is used, the data is a C509CertificationRequestTemplate structure.

Type name: application

Subtype name: cose-c509-crtemplate

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: See the Security Considerations section of [[this document]].

Interoperability considerations: N/A

Published specification: [[this document]]

Applications that use this media type: Applications that employ COSE and C509 Certification Request.

Fragment identifier considerations: N/A

Additional information:

* Deprecated alias names for this type: N/A

* Magic number(s): TBD18

* File extension(s): .c509

* Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: COSE WG

Change controller: IETF

8.18.4. Media Type application/cose-c509-privkey

When the application/cose-c509-privkey media type is used, the data is a C509PrivateKey structure.

Type name: application

Subtype name: cose-c509-privkey

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: See the Security Considerations section of [[this document]].

Interoperability considerations: N/A

Published specification: [[this document]]

Applications that use this media type: Applications that employ COSE and use C509 as a certificate type.

Fragment identifier considerations: N/A

Additional information:

* Deprecated alias names for this type: N/A

* Magic number(s): TBD12

* File extension(s): .c509

* Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: COSE WG

Change controller: IETF

8.18.5. Media Type application/cose-c509-pem

When the application/cose-c509-pem media type is used, the data is a C509PEM structure.

Type name: application

Subtype name: cose-c509-pem

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: See the Security Considerations section of [[this document]].

Interoperability considerations: N/A

Published specification: [[this document]]

Applications that use this media type: Applications that employ COSE and use C509 as a certificate type.

Fragment identifier considerations: N/A

Additional information:

* Deprecated alias names for this type: N/A

* Magic number(s): TBD13

* File extension(s): .c509

* Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: COSE WG

Change controller: IETF

8.18.6. Media Type application/cose-certhash

When the application/cose-certhash media type is used, the data is a COSE_CertHash structure as defined in [RFC9360]. If the parameter "usage" is set to "c509", the hash value is calculated over a C509 certificate.

Type name: application

Subtype name: cose-certhash

Required parameters: N/A

Optional parameters: usage

* Can be absent to provide no further information about what the hash value is calculated over.

* Can be set to "c509" to indicate that the COSE_CertHash structure as defined in [RFC9360] is used, with hashValue calculated over a C509 certificate as defined in Section 3.4.

Encoding considerations: binary

Security considerations: See the Security Considerations section of [RFC9360].

Interoperability considerations: N/A

Published specification: [[this document]]

Applications that use this media type: Applications that employ COSE and use X.509 or C509 as certificate type.

Fragment identifier considerations: N/A

Additional information:

* Deprecated alias names for this type: N/A

* Magic number(s): N/A

* File extension(s): N/A

* Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: COSE WG

Change controller: IETF

8.19. CoAP Content-Formats Registry

IANA is requested to add entries for "application/cose-c509-cert", "application/cose-c509-pkcs10", "application/cose-c509-crtemplate", "application/cose-c509-privkey" and "application/cose-c509-pem" to the "CoAP Content-Formats" registry in the registry group "Constrained RESTful Environments (CoRE) Parameters". A dedicated Content-Format ID is requested for the "application/cose-c509-cert" media type in the case when the parameter "usage" is set to "chain",

see Section 8.18.1.

IANA is requested to add entries for "application/cose-certhash" to the "CoAP Content-Formats" registry in the registry group "Constrained RESTful Environments (CoRE) Parameters". A dedicated Content-Format ID is requested in the case when the parameter "usage" is set to "c509", see Section 8.18.6.

IANA is requested to add entries for "application/cbor" to the "CoAP Content-Formats" registry in the registry group "Constrained RESTful Environments (CoRE) Parameters", in the case when the encoding is a CBOR text string containing a URI, see [RFC3986].

Content Format	Content Coding	Media Type	ID	Reference
application/ cose-c509-cert	-	[[link to 8.18]]	TBD3	[[this document]]
application/ cose-c509-cert; usage=chain	-	[[link to 8.18]]	TBD15	[[this document]]
application/ cose-c509-pkcs10	-	[[link to 8.18]]	TBD4	[[this document]]
application/ cose-c509-crtemplate	-	[[link to 8.18]]	TBD19	[[this document]]
application/ cose-c509-privkey	-	[[link to 8.18]]	TBD10	[[this document]]
application/ cose-c509-pem	-	[[link to 8.18]]	TBD11	[[this document]]
application/ cose-certhash	-	[[link to 8.18]]	TBD16	[[this document]]
application/ cose-certhash; usage=c509	-	[[link to 8.18]]	TBD17	[[this document]]

Figure 20: CoAP Content-Format IDs

8.20. TLS Certificate Types Registry

This document registers the following entry in the "TLS Certificate Types" registry in the registry group "Transport Layer Security (TLS) Extensions". The new certificate type can be used with additional TLS certificate compression [RFC8879]. For TLS 1.3, the C509 certificate type is defined as a new case in the CertificateEntry struct specified in Section 4.4.2 of [RFC8446]:

```
case C509:
  opaque c509_data<1..2^24-1>;
```

where c509_data is the CBOR sequence ~C509Certificate (an unwrapped C509Certificate). For TLS 1.2 the same construction is applied with a similar union type defined for the Certificate struct in Section 7.4.2 of [RFC5246]. Note that, similar to COSE_C509, the TLS handshake contains the length of each certificate. The TLS extensions client_certificate_type and server_certificate_type [RFC7250] are used to negotiate the use of C509.

Value	Name	Recommended	Comment
TBD5	C509 Certificate	N	

8.21. TLSA Selectors Registry

This document registers the following entry in the "TLSA Selectors" registry in the registry group "DNS-Based Authentication of Named Entities (DANE) Parameters". The C509 certificate data, C509CertData, is defined in Section 3.4.

Value	Acronym	Short Description	Reference
TBD7	C509	C509 certificate data	[[this document]]

The TLSA selectors registry defined in [RFC6698] originally only applied to PKIX [RFC5280] certificates in DER encoding. This specification updates [RFC6698] to accept the use of C509 certificates.

8.22. EDHOC Authentication Credential Types Registry

This document registers the following entry in the "EDHOC Authentication Credential Types" registry in the registry group "Ephemeral Diffie-Hellman Over COSE (EDHOC)". This is useful to identify C509 certificates as a supported authentication credential type to use with EDHOC [RFC9528], for example, during discovery of EDHOC resources, see [RFC9668].

Value	Description	Reference
3	C509 certificate	[[this document]]

8.23. Relative Distinguished Name Attribute

This document registers the following entry in the "SMI Security for PKIX Relative Distinguished Name Attribute" registry [RFC7299]:

Decimal	Description	Reference
TBD30	id-rdna-c509Name	[[this document]]

9. References

9.1. Normative References

[I-D.ietf-lamps-macaddress-on]
Housley, R., Bonnell, C., Mandel, J., Okubo, T., and M. StJohns, "Media Access Control (MAC) Addresses in X.509 Certificates", Work in Progress, Internet-Draft, draft-ietf-lamps-macaddress-on-07, 12 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-macaddress-on-07>.

[POSIX] "IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(TM)) Base Specifications, Issue 7", January 2018, <https://pubs.opengroup.org/onlinepubs/9699919799/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC2985] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, DOI 10.17487/RFC2985, November 2000, <https://www.rfc-editor.org/rfc/rfc2985>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/rfc/rfc2986>.

[RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <https://www.rfc-editor.org/rfc/rfc3779>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC4108] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, DOI 10.17487/RFC4108, August 2005, <https://www.rfc-editor.org/rfc/rfc4108>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/rfc/rfc5958>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/rfc/rfc6066>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/rfc/rfc6698>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/rfc/rfc7030>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/rfc/rfc7120>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/rfc/rfc7250>.

[RFC7299] Housley, R., "Object Identifier Registry for the PKIX Working Group", RFC 7299, DOI 10.17487/RFC7299, July 2014, <https://www.rfc-editor.org/rfc/rfc7299>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8295] Turner, S., "EST (Enrollment over Secure Transport) Extensions", RFC 8295, DOI 10.17487/RFC8295, January 2018, <https://www.rfc-editor.org/rfc/rfc8295>.

[RFC8360] Huston, G., Michaelson, G., Martinez, C., Bruijnzeels, T., Newton, A., and D. Shaw, "Resource Public Key Infrastructure (RPKI) Validation Reconsidered", RFC 8360, DOI 10.17487/RFC8360, April 2018, <https://www.rfc-editor.org/rfc/rfc8360>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/rfc/rfc8742>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[RFC9090] Bormann, C., "Concise Binary Object Representation (CBOR) Tags for Object Identifiers", RFC 9090, DOI 10.17487/RFC9090, July 2021, <https://www.rfc-editor.org/rfc/rfc9090>.

[RFC9277] Richardson, M. and C. Bormann, "On Stable Storage for Items in Concise Binary Object Representation (CBOR)", RFC 9277, DOI 10.17487/RFC9277, August 2022, <https://www.rfc-editor.org/rfc/rfc9277>.

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", RFC 9360, DOI 10.17487/RFC9360, February 2023, <https://www.rfc-editor.org/rfc/rfc9360>.

[RFC9542] Eastlake 3rd, D., Abley, J., and Y. Li, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 9542, DOI 10.17487/RFC9542, April 2024, <https://www.rfc-editor.org/rfc/rfc9542>.

[RFC9549] Housley, R., "Internationalization Updates to RFC 5280", RFC 9549, DOI 10.17487/RFC9549, March 2024, <https://www.rfc-editor.org/rfc/rfc9549>.

[RFC9598] Melnikov, A., Chuang, W., and C. Bonnell, "Internationalized Email Addresses in X.509 Certificates", RFC 9598, DOI 10.17487/RFC9598, May 2024, <https://www.rfc-editor.org/rfc/rfc9598>.

[RFC9668] Palombini, F., Tiloca, M., Hglund, R., Hristozov, S., and G. Selander, "Using Ephemeral Diffie-Hellman Over COSE (EDHOC) with the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE)", RFC 9668, DOI 10.17487/RFC9668, November 2024, <https://www.rfc-editor.org/rfc/rfc9668>.

[RFC9883] Housley, R., "An Attribute for Statement of Possession of a Private Key", RFC 9883, DOI 10.17487/RFC9883, October 2025, <https://www.rfc-editor.org/rfc/rfc9883>.

[SECG] "Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, ver. 2", 2009, <https://secg.org/sec1-v2.pdf>.

[Wi-SUN] "Wi-SUN Alliance", n.d., <https://wi-sun.org>.

[X.501] "Information Technology - Open Systems Interconnection - The Directory: Models, ITU-T X.501", December 2019, <https://www.itu.int/rec/T-REC-X.501/en>.

[X.520] "Information Technology - Open Systems Interconnection - The Directory: Selected attribute types", October 2019, <https://www.itu.int/rec/T-REC-X.520/en>.

[X.690] "ASN.1 encoding rules. Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", n.d., <https://www.itu.int/rec/T-REC-X.690>.

9.2. Informative References

[CAB-Code] CA/Browser Forum, "CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Code Signing Certificates Version 3.8.0"", August 2024, <https://cabforum.org/baseline-requirements-code-signing/>.

[CAB-TLS] CA/Browser Forum, "CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates Version 2.1.4"", March 2025, <https://cabforum.org/baseline-requirements-documents/>.

[CborMe] Bormann, C., "CBOR Playground", May 2018, <https://cbor.me/>.

[GSMA-eUICC]
GSMA, "GSMA eUICC PKI Certificate Policy Version 2.2", January 2025, <https://www.gsma.com/solutions-and-impact/technologies/esim/wp-content/uploads/2025/01/SGP.14-v2.2.pdf>.

[GSMA-SGP.22]
"GSMA RSP Technical Specification Version 3.1 Final", December 2023, <https://www.gsma.com/solutions-and-impact/technologies/esim/wp-content/uploads/2023/12/SGP.22-v3.1.pdf>.

[IANA-AFI] "Address Family Numbers", n.d., <https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>.

[IANA-CBOR-TAGS]
IANA, "Concise Binary Object Representation (CBOR) Tags", n.d., <https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml>.

[IANA-SAFI]
"Subsequent Address Family Identifiers (SAFI) Parameters", n.d., <https://www.iana.org/assignments/safi-namespace/safi-namespace.xhtml>.

[IEEE-802.1AR]
Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networksSecure Device Identity", IEEE Standard 802.1AR-2018 , August 2018, <https://standards.ieee.org/standard/802_1AR-2018.html>.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/rfc/rfc3161>.

[RFC4524] Zeilenga, K., Ed., "COSINE LDAP/X.500 Schema", RFC 4524, DOI 10.17487/RFC4524, June 2006, <https://www.rfc-editor.org/rfc/rfc4524>.

[RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <https://www.rfc-editor.org/rfc/rfc6487>.

[RFC6955] Schaad, J. and H. Prafullchandra, "Diffie-Hellman Proof-of-Possession Algorithms", RFC 6955, DOI 10.17487/RFC6955, May 2013, <https://www.rfc-editor.org/rfc/rfc6955>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/rfc/rfc6960>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/rfc/rfc7228>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/rfc/rfc7468>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/rfc/rfc7925>.

[RFC7932] Alakuijala, J. and Z. Szabadka, "Brotli Compressed Data Format", RFC 7932, DOI 10.17487/RFC7932, July 2016, <https://www.rfc-editor.org/rfc/rfc7932>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8603] Jenkins, M. and L. Zieglar, "Commercial National Security Algorithm (CNSA) Suite Certificate and Certificate Revocation List (CRL) Profile", RFC 8603, DOI 10.17487/RFC8603, May 2019, <https://www.rfc-editor.org/rfc/rfc8603>.

[RFC8879] Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", RFC 8879, DOI 10.17487/RFC8879, December 2020, <https://www.rfc-editor.org/rfc/rfc8879>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[RFC9148] van der Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol", RFC 9148, DOI 10.17487/RFC9148, April 2022, <https://www.rfc-editor.org/rfc/rfc9148>.

[RFC9190] Preu Mattsson, J. and M. Sethi, "EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3", RFC 9190, DOI 10.17487/RFC9190, February 2022, <https://www.rfc-editor.org/rfc/rfc9190>.

[RFC9191] Sethi, M., Preu Mattsson, J., and S. Turner, "Handling Large Certificates and Long Certificate Chains in TLS-Based EAP Methods", RFC 9191, DOI 10.17487/RFC9191, February 2022, <https://www.rfc-editor.org/rfc/rfc9191>.

[RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <https://www.rfc-editor.org/rfc/rfc9528>.

[RFC9908] Richardson, M., Ed., Friel, O., von Oheimb, D., and D. Harkins, "Clarification and Enhancement of the CSR Attributes Definition in RFC 7030", RFC 9908, DOI 10.17487/RFC9908, January 2026, <https://www.rfc-editor.org/rfc/rfc9908>.

[SP-800-56A] Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography", NIST Special Publication 800-56A Revision 3, April 2018, <https://doi.org/10.6028/NIST.SP.800-56Ar3>.

[X.509-IoT] Forsby, F., Furuhed, M., Papadimitratos, P., and S. Raza, "Lightweight X.509 Digital Certificates for the Internet of Things.", Springer, Cham. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 242., July 2018, <https://doi.org/10.1007/978-3-319-93797-7_14>.

Appendix A. C509 Certificate Examples

A.1. Example: RFC 7925 profiled X.509 Certificate

Example of an [RFC7925] profiled X.509 certificate parsed with OpenSSL.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 128269 (0x1f50d)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: CN=RFC test CA
        Validity
            Not Before: Jan  1 00:00:00 2023 GMT
            Not After : Jan  1 00:00:00 2026 GMT
        Subject: CN=01-23-45-FF-FE-67-89-AB
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:b1:21:6a:b9:6e:5b:3b:33:40:f5:bd:f0:2e:69:
                    3f:16:21:3a:04:52:5e:d4:44:50:b1:01:9c:2d:fd:
                    38:38:ab:ac:4e:14:d8:6c:09:83:ed:5e:9e:ef:24:
                    48:c6:86:1c:c4:06:54:71:77:e6:02:60:30:d0:51:
                    f7:79:2a:c2:06
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage:
                Digital Signature
    Signature Algorithm: ecdsa-with-SHA256
        30:46:02:21:00:d4:32:0b:1d:68:49:e3:09:21:9d:30:03:7e:
        13:81:66:f2:50:82:47:dd:da:e7:6c:ce:ea:55:05:3c:10:8e:
        90:02:21:00:d5:51:f6:d6:01:06:f1:ab:b4:84:cf:be:62:56:
        c1:78:e4:ac:33:14:ea:19:19:1e:8b:60:7d:a5:ae:3b:da:16
```

The DER encoding of the above certificate is 316 bytes.

```
30 82 01 38 30 81 de a0 03 02 01 02 02 03 01 f5 0d 30 0a 06 08 2a 86
48 ce 3d 04 03 02 30 16 31 14 30 12 06 03 55 04 03 0c 0b 52 46 43 20
74 65 73 74 20 43 41 30 1e 17 0d 32 33 30 31 30 31 30 30 30 30 30 30
5a 17 0d 32 36 30 31 30 31 30 30 30 30 30 30 5a 30 22 31 20 30 1e 06
03 55 04 03 0c 17 30 31 2d 32 33 2d 34 35 2d 46 46 2d 46 45 2d 36 37
2d 38 39 2d 41 42 30 59 30 13 06 07 2a 86 48 ce 3d 02 01 06 08 2a 86
48 ce 3d 03 01 07 03 42 00 04 b1 21 6a b9 6e 5b 3b 33 40 f5 bd f0 2e
69 3f 16 21 3a 04 52 5e d4 44 50 b1 01 9c 2d fd 38 38 ab ac 4e 14 d8
6c 09 83 ed 5e 9e ef 24 48 c6 86 1c c4 06 54 71 77 e6 02 60 30 d0 51
f7 79 2a c2 06 a3 0f 30 0d 30 0b 06 03 55 1d 0f 04 04 03 02 07 80 30
0a 06 08 2a 86 48 ce 3d 04 03 02 03 49 00 30 46 02 21 00 d4 32 0b 1d
68 49 e3 09 21 9d 30 03 7e 13 81 66 f2 50 82 47 dd da e7 6c ce ea 55
05 3c 10 8e 90 02 21 00 d5 51 f6 d6 01 06 f1 ab b4 84 cf be 62 56 c1
78 e4 ac 33 14 ea 19 19 1e 8b 60 7d a5 ae 3b da 16
```

A.1.1. Example: C509 Certificate Encoding

This section shows the C509 encoding of the X.509 certificate in the previous section. The point-compressed public key is represented as described in Section 3.2.1.

Figure 21 shows the diagnostic notation of the unwrapped CBOR sequence, ~C509Certificate, see Section 3.1.

/This defines a CBOR Sequence (RFC 8742):/

```
  3,                    / version and certificate type /
  h'01f50d',            / certificateSerialNumber /
  0,                    / signatureAlgorithm /
  "RFC test CA",        / issuer /
  1672531200,           / notBefore /
  1767225600,           / notAfter /
  48(h'0123456789AB'), / subject, EUI-64 /
  1,                    / subjectPublicKeyAlgorithm /
  h'FEB1216AB96E5B3B3340F5BDF02E693F16213A04525ED44450
    B1019C2DFD3838AB',
  1,                    / single extension:
                          non-critical keyUsage
                          digitalSignature /
  h'D4320B1D6849E309219D30037E138166F2508247DDDAE76CCE
    EA55053C108E90D551F6D60106F1ABB484CFBE6256C178E4AC
    3314EA19191E8B607DA5AE3BDA16'
```

Figure 21: CBOR diagnostic notation of ~C509Certificate

Figure 22 shows the plain hex format of the unwrapped CBOR sequence. The size is 140 bytes.

```
03
43 01 F5 0D
00
6B 52 46 43 20 74 65 73 74 20 43 41
1A 63 B0 CD 00
1A 69 55 B9 00
D8 30 46 01 23 45 67 89 AB
01
58 21 FE B1 21 6A B9 6E 5B 3B 33 40 F5 BD F0 2E 69 3F 16 21 3A 04 52
5E D4 44 50 B1 01 9C 2D FD 38 38 AB
01
58 40 D4 32 0B 1D 68 49 E3 09 21 9D 30 03 7E 13 81 66 F2 50 82 47 DD
DA E7 6C CE EA 55 05 3C 10 8E 90 D5 51 F6 D6 01 06 F1 AB B4 84 CF BE
62 56 C1 78 E4 AC 33 14 EA 19 19 1E 8B 60 7D A5 AE 3B DA 16
```

Figure 22: CBOR plain hex format of ~C509Certificate.

A.1.2. Example: Natively Signed C509 Certificate

This section shows the natively signed C509 certificate corresponding to the certificate in the previous section. It is identical except for c509CertificateType, the encoding of point compression (see Section 3.2.1), and signatureValue.

Figure 23 shows the diagnostic notation of the natively signed unwrapped CBOR sequence, ~C509Certificate.

/This defines a CBOR Sequence (RFC 8742):/

```
2,
h'01f50d',
0,
"RFC test CA",
1672531200,
1767225600,
48(h'0123456789AB'),
1,
h'02B1216AB96E5B3B3340F5BDF02E693F16213A04525ED44450
  B1019C2DFD3838AB',
1,
h'EB0D472731F689BC00F5880B12C68B3F9FD38B23FADFCA2095
  0F3F241B60A202579CAC28CD3B7494D5FA5D8BBAB4600357E5
  50AB9FA9A65D9BA2B3B82E668CC6'
```

Figure 23: CBOR diagnostic notation of ~C509Certificate

Figure 24 shows the plain hex format of the natively signed unwrapped CBOR sequence. The size is 140 bytes.

```
02
43 01 F5 0D
00
6B 52 46 43 20 74 65 73 74 20 43 41
1A 63 B0 CD 00
1A 69 55 B9 00
D8 30 46 01 23 45 67 89 AB
01
58 21 02 B1 21 6A B9 6E 5B 3B 33 40 F5 BD F0 2E 69 3F 16 21 3A 04 52
5E D4 44 50 B1 01 9C 2D FD 38 38 AB
01
58 40 EB 0D 47 27 31 F6 89 BC 00 F5 88 0B 12 C6 8B 3F 9F D3 8B 23 FA
DF CA 20 95 0F 3F 24 1B 60 A2 02 57 9C AC 28 CD 3B 74 94 D5 FA 5D 8B
BA B4 60 03 57 E5 50 AB 9F A9 A6 5D 9B A2 B3 B8 2E 66 8C C6
```

Figure 24: CBOR plain hex format of ~C509Certificate.

A.1.3. C509 for Diffie-Hellman keys

The two previous examples illustrate keyUsage digitalSignature. A C509 certificate for a public Diffie-Hellman key would instead have key usage keyAgreement encoded according to Section 3.3 (in this case of single extension encoded as integer 16 instead of 1 for digital signature) but otherwise identical in format. Note that Section 5.6.3.2 of [SP-800-56A] allows a key agreement key pair to be used to sign a certification request.

A.1.4. Example: Additional Keys for the Example Certificates

Below are the issuer key pair and the subject private key corresponding to the above example certificates. The private keys are encoded as in COSE [RFC9052]. This issuer key pair can be used to sign or verify the example certificates, and the subject private key allows those certificates to be used in test vectors for other protocols such as EDHOC.

issuerPublicKeyAlgorithm :
1 (EC Public Key (Weierstrass) with secp256r1)

issuerPublicKey :
h’02AE4CDB01F614DEFC7121285FDC7F5C6D1D42C95647F061BA0080DF678867845E’

issuerPrivateKey :
h’DC66B3415456D649429B53223DF7532B942D6B0E0842C30BCA4C0ACF91547BB2’

subjectPrivateKey :
h’D718111F3F9BD91B92FF6877F386BDBFCEA7154268FD7F2FB56EE17D99EA16D4’

A.1.5. Examples: C509Certificate and C509CertData

This section exemplifies other CBOR objects defined in this specification, based on the natively signed C509 certificate in Appendix A.1.2.

Figure 25 shows the encoding of the corresponding C509Certificate, i.e., the CBOR array wrapping of the CBOR sequence ~C509Certificate, see Section 3.1.

```
8B
02
43 01 F5 0D
00
6B 52 46 43 20 74 65 73 74 20 43 41
1A 63 B0 CD 00
1A 69 55 B9 00
D8 30 46 01 23 45 67 89 AB
01
58 21 02 B1 21 6A B9 6E 5B 3B 33 40 F5 BD F0 2E 69 3F 16 21 3A 04 52
5E D4 44 50 B1 01 9C 2D FD 38 38 AB
01
58 40 EB 0D 47 27 31 F6 89 BC 00 F5 88 0B 12 C6 8B 3F 9F D3 8B 23 FA
DF CA 20 95 0F 3F 24 1B 60 A2 02 57 9C AC 28 CD 3B 74 94 D5 FA 5D 8B
BA B4 60 03 57 E5 50 AB 9F A9 A6 5D 9B A2 B3 B8 2E 66 8C C6
```

Figure 25: C509Certificate: The CBOR array wrapping of ~C509Certificate

Note that C509Certificate is identical to ~C509Certificate in Appendix A.1.2 except for the prefix 8B (which indicates that it is a CBOR array with 11 elements).

Figure 26 shows the encoding of the corresponding C509CertData, i.e., the CBOR byte string wrapping of the CBOR sequence ~C509Certificate, see Section 3.4.

```
58 8C
02
43 01 F5 0D
00
6B 52 46 43 20 74 65 73 74 20 43 41
1A 63 B0 CD 00
1A 69 55 B9 00
D8 30 46 01 23 45 67 89 AB
01
58 21 02 B1 21 6A B9 6E 5B 3B 33 40 F5 BD F0 2E 69 3F 16 21 3A 04 52
5E D4 44 50 B1 01 9C 2D FD 38 38 AB
01
58 40 EB 0D 47 27 31 F6 89 BC 00 F5 88 0B 12 C6 8B 3F 9F D3 8B 23 FA
DF CA 20 95 0F 3F 24 1B 60 A2 02 57 9C AC 28 CD 3B 74 94 D5 FA 5D 8B
BA B4 60 03 57 E5 50 AB 9F A9 A6 5D 9B A2 B3 B8 2E 66 8C C6
```

Figure 26: C509CertData: CBOR byte string wrapping of ~C509Certificate.

Note that C509CertData is identical to ~C509Certificate in Appendix A.1.2 except for the prefix 58 8C (which indicates that it is a CBOR byte string of 140 bytes).

A.2. Example: IEEE 802.1AR profiled X.509 Certificate

An example of an IEEE 802.1AR profiled X.509 certificate (Secure Device Identifier, DevID) is provided in Appendix C.2 of [RFC9148]. The certificate is shown below including details of the hardwareModuleName type of otherName in subjectAltName, see Section 3.3.

```
   Certificate:
     Data:
       Version: 3 (0x2)
       Serial Number: 9112578475118446130 (0x7e7661d7b54e4632)
       Signature Algorithm: ecdsa-with-SHA256
       Issuer: C=US, ST=CA, O=Example Inc, OU=certification,
               CN=802.1AR CA
       Validity
         Not Before: Jan 31 11:29:16 2019 GMT
         Not After : Dec 31 23:59:59 9999 GMT
       Subject: C=US, ST=CA, L=LA, O=example Inc,
                OU=IoT/serialNumber=Wt1234
       Subject Public Key Info:
         Public Key Algorithm: id-ecPublicKey
           Public-Key: (256 bit)
           pub:
             04:c8:b4:21:f1:1c:25:e4:7e:3a:c5:71:23:bf:2d:
             9f:dc:49:4f:02:8b:c3:51:cc:80:c0:3f:15:0b:f5:
             0c:ff:95:8d:75:41:9d:81:a6:a2:45:df:fa:e7:90:
             be:95:cf:75:f6:02:f9:15:26:18:f8:16:a2:b2:3b:
             56:38:e5:9f:d9
             ASN1 OID: prime256v1
             NIST CURVE: P-256
       X509v3 extensions:
         X509v3 Basic Constraints:
           CA:FALSE
         X509v3 Subject Key Identifier:
           96:60:0D:87:16:BF:7F:D0:E7:52:D0:AC:76:07:77:AD:66:5D:02:A0
         X509v3 Authority Key Identifier:
           68:D1:65:51:F9:51:BF:C8:2A:43:1D:0D:9F:08:BC:2D:20:5B:11:60
         X509v3 Key Usage: critical
           Digital Signature, Key Encipherment
         X509v3 Subject Alternative Name:
           otherName:
             type-id: 1.3.6.1.5.5.7.8.4 (id-on-hardwareModuleName)
             value:
               hwType: 1.3.6.1.4.1.6715.10.1
               hwSerialNum: 01:02:03:04
     Signature Algorithm: ecdsa-with-SHA256
     Signature Value:
       30:46:02:21:00:c0:d8:19:96:d2:50:7d:69:3f:3c:48:ea:a5:
       ee:94:91:bd:a6:db:21:40:99:d9:81:17:c6:3b:36:13:74:cd:
       86:02:21:00:a7:74:98:9f:4c:32:1a:5c:f2:5d:83:2a:4d:33:
       6a:08:ad:67:df:20:f1:50:64:21:18:8a:0a:de:6d:34:92:36
```

The DER encoding of the certificate is 577 bytes:

```
30 82 02 3D 30 82 01 E2 A0 03 02 01 02 02 08 7E 76 61 D7 B5 4E 46 32
30 0A 06 08 2A 86 48 CE 3D 04 03 02 30 5D 31 0B 30 09 06 03 55 04 06
13 02 55 53 31 0B 30 09 06 03 55 04 08 0C 02 43 41 31 14 30 12 06 03
55 04 0A 0C 0B 45 78 61 6D 70 6C 65 20 49 6E 63 31 16 30 14 06 03 55
04 0B 0C 0D 63 65 72 74 69 66 69 63 61 74 69 6F 6E 31 13 30 11 06 03
55 04 03 0C 0A 38 30 32 2E 31 41 52 20 43 41 30 20 17 0D 31 39 30 31
33 31 31 31 32 39 31 36 5A 18 0F 39 39 39 39 31 32 33 31 32 33 35 39
35 39 5A 30 5C 31 0B 30 09 06 03 55 04 06 13 02 55 53 31 0B 30 09 06
03 55 04 08 0C 02 43 41 31 0B 30 09 06 03 55 04 07 0C 02 4C 41 31 14
30 12 06 03 55 04 0A 0C 0B 65 78 61 6D 70 6C 65 20 49 6E 63 31 0C 30
0A 06 03 55 04 0B 0C 03 49 6F 54 31 0F 30 0D 06 03 55 04 05 13 06 57
74 31 32 33 34 30 59 30 13 06 07 2A 86 48 CE 3D 02 01 06 08 2A 86 48
CE 3D 03 01 07 03 42 00 04 C8 B4 21 F1 1C 25 E4 7E 3A C5 71 23 BF 2D
9F DC 49 4F 02 8B C3 51 CC 80 C0 3F 15 0B F5 0C FF 95 8D 75 41 9D 81
A6 A2 45 DF FA E7 90 BE 95 CF 75 F6 02 F9 15 26 18 F8 16 A2 B2 3B 56
38 E5 9F D9 A3 81 8A 30 81 87 30 09 06 03 55 1D 13 04 02 30 00 30 1D
06 03 55 1D 0E 04 16 04 14 96 60 0D 87 16 BF 7F D0 E7 52 D0 AC 76 07
77 AD 66 5D 02 A0 30 1F 06 03 55 1D 23 04 18 30 16 80 14 68 D1 65 51
F9 51 BF C8 2A 43 1D 0D 9F 08 BC 2D 20 5B 11 60 30 0E 06 03 55 1D 0F
01 01 FF 04 04 03 02 05 A0 30 2A 06 03 55 1D 11 04 23 30 21 A0 1F 06
08 2B 06 01 05 05 07 08 04 A0 13 30 11 06 09 2B 06 01 04 01 B4 3B 0A
01 04 04 01 02 03 04 30 0A 06 08 2A 86 48 CE 3D 04 03 02 03 49 00 30
46 02 21 00 C0 D8 19 96 D2 50 7D 69 3F 3C 48 EA A5 EE 94 91 BD A6 DB
21 40 99 D9 81 17 C6 3B 36 13 74 CD 86 02 21 00 A7 74 98 9F 4C 32 1A
5C F2 5D 83 2A 4D 33 6A 08 AD 67 DF 20 F1 50 64 21 18 8A 0A DE 6D 34
92 36
```

A.2.1. Example: C509 Certificate Encoding

The CBOR encoding (~C509Certificate) of the same X.509 certificate is shown below in CBOR diagnostic format.

```
/This defines a CBOR Sequence (RFC 8742):/

 3,
 h'7E7661D7B54E4632',
 0,
 [
  -4, "US",
   6, "CA",
   8, "Example Inc",
   9, "certification",
   1, "802.1AR CA"
 ],
 1548934156,
 null,
 [
  -4, "US",
   6, "CA",
   5, "LA",
   8, "example Inc",
   9, "IoT",
  -3, "Wt1234"
 ],
 1,
 h'FDC8B421F11C25E47E3AC57123BF2D9FDC494F028BC351CC80C03F150BF50CFF
   95',
 [
   4, -2,
   1, h'96600D8716BF7FD0E752D0AC760777AD665D02A0',
   7, h'68D16551F951BFC82A431D0D9F08BC2D205B1160',
  -2, 5,
  3, [-1, [h'2B06010401B43B0A01', h'01020304']]
     / subjectAltName with hardwareModuleName /
 ],
 h'C0D81996D2507D693F3C48EAA5EE9491BDA6DB214099D98117C63B361374CD86
   A774989F4C321A5CF25D832A4D336A08AD67DF20F1506421188A0ADE6D349236'
```

The size of the CBOR encoding (CBOR sequence) is 275 bytes:

```
03 48 7E 76 61 D7 B5 4E 46 32 00 8A 23 62 55 53 06 62 43 41 08 6B 45
78 61 6D 70 6C 65 20 49 6E 63 09 6D 63 65 72 74 69 66 69 63 61 74 69
6F 6E 01 6A 38 30 32 2E 31 41 52 20 43 41 1A 5C 52 DC 0C F6 8C 23 62
55 53 06 62 43 41 05 62 4C 41 08 6B 65 78 61 6D 70 6C 65 20 49 6E 63
09 63 49 6F 54 22 66 57 74 31 32 33 34 01 58 21 FD C8 B4 21 F1 1C 25
E4 7E 3A C5 71 23 BF 2D 9F DC 49 4F 02 8B C3 51 CC 80 C0 3F 15 0B F5
0C FF 95 8A 04 21 01 54 96 60 0D 87 16 BF 7F D0 E7 52 D0 AC 76 07 77
AD 66 5D 02 A0 07 54 68 D1 65 51 F9 51 BF C8 2A 43 1D 0D 9F 08 BC 2D
20 5B 11 60 21 05 03 82 20 82 49 2B 06 01 04 01 B4 3B 0A 01 44 01 02
03 04 58 40 C0 D8 19 96 D2 50 7D 69 3F 3C 48 EA A5 EE 94 91 BD A6 DB
21 40 99 D9 81 17 C6 3B 36 13 74 CD 86 A7 74 98 9F 4C 32 1A 5C F2 5D
83 2A 4D 33 6A 08 AD 67 DF 20 F1 50 64 21 18 8A 0A DE 6D 34 92 36
```

A.3. Example: CAB Baseline ECDSA HTTPS X.509 Certificate

The www.ietf.org HTTPS server replies with a certificate message with 2 certificates. The DER encoding of the first certificate is 1209 bytes.

```
30 82 04 b5 30 82 04 5a a0 03 02 01 02 02 10 04 7f a1 e3 19 28 ee 40
3b a0 b8 3a 39 56 73 fc 30 0a 06 08 2a 86 48 ce 3d 04 03 02 30 4a 31
0b 30 09 06 03 55 04 06 13 02 55 53 31 19 30 17 06 03 55 04 0a 13 10
43 6c 6f 75 64 66 6c 61 72 65 2c 20 49 6e 63 2e 31 20 30 1e 06 03 55
04 03 13 17 43 6c 6f 75 64 66 6c 61 72 65 20 49 6e 63 20 45 43 43 20
43 41 2d 33 30 1e 17 0d 32 30 30 37 32 39 30 30 30 30 30 30 5a 17 0d
32 31 30 37 32 39 31 32 30 30 30 30 5a 30 6d 31 0b 30 09 06 03 55 04
06 13 02 55 53 31 0b 30 09 06 03 55 04 08 13 02 43 41 31 16 30 14 06
03 55 04 07 13 0d 53 61 6e 20 46 72 61 6e 63 69 73 63 6f 31 19 30 17
06 03 55 04 0a 13 10 43 6c 6f 75 64 66 6c 61 72 65 2c 20 49 6e 63 2e
31 1e 30 1c 06 03 55 04 03 13 15 73 6e 69 2e 63 6c 6f 75 64 66 6c 61
72 65 73 73 6c 2e 63 6f 6d 30 59 30 13 06 07 2a 86 48 ce 3d 02 01 06
08 2a 86 48 ce 3d 03 01 07 03 42 00 04 96 3e cd d8 4d cd 1b 93 a1 cf
43 2d 1a 72 17 d6 c6 3b de 33 55 a0 2f 8c fb 5a d8 99 4c d4 4e 20 5f
15 f6 e3 d2 3b 38 2b a6 49 9b b1 7f 34 1f a5 92 fa 21 86 1f 16 d3 12
06 63 24 05 fd 70 42 bd a3 82 02 fd 30 82 02 f9 30 1f 06 03 55 1d 23
04 18 30 16 80 14 a5 ce 37 ea eb b0 75 0e 94 67 88 b4 45 fa d9 24 10
87 96 1f 30 1d 06 03 55 1d 0e 04 16 04 14 cc 0b 50 e7 d8 37 db f2 43
f3 85 3d 48 60 f5 3b 39 be 9b 2a 30 2e 06 03 55 1d 11 04 27 30 25 82
15 73 6e 69 2e 63 6c 6f 75 64 66 6c 61 72 65 73 73 6c 2e 63 6f 6d 82
0c 77 77 77 2e 69 65 74 66 2e 6f 72 67 30 0e 06 03 55 1d 0f 01 01 ff
04 04 03 02 07 80 30 1d 06 03 55 1d 25 04 16 30 14 06 08 2b 06 01 05
05 07 03 01 06 08 2b 06 01 05 05 07 03 02 30 7b 06 03 55 1d 1f 04 74
30 72 30 37 a0 35 a0 33 86 31 68 74 74 70 3a 2f 2f 63 72 6c 33 2e 64
69 67 69 63 65 72 74 2e 63 6f 6d 2f 43 6c 6f 75 64 66 6c 61 72 65 49
6e 63 45 43 43 43 41 2d 33 2e 63 72 6c 30 37 a0 35 a0 33 86 31 68 74
74 70 3a 2f 2f 63 72 6c 34 2e 64 69 67 69 63 65 72 74 2e 63 6f 6d 2f
43 6c 6f 75 64 66 6c 61 72 65 49 6e 63 45 43 43 43 41 2d 33 2e 63 72
6c 30 4c 06 03 55 1d 20 04 45 30 43 30 37 06 09 60 86 48 01 86 fd 6c
```

```
01 01 30 2a 30 28 06 08 2b 06 01 05 05 07 02 01 16 1c 68 74 74 70 73
3a 2f 2f 77 77 77 2e 64 69 67 69 63 65 72 74 2e 63 6f 6d 2f 43 50 53
30 08 06 06 67 81 0c 01 02 02 30 76 06 08 2b 06 01 05 05 07 01 01 04
6a 30 68 30 24 06 08 2b 06 01 05 05 07 30 01 86 18 68 74 74 70 3a 2f
2f 6f 63 73 70 2e 64 69 67 69 63 65 72 74 2e 63 6f 6d 30 40 06 08 2b
06 01 05 05 07 30 02 86 34 68 74 74 70 3a 2f 2f 63 61 63 65 72 74 73
2e 64 69 67 69 63 65 72 74 2e 63 6f 6d 2f 43 6c 6f 75 64 66 6c 61 72
65 49 6e 63 45 43 43 43 41 2d 33 2e 63 72 74 30 0c 06 03 55 1d 13 01
01 ff 04 02 30 00 30 82 01 05 06 0a 2b 06 01 04 01 d6 79 02 04 02 04
81 f6 04 81 f3 00 f1 00 76 00 f6 5c 94 2f d1 77 30 22 14 54 18 08 30
94 56 8e e3 4d 13 19 33 bf df 0c 2f 20 0b cc 4e f1 64 e3 00 00 01 73
9c 83 5f 8e 00 00 04 03 00 47 30 45 02 21 00 f8 d1 b4 a9 3d 2f 0d 4c
41 76 df b4 88 bc c7 3b 86 44 3d 7d e0 0e 6a c8 17 4d 89 48 a8 84 36
68 02 20 29 ff 5a 34 06 8a 24 0c 69 50 27 88 e8 ee 25 ab 7e d2 cb cf
68 6e ce 7b 5f 96 b4 31 a9 07 02 fa 00 77 00 5c dc 43 92 fe e6 ab 45
44 b1 5e 9a d4 56 e6 10 37 fb d5 fa 47 dc a1 73 94 b2 5e e6 f6 c7 0e
ca 00 00 01 73 9c 83 5f be 00 00 04 03 00 48 30 46 02 21 00 e8 91 c1
97 bf b0 e3 d3 0c b6 ce e6 0d 94 c3 c7 5f d1 17 53 36 93 11 08 d8 98
12 d4 d2 9d 81 d0 02 21 00 a1 59 d1 6c 46 47 d1 48 37 57 fc d6 ce 4e
75 ec 7b 5e f6 57 ef e0 28 f8 e5 cc 47 92 68 2d ac 43 30 0a 06 08 2a
86 48 ce 3d 04 03 02 03 49 00 30 46 02 21 00 bd 63 cf 4f 7e 5c fe 6c
29 38 5e a7 1c fb fc 1e 3f 7b 1c d0 72 51 a2 21 f7 77 69 c0 f4 71 df
ea 02 21 00 b5 c0 6c c4 58 54 fa 30 b2 82 88 b1 d3 bb 9a 66 61 ed 50
31 72 5b 1a 82 02 e0 da 5b 59 f9 54 02
```

A.3.1. Example: C509 Certificate Encoding

The CBOR encoding (~C509Certificate) of the first X.509 certificate is shown below in CBOR diagnostic format.

```
/This defines a CBOR Sequence (RFC 8742):/

3,
h'047FA1E31928EE403BA0B83A395673FC',
0,
[
 -4, "US",
 -8, "Cloudflare, Inc.",
 -1, "Cloudflare Inc ECC CA-3"
],
1595980800,
1627560000,
[
 -4, "US",
 -6, "CA",
 -5, "San Francisco",
 -8, "Cloudflare, Inc.",
 -1, "sni.cloudflaressl.com"
```

```
   ],
   1,
   h'FD963ECDD84DCD1B93A1CF432D1A7217D6C63BDE3355A02F8CFB5AD8994CD44E
     20',
   [
    7, h'A5CE37EAEBB0750E946788B445FAD9241087961F',
    1, h'CC0B50E7D837DBF243F3853D4860F53B39BE9B2A',
    3, [2, "sni.cloudflaressl.com", 2, "www.ietf.org"],
   -2, 1,
    8, [1, 2],
    5, [
        ["http://crl3.digicert.com/CloudflareIncECCCA-3.crl",
          null, null],
        ["http://crl4.digicert.com/CloudflareIncECCCA-3.crl",
          null, null]
       ],
    6, [h'6086480186FD6C0101', [1, "https://www.digicert.com/CPS"],
        2, []],
    9, [1, "http://ocsp.digicert.com",
        2, "http://cacerts.digicert.com/CloudflareIncECCCA-3.crt"],
   -4, -2,
    h'2B06010401D679020402',
    h'0481F300F1007600F65C942FD1773022145418083094568EE34D131933BFDF0C
      2F200BCC4EF164E3000001739C835F8E00000403004730450221 00F8D1B4A93D
      2F0D4C4176DFB488BCC73B86443D7DE00E6AC8174D8948A8843668022029FF5A
      34068A240C69502788E8EE25AB7ED2CBCF686ECE7B5F96B431A90702FA007700
      5CDC4392FEE6AB4544B15E9AD456E61037FBD5FA47DCA17394B25EE6F6C70ECA
      000001739C835FBE00000403004830460221 00E891C197BFB0E3D30CB6CEE60D
      94C3C75FD1175336931108D89812D4D29D81D0022100A159D16C4647D1483757
      FCD6CE4E75EC7B5EF657EFE028F8E5CC4792682DAC43'
   ],
   h'BD63CF4F7E5CFE6C29385EA71CFBFC1E3F7B1CD07251A221F77769C0F471DFEA
     B5C06CC45854FA30B28288B1D3BB9A6661ED5031725B1A8202E0DA5B59F95402'
```

The size of the CBOR encoding (CBOR sequence) is 835 bytes.

A.4. Example: CAB Baseline RSA HTTPS X.509 Certificate

The tools.ietf.org HTTPS server replies with a certificate message with 4 certificates. The DER encoding of the first certificate is 1647 bytes.

```
30 82 06 6b 30 82 05 53 a0 03 02 01 02 02 09 00 a6 a5 5c 87 0e 39 b4
0e 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b 05 00 30 81 c6 31 0b 30 09
06 03 55 04 06 13 02 55 53 31 10 30 0e 06 03 55 04 08 13 07 41 72 69
7a 6f 6e 61 31 13 30 11 06 03 55 04 07 13 0a 53 63 6f 74 74 73 64 61
6c 65 31 25 30 23 06 03 55 04 0a 13 1c 53 74 61 72 66 69 65 6c 64 20
54 65 63 68 6e 6f 6c 6f 67 69 65 73 2c 20 49 6e 63 2e 31 33 30 31 06
```

```
03 55 04 0b 13 2a 68 74 74 70 3a 2f 2f 63 65 72 74 73 2e 73 74 61 72
66 69 65 6c 64 74 65 63 68 2e 63 6f 6d 2f 72 65 70 6f 73 69 74 6f 72
79 2f 31 34 30 32 06 03 55 04 03 13 2b 53 74 61 72 66 69 65 6c 64 20
53 65 63 75 72 65 20 43 65 72 74 69 66 69 63 61 74 65 20 41 75 74 68
6f 72 69 74 79 20 2d 20 47 32 30 1e 17 0d 32 30 31 30 30 31 31 39 33
38 33 36 5a 17 0d 32 31 31 31 30 32 31 39 33 38 33 36 5a 30 3e 31 21
30 1f 06 03 55 04 0b 13 18 44 6f 6d 61 69 6e 20 43 6f 6e 74 72 6f 6c
20 56 61 6c 69 64 61 74 65 64 31 19 30 17 06 03 55 04 03 0c 10 2a 2e
74 6f 6f 6c 73 2e 69 65 74 66 2e 6f 72 67 30 82 01 22 30 0d 06 09 2a
86 48 86 f7 0d 01 01 01 05 00 03 82 01 0f 00 30 82 01 0a 02 82 01 01
00 b1 e1 37 e8 eb 82 d6 89 fa db f5 c2 4b 77 f0 2c 4a de 72 6e 3e 13
60 d1 a8 66 1e c4 ad 3d 32 60 e5 f0 99 b5 f4 7a 7a 48 55 21 ee 0e 39
12 f9 ce 0d ca f5 69 61 c7 04 ed 6e 0f 1d 3b 1e 50 88 79 3a 0e 31 41
16 f1 b1 02 64 68 a5 cd f5 4a 0a ca 99 96 35 08 c3 7e 27 5d d0 a9 cf
f3 e7 28 af 37 d8 b6 7b dd f3 7e ae 6e 97 7f f7 ca 69 4e cc d0 06 df
5d 27 9b 3b 12 e7 e6 fe 08 6b 52 7b 82 11 7c 72 b3 46 eb c1 e8 78 b8
0f cb e1 eb bd 06 44 58 dc 83 50 b2 a0 62 5b dc 81 b8 36 e3 9e 7c 79
b2 a9 53 8a e0 0b c9 4a 2a 13 39 31 13 bd 2c cf a8 70 cf 8c 8d 3d 01
a3 88 ae 12 00 36 1d 1e 24 2b dd 79 d8 53 01 26 ed 28 4f c9 86 94 83
4e c8 e1 14 2e 85 b3 af d4 6e dd 69 46 af 41 25 0e 7a ad 8b f2 92 ca
79 d9 7b 32 4f f7 77 e8 f9 b4 4f 23 5c d4 5c 03 ae d8 ab 3a ca 13 5f
5d 5d 5d a1 02 03 01 00 01 a3 82 02 e1 30 82 02 dd 30 0c 06 03 55 1d
13 01 01 ff 04 02 30 00 30 1d 06 03 55 1d 25 04 16 30 14 06 08 2b 06
01 05 05 07 03 01 06 08 2b 06 01 05 05 07 03 02 30 0e 06 03 55 1d 0f
01 01 ff 04 04 03 02 05 a0 30 3d 06 03 55 1d 1f 04 36 30 34 30 32 a0
30 a0 2e 86 2c 68 74 74 70 3a 2f 2f 63 72 6c 2e 73 74 61 72 66 69 65
6c 64 74 65 63 68 2e 63 6f 6d 2f 73 66 69 67 32 73 31 2d 32 34 32 2e
63 72 6c 30 63 06 03 55 1d 20 04 5c 30 5a 30 4e 06 0b 60 86 48 01 86
fd 6e 01 07 17 01 30 3f 30 3d 06 08 2b 06 01 05 05 07 02 01 16 31 68
74 74 70 3a 2f 2f 63 65 72 74 69 66 69 63 61 74 65 73 2e 73 74 61 72
66 69 65 6c 64 74 65 63 68 2e 63 6f 6d 2f 72 65 70 6f 73 69 74 6f 72
79 2f 30 08 06 06 67 81 0c 01 02 01 30 81 82 06 08 2b 06 01 05 05 07
01 01 04 76 30 74 30 2a 06 08 2b 06 01 05 05 07 30 01 86 1e 68 74 74
70 3a 2f 2f 6f 63 73 70 2e 73 74 61 72 66 69 65 6c 64 74 65 63 68 2e
63 6f 6d 2f 30 46 06 08 2b 06 01 05 05 07 30 02 86 3a 68 74 74 70 3a
2f 2f 63 65 72 74 69 66 69 63 61 74 65 73 2e 73 74 61 72 66 69 65 6c
64 74 65 63 68 2e 63 6f 6d 2f 72 65 70 6f 73 69 74 6f 72 79 2f 73 66
69 67 32 2e 63 72 74 30 1f 06 03 55 1d 23 04 18 30 16 80 14 25 45 81
68 50 26 38 3d 3b 2d 2c be cd 6a d9 b6 3d b3 66 63 30 2b 06 03 55 1d
11 04 24 30 22 82 10 2a 2e 74 6f 6f 6c 73 2e 69 65 74 66 2e 6f 72 67
82 0e 74 6f 6f 6c 73 2e 69 65 74 66 2e 6f 72 67 30 1d 06 03 55 1d 0e
04 16 04 14 ad 8a b4 1c 07 51 d7 92 89 07 b0 b7 84 62 2f 36 55 7a 5f
4d 30 82 01 06 06 0a 2b 06 01 04 01 d6 79 02 04 02 04 81 f7 04 81 f4
00 f2 00 77 00 f6 5c 94 2f d1 77 30 22 14 54 18 08 30 94 56 8e e3 4d
13 19 33 bf df 0c 2f 20 0b cc 4e f1 64 e3 00 00 01 74 e5 ac 71 13 00
00 04 03 00 48 30 46 02 21 00 8c f5 48 52 ce 56 35 43 39 11 cf 10 cd
b9 1f 52 b3 36 39 22 3a d1 38 a4 1d ec a6 fe de 1f e9 0f 02 21 00 bc
a2 25 43 66 c1 9a 26 91 c4 7a 00 b5 b6 53 ab bd 44 c2 f8 ba ae f4 d2
```

```
da f2 52 7c e6 45 49 95 00 77 00 5c dc 43 92 fe e6 ab 45 44 b1 5e 9a
d4 56 e6 10 37 fb d5 fa 47 dc a1 73 94 b2 5e e6 f6 c7 0e ca 00 00 01
74 e5 ac 72 3c 00 00 04 03 00 48 30 46 02 21 00 a5 e0 90 6e 63 e9 1d
4f dd ef ff 03 52 b9 1e 50 89 60 07 56 4b 44 8a 38 28 f5 96 dc 6b 28
72 6d 02 21 00 fc 91 ea ed 02 16 88 66 05 4e e1 8a 2e 53 46 c4 cc 51
fe b3 fa 10 a9 1d 2e db f9 91 25 f8 6c e6 30 0d 06 09 2a 86 48 86 f7
0d 01 01 0b 05 00 03 82 01 01 00 14 04 3f a0 be d2 ee 3f a8 6e 3a 1f
78 8e a0 4c 35 53 0f 11 06 1f ff 60 a1 6d 0b 83 e9 d9 2a db b3 3f 9d
b3 d7 e0 59 4c 19 a8 e4 19 a5 0c a7 70 72 77 63 d5 fe 64 51 0a d2 7a
d6 50 a5 8a 92 38 ec cb 2f 0f 5a c0 64 58 4d 5c 06 b9 73 63 68 27 8b
89 34 dc 79 c7 1d 3a fd 34 5f 83 14 41 58 49 80 68 29 80 39 8a 86 72
69 cc 79 37 ce e3 97 f7 dc f3 95 88 ed 81 03 29 00 d2 a2 c7 ba ab d6
3a 8e ca 09 0b d9 fb 39 26 4b ff 03 d8 8e 2d 3f 6b 21 ca 8a 7d d8 5f
fb 94 ba 83 de 9c fc 15 8d 61 fa 67 2d b0 c7 db 3d 25 0a 41 4a 85 d3
7f 49 46 37 3c f4 b1 75 d0 52 f3 dd c7 66 f1 4b fd aa 00 ed bf e4 7e
ed 01 ec 7b e4 f6 46 fc 31 fd 72 fe 03 d2 f2 65 af 4d 7e e2 81 9b 7a
fd 30 3c f5 52 f4 05 34 a0 8a 3e 19 41 58 c8 a8 e0 51 71 84 09 15 ae
ec a5 77 75 fa 18 f7 d5 77 d5 31 cc c7 2d
```

A.4.1. Example: C509 Certificate Encoding

The CBOR encoding (~C509Certificate) of the first X.509 certificate is shown below in CBOR diagnostic format.

/This defines a CBOR Sequence (RFC 8742):/

```
3,
h'A6A55C870E39B40E',
23,
[
 -4, "US",
 -6, "Arizona",
 -5, "Scottsdale",
 -8, "Starfield Technologies, Inc.",
 -9, "http://certs.starfieldtech.com/repository/",
 -1, "Starfield Secure Certificate Authority - G2"
],
1601581116,
1635881916,
[
  -9, "Domain Control Validated",
   1, "*.tools.ietf.org"
],
0,
h'B1E137E8EB82D689FADBF5C24B77F02C4ADE726E3E1360D1A8661EC4AD3D3260
  E5F099B5F47A7A485521EE0E3912F9CE0DCAF56961C704ED6E0F1D3B1E508879
  3A0E314116F1B1026468A5CDF54A0ACA99963508C37E275DD0A9CFF3E728AF37
  D8B67BDDF37EAE6E977FF7CA694ECCD006DF5D279B3B12E7E6FE086B527B8211
```

```
     7C72B346EBC1E878B80FCBE1EBBD064458DC8350B2A0625BDC81B836E39E7C79
     B2A9538AE00BC94A2A13393113BD2CCFA870CF8C8D3D01A388AE1200361D1E24
     2BDD79D8530126ED284FC98694834EC8E1142E85B3AFD46EDD6946AF41250E7A
     AD8BF292CA79D97B324FF777E8F9B44F235CD45C03AED8AB3ACA135F5D5D5DA1',
   [
   -4, -2,
    8, [ 1, 2 ],
    -2, 5,
    5, "http://crl.starfieldtech.com/sfig2s1-242.crl",
    6, [ h'6086480186FD6E01071701',
        [1, "http://certificates.starfieldtech.com/repository/"],
        1,
        []
       ],
    9, [ 1, "http://ocsp.starfieldtech.com/", 2,
         "http://certificates.starfieldtech.com/repository/sfig2.crt"],
    7, h'254581685026383D3B2D2CBECD6AD9B63DB36663',
    3, [ 2, "*.tools.ietf.org", 2, "tools.ietf.org" ],
    1, h'AD8AB41C0751D7928907B0B784622F36557A5F4D',
    h'2B06010401D679020402',
    h'0481F400F2007700F65C942FD1773022145418083094568EE34D131933BFDF0C
      2F200BCC4EF164E300000174E5AC7113000004030048304602210 08CF54852CE
      5635433911CF10CDB91F52B33639223AD138A41DECA6FEDE1FE90F022100BCA2
      254366C19A2691C47A00B5B653ABBD44C2F8BAAEF4D2DAF2527CE64549950077
      005CDC4392FEE6AB4544B15E9AD456E61037FBD5FA47DCA17394B25EE6F6C70E
      CA00000174E5AC723C000004030048304602210 0A5E0906E63E91D4FDDEFFF03
      52B91E50896007564B448A3828F596DC6B28726D022100FC91EAED0216886605
      4EE18A2E5346C4CC51FEB3FA10A91D2EDBF99125F86CE6'
   ],
   h'14043FA0BED2EE3FA86E3A1F788EA04C35530F11061FFF60A16D0B83E9D92ADB
     B33F9DB3D7E0594C19A8E419A50CA770727763D5FE64510AD27AD650A58A9238
     ECCB2F0F5AC064584D5C06B9736368278B8934DC79C71D3AFD345F8314415849
     80682980398A867269CC7937CEE397F7DCF39588ED81032900D2A2C7BAABD63A
     8ECA090BD9FB39264BFF03D88E2D3F6B21CA8A7DD85FFB94BA83DE9CFC158D61
     FA672DB0C7DB3D250A414A85D37F4946373CF4B175D052F3DDC766F14BFDAA00
     EDBFE47EED01EC7BE4F646FC31FD72FE03D2F265AF4D7EE2819B7AFD303CF552
     F40534A08A3E194158C8A8E05171840915AEECA57775FA18F7D577D531CCC72D'
```

The size of the CBOR encoding (CBOR sequence) is 1295 bytes.

A.5. Example: Certificate with Extensions IPAddrBlocks and IPAddrBlocksV2

An example X.509 certificate with extensions IPAddrBlocks and IPAddrBlocksV2.

```
   Certificate:
     Version: v3 (2)
     Serial Number:
       12:34
     Issuer: CN=selfsign-brainpoolp384r1,SURNAME=my surname,T=my title,
             GIVENNAME=my givenName,Name=my name
     Validity:
       Not Before: Thu Jan 02 01:00:00 CET 2025
       Not After : Fri Jan 02 01:00:00 CET 2026
     Subject: CN=selfsign-brainpoolp384r1,SURNAME=my surname,T=my title
              ,GIVENNAME=my givenName,Name=my name
     Subject Public Key Info:
       Public Key Algorithm: EC/BRAINPOOLP384R1
       Pub:
         04:67:09:c9:92:91:9b:49:c4:8f:d9:31:d0:5c:49:7d:38:65:
         e6:08:4c:91:df:3a:4c:7e:78:1f:41:85:43:b0:23:d5:9e:8b:
         f2:5d:13:3f:b1:a0:94:e9:d4:2c:8f:a6:ed:3b:46:e9:88:3a:
         35:ab:d4:b0:a9:d3:0a:ae:fd:9b:7e:88:ed:38:00:56:5d:1e:
         7f:06:33:13:4d:65:19:29:2d:49:bd:55:ec:30:a1:67:19:7f:
         ec:0f:74:29:82:2b:95
     X509v3 extensions:
       X509v3 keyUsage:
         digitalSignature
       X509v3 sbgp-ipAddrBlock:
         IPv4:
           192.0.2.0/24
           198.51.100.0/28
           203.0.113.0/24
         IPv6:
           2001:db8:1234::/48
           3fff:600:: - 3fff:fff:ffff:ffff:ffff:ffff:ffff:ffff
       X509v3 sbgp-ipAddrBlockV2:
         IPv4 unicast:
           192.0.2.0/24
           198.51.100.0/28
           203.0.113.0/24
         IPv6 unicast:
           2001:db8:1234::/48
           3fff:3:: - 3fff:122:0:2233:3344:5566:ffff:ffff
     Signature Algorithm: SHA384WITHECDSA
     Signature Value:
       30:64:02:30:67:09:c9:92:91:9b:49:c4:8f:d9:31:d0:5c:49:
       7d:38:65:e6:08:4c:91:df:3a:4c:7e:78:1f:41:85:43:b0:23:
       d5:9e:8b:f2:5d:13:3f:b1:a0:94:e9:d4:2c:8f:a6:ed:02:30:
       20:ed:9f:db:5a:30:9b:2c:87:04:dd:a5:f1:44:f1:7b:b3:16:
       b9:8c:29:11:24:fb:a5:cf:ec:6e:f9:7f:26:88:06:9a:e6:c5:
       2e:2b:3c:e2:23:12:8d:d1:0c:2a:a7:30
```

The DER encoding of the certificate is 717 bytes:

```
30 82 02 c9 30 82 02 50 a0 03 02 01 02 02 02 12 34 30 0a 06 08 2a 86
48 ce 3d 04 03 03 30 74 31 21 30 1f 06 03 55 04 03 0c 18 73 65 6c 66
73 69 67 6e 2d 62 72 61 69 6e 70 6f 6f 6c 70 33 38 34 72 31 31 13 30
11 06 03 55 04 04 0c 0a 6d 79 20 73 75 72 6e 61 6d 65 31 11 30 0f 06
03 55 04 0c 0c 08 6d 79 20 74 69 74 6c 65 31 15 30 13 06 03 55 04 2a
0c 0c 6d 79 20 67 69 76 65 6e 4e 61 6d 65 31 10 30 0e 06 03 55 04 29
0c 07 6d 79 20 6e 61 6d 65 30 1e 17 0d 32 35 30 31 30 32 30 30 30 30
30 30 5a 17 0d 32 36 30 31 30 32 30 30 30 30 30 30 5a 30 74 31 21 30
1f 06 03 55 04 03 0c 18 73 65 6c 66 73 69 67 6e 2d 62 72 61 69 6e 70
6f 6f 6c 70 33 38 34 72 31 31 13 30 11 06 03 55 04 04 0c 0a 6d 79 20
73 75 72 6e 61 6d 65 31 11 30 0f 06 03 55 04 0c 0c 08 6d 79 20 74 69
74 6c 65 31 15 30 13 06 03 55 04 2a 0c 0c 6d 79 20 67 69 76 65 6e 4e
61 6d 65 31 10 30 0e 06 03 55 04 29 0c 07 6d 79 20 6e 61 6d 65 30 7a
30 14 06 07 2a 86 48 ce 3d 02 01 06 09 2b 24 03 03 02 08 01 01 0b 03
62 00 04 67 09 c9 92 91 9b 49 c4 8f d9 31 d0 5c 49 7d 38 65 e6 08 4c
91 df 3a 4c 7e 78 1f 41 85 43 b0 23 d5 9e 8b f2 5d 13 3f b1 a0 94 e9
d4 2c 8f a6 ed 3b 46 e9 88 3a 35 ab d4 b0 a9 d3 0a ae fd 9b 7e 88 ed
38 00 56 5d 1e 7f 06 33 13 4d 65 19 29 2d 49 bd 55 ec 30 a1 67 19 7f
ec 0f 74 29 82 2b 95 a3 81 b0 30 81 ad 30 0b 06 03 55 1d 0f 04 04 03
02 07 80 30 48 06 08 2b 06 01 05 05 07 01 07 04 3c 30 3a 30 19 04 02
00 01 30 13 03 04 00 c0 00 02 03 05 04 c6 33 64 00 03 04 00 cb 00 71
30 1d 04 02 00 02 30 17 03 07 00 20 01 0d b8 12 34 30 0c 03 04 00 3f
ff 06 03 04 00 3f ff 0f 30 54 06 08 2b 06 01 05 05 07 01 1c 04 48 30
46 30 1a 04 03 00 01 01 30 13 03 04 00 c0 00 02 03 05 04 c6 33 64 00
03 04 00 cb 00 71 30 28 04 03 00 02 01 30 21 03 07 00 20 01 0d b8 12
34 30 16 03 05 00 3f ff 00 03 03 0d 00 3f ff 01 22 00 00 22 33 33 44
55 66 30 0a 06 08 2a 86 48 ce 3d 04 03 03 03 67 00 30 64 02 30 67 09
c9 92 91 9b 49 c4 8f d9 31 d0 5c 49 7d 38 65 e6 08 4c 91 df 3a 4c 7e
78 1f 41 85 43 b0 23 d5 9e 8b f2 5d 13 3f b1 a0 94 e9 d4 2c 8f a6 ed
02 30 20 ed 9f db 5a 30 9b 2c 87 04 dd a5 f1 44 f1 7b b3 16 b9 8c 29
11 24 fb a5 cf ec 6e f9 7f 26 88 06 9a e6 c5 2e 2b 3c e2 23 12 8d d1
0c 2a a7 30
```

A.5.1. Example: C509 Certificate Encoding

The CBOR encoding (~C509Certificate) of the X.509 certificate is shown below in CBOR diagnostic format.

```
/This defines a CBOR Sequence (RFC 8742):/

3,
h'1234',
1,
null,
1735776000,
1767312000,
[
  1,"selfsign-brainpoolp384r1",
  2, "my surname",
  10, "my title",
  13, "my givenName",
  25, "my name"
],
25,
h'046709C992919B49C48FD931D05C497D3865E6084C91DF3A4C7E781F418543B0
  23D59E8BF25D133FB1A094E9D42C8FA6ED3B46E9883A35ABD4B0A9D30AAEFD9B
  7E88ED3800565D1E7F0633134D6519292D49BD55EC30A167197FEC0F7429822B
  95',
[
  2, 1,
  32, [
       1, null, [ 29360130, 24770733054, -24770012047 ],
       2, null, [
                  316663873933876,
                  [ -316663852962606, 9 ]
                ]
      ],
  34, [
       1, 1, [ 29360130, 24770733054, -24770012047 ],
       2, 1, [
                h'0020010DB81234',
                [ h'003FFF0003', h'003FFF01220000223333445566' ]
              ]
      ]
],
h'6709C992919B49C48FD931D05C497D3865E6084C91DF3A4C7E781F418543B023
  D59E8BF25D133FB1A094E9D42C8FA6ED20ED9FDB5A309B2C8704DDA5F144F17B
  B316B98C291124FBA5CFEC6EF97F2688069AE6C52E2B3CE223128DD10C2AA730'
```

Acknowledgments

The authors want to thank Henk Birkholz, Mike Bishop, Mohamed Boucadair, Corey Bonnell, Carsten Bormann, Deb Cooley, Roman Danyliw, Viktor Dukhovni, Paul Hoffman, Russ Housley, Christopher Inacio, Olle Johansson, Benjamin Kaduk, Ted Lemon, Ilari Liusvaara, Laurence Lundblade, Francesca Palombini, Thomas Peterson, Michael Richardson, Stefan Santesson, Jim Schaad, Brian Sipos, Rene Struik, Ketan Talaulikar, Fraser Tweedale, Gunter Van de Velde, ric Vyncke, and Paul Wouters for reviewing and commenting on intermediate versions of the draft.

Authors' Addresses

John Preu Mattsson
Ericsson AB
Email: john.mattsson@ericsson.com

Gran Selander
Ericsson AB
Email: goran.selander@ericsson.com

Shahid Raza
University of Glasgow
Email: shahid.raza@glasgow.ac.uk

Joel Hglund
RISE AB
Email: joel.hoglund@ri.se

Martin Furuhed
IN Groupe
Email: martin.furuhed@ingroupe.com

Lijun Liao
NIO
Email: lijun.liao@nio.io