

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 5 September 2024

J. Preuß Mattsson  
G. Selander  
Ericsson AB  
S. Raza  
J. Höglund  
RISE AB  
M. Furuhed  
Nexus Group  
4 March 2024

CBOR Encoded X.509 Certificates (C509 Certificates)  
draft-ietf-cose-cbor-encoded-cert-09

## 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 all certificates compatible with the RFC 7925, IEEE 802.1AR (DevID), CNSA, RPKI, GSMA eUICC, and CA/Browser Forum Baseline Requirements profiles. When used to re-encode DER encoded X.509 certificates, the CBOR encoding can in many cases reduce the size of RFC 7925 profiled certificates with over 50% while also significantly reducing memory and code size compared to ASN.1. The CBOR encoded structure can alternatively be signed directly ("natively signed"), which does not require re-encoding for the signature to be verified. The document also specifies C509 Certificate Signing Requests, C509 COSE headers, a C509 TLS certificate type, and a C509 file format.

## 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 is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 September 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction . . . . .	3
2. Notational Conventions . . . . .	5
3. C509 Certificate . . . . .	5
3.1. Message Fields . . . . .	6
3.2. Encoding of subjectPublicKey and issuerSignatureValue . .	10
3.3. Encoding of Extensions . . . . .	11
3.4. COSE Header Parameters . . . . .	16
3.5. Private Key Structures . . . . .	17
4. C509 Certificate Signing Request . . . . .	18
5. C509 Processing and Certificate Issuance . . . . .	20
6. Legacy Considerations . . . . .	21
7. Expected Certificate Sizes . . . . .	21
8. Security Considerations . . . . .	22
9. IANA Considerations . . . . .	23
9.1. C509 Certificate Types Registry . . . . .	23

9.2.	C509 Certificate Request Types Registry . . . . .	24
9.3.	C509 Attributes Registry . . . . .	24
9.4.	C509 Extensions Registry . . . . .	29
9.5.	C509 Certificate Policies Registry . . . . .	33
9.6.	C509 Policies Qualifiers Registry . . . . .	36
9.7.	C509 Information Access Registry . . . . .	36
9.8.	C509 Extended Key Usages Registry . . . . .	38
9.9.	C509 General Names Registry . . . . .	40
9.10.	C509 Signature Algorithms Registry . . . . .	41
9.11.	C509 Public Key Algorithms Registry . . . . .	46
9.12.	COSE Header Parameters Registry . . . . .	49
9.13.	Media Type application/cose-c509-cert . . . . .	49
9.14.	Media Type application/cose-c509-pkcs10 . . . . .	50
9.15.	Media Type application/cose-c509-privkey . . . . .	51
9.16.	Media Type application/cose-c509-pem . . . . .	52
9.17.	CoAP Content-Formats Registry . . . . .	53
9.18.	TLS Certificate Types Registry . . . . .	54
9.19.	CBOR Tags Registry . . . . .	54
10.	References . . . . .	54
10.1.	Normative References . . . . .	54
10.2.	Informative References . . . . .	56
Appendix A.	Example C509 Certificates . . . . .	59
A.1.	Example RFC 7925 profiled X.509 Certificate . . . . .	59
A.2.	Example IEEE 802.1AR profiled X.509 Certificate . . . . .	63
A.3.	Example CAB Baseline ECDSA HTTPS X.509 Certificate . . . . .	67
A.4.	Example CAB Baseline RSA HTTPS X.509 Certificate . . . . .	69
Acknowledgments	. . . . .	72
Authors' Addresses	. . . . .	72

## 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 40 - 50 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 RPKI [RFC6487] where a single certificate can be very large. More compact certificate representations are therefore desirable in many use cases. Due to the current PKI usage of DER encoded X.509 certificates, keeping compatibility with DER encoded X.509 is necessary at least for a transition period. However, the use of a more compact encoding with the Concise Binary Object Representation (CBOR) [RFC8949] reduces the certificate size significantly which has known performance benefits in terms of

decreased communication overhead, power consumption, latency, storage, etc. The use of CBOR also reduces code complexity, code size, memory usage, and CPU usage.

CBOR is a data format designed for small code size and small message size. CBOR builds on the JSON data model but extends it by e.g. encoding binary data directly without base64 conversion. In addition to the binary CBOR encoding, CBOR also has a diagnostic notation that is readable and editable by humans. The Concise Data Definition Language (CDDL) [RFC8610] provides a way to express structures for protocol messages and APIs that use CBOR. RFC 8610 also extends the diagnostic notation.

CBOR data items are encoded to or decoded from byte strings using a type-length-value encoding scheme, where the three highest order bits of the initial byte contain information about the major type. CBOR supports several different types of data items, in addition to integers (int, uint), simple values (e.g. null), byte strings (bstr), and text strings (tstr), CBOR also supports arrays [] of data items, maps {} of pairs of data items, and sequences of data items. For a complete specification and examples, see [RFC8949], [RFC8610], and [RFC8742]. We recommend implementors to get used to CBOR by using the CBOR playground [CborMe].

CAB Baseline Requirements [CAB-TLS], RFC 7925 [RFC7925], IEEE 802.1AR [IEEE-802.1AR], and CNSA [RFC8603] specify certificate profiles which can be applied to certificate based authentication with, e.g., TLS [RFC8446], QUIC [RFC9000], DTLS [RFC9147], COSE [RFC9052], EDHOC [I-D.ietf-lake-edhoc], or Compact TLS 1.3 [I-D.ietf-tls-ctls]. RFC 7925 [RFC7925], RFC7925bis [I-D.ietf-uta-tls13-iot-profile], and IEEE 802.1AR [IEEE-802.1AR] specifically target Internet of Things deployments. This document specifies a CBOR encoding based on [X.509-IoT], which can support large parts of RFC 5280. The encoding supports all RFC 7925, IEEE 802.1AR, CAB Baseline [CAB-TLS], [CAB-Code], RPKI [RFC6487], eUICC [GSMA-eUICC] profiled X.509 certificates, and is designed to render a compact encoding of certificates used in constrained environments.

The resulting certificates are called C509 Certificates. This document does not specify a certificate profile. Two variants are defined 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.

2. Natively signed C509 certificates, where the signature is calculated over the CBOR encoding instead of over the DER encoding as in 1. This removes the need for ASN.1 and DER parsing and the associated complexity but they are not backwards compatible with implementations requiring DER encoded X.509.

Natively signed C509 certificates can be applied in devices that are only required to authenticate to natively signed C509 certificate compatible servers, which is not a major restriction for many IoT deployments where the parties issuing and verifying certificates can be a restricted ecosystem.

This document also specifies C509 Certificate Signing Requests, see Section 4; COSE headers for use of the C509 certificates with COSE, see Section 9.12; and a TLS certificate type for use of the C509 certificates with TLS and QUIC (with or without additional TLS certificate compression), see Section 9.18.

## 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 for C509 certificates, with the overall objective to produce a very compact representation supporting large parts of [RFC5280], and everything in [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, OID are replaced with short integers or complemented with CBOR OID and PEN encodings [RFC9090], and redundant encoding is removed. Combining these different components reduces the certificate size significantly, which is not possible with general purpose compression algorithms, see Figure 4.

The 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 is adhering to the restrictions given by [RFC5280]. The re-encoding is known to work with DER encoded certificates but might work with other canonical encodings. The re-encoding does not work for BER encoded certificates.

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

### 3.1. Message Fields

The X.509 fields and their CBOR encodings are listed below, and used in the definition of C509 certificates, see Figure 1.

C509 certificates are defined in terms of DER encoded [RFC5280] X.509 certificates:

- \* version. The 'version' field is encoded in the 'c509CertificateType' CBOR int. The field 'c509CertificateType' also indicates the type of the C509 certificate. Currently, the type can be a natively signed C509 certificate following X.509 v3 (c509CertificateType = 0) or a CBOR re-encoded X.509 v3 DER certificate (c509CertificateType = 1), see Section 9.1.
- \* serialNumber. The 'serialNumber' 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.
- \* signature. The 'signature' field is always the same as the 'signatureAlgorithm' field and therefore omitted from the CBOR encoding.
- \* issuer. In the general case, the sequence of 'RelativeDistinguishedName' is encoded as a CBOR array of CBOR arrays of Attributes. Typically, each RelativeDistinguishedName only contains a single attribute and the sequence is then encoded as a CBOR array of Attributes. Each Attribute is encoded as either
  - a (CBOR int, CBOR text string) pair, or
  - a (unwrapped CBOR OID, CBOR bytes) pair, or
  - a (CBOR PEN, CBOR bytes) pair.

The absolute value of the CBOR int (see Figure 7) encodes the attribute type and the sign is used to represent the character string type; positive for Utf8String, negative for PrintableString. The Attribute Email Address is always an IA5String. In natively signed C509 certificates all text strings are UTF-8 encoded and all attributeType SHALL be non-negative. Text strings SHALL still adhere to any X.509 restrictions, i.e., serialNumber SHALL only contain the 74 character subset of ASCII allowed by PrintableString and countryName SHALL have length 2. The string types teletexString, universalString, and bmpString are not supported. If Name contains a single Attribute containing an utf8String encoded 'common name' it is encoded as follows:

- If the text string has an even length 2 and contains only the symbols '0''9' or 'a''f', it is encoded as a CBOR byte string, prefixed with an initial byte set to '00'.
  - If the text string contains an EUI-64 of the form "HH-HH-HH-HH-HH-HH-HH-HH" where 'H' is one of the symbols '0''9' or 'A''F' it is encoded as a CBOR byte string prefixed with an initial byte set to '01', for a total length of 9. An EUI-64 mapped from a 48-bit MAC address (i.e., of the form "HH-HH-HH-HH-HH-HH-HH-HH) is encoded as a CBOR byte string prefixed with an initial byte set to '01', for a total length of 7.
  - Otherwise it is encoded as a CBOR text string.
- \* 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, 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 CBOR null.
- \* subject. The 'subject' is encoded exactly like issuer.

- \* `subjectPublicKeyInfo`. The `'AlgorithmIdentifier'` field including parameters is encoded as the CBOR int `'subjectPublicKeyAlgorithm'` (see Section 9.11) 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. 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.
- \* `issuerUniqueID`. Not supported.
- \* `subjectUniqueID`. Not supported.
- \* `extensions`. The `'extensions'` field is encoded as a CBOR array where each extension is encoded as either
  - a CBOR int (see Section 9.4) followed by an optional CBOR item of any type, or
  - an unwrapped CBOR OID tag [RFC9090] followed by an optional CBOR bool encoding `'critical'` and the DER encoded value of the `'extnValue'` encoded as a CBOR byte string, or
  - a CBOR PEN tag [RFC9090] followed by an optional CBOR bool encoding `'critical'` and the DER encoded value of the `'extnValue'` encoded as a CBOR byte string.

If the array contains exactly two ints and the absolute value of the first int is 2 (corresponding to `keyUsage`), the array is omitted and the extensions is encoded as a single CBOR int with the absolute value of the second int and the sign of the first int. Extensions are encoded as specified in Section 3.3. The extensions mandated to be supported by [RFC7925] and [IEEE-802.1AR] are given special treatment. An omitted `'extensions'` field is encoded as an empty CBOR array.

- \* `signatureAlgorithm`. The `'signatureAlgorithm'` field including parameters is encoded as a CBOR int (see Section 9.10) or as an array with an unwrapped CBOR OID tag [RFC9090] optionally followed by the parameters encoded as a CBOR byte string.
- \* `signatureValue`. In general, the `'signatureValue'` BIT STRING value field is encoded as the CBOR byte string `issuerSignatureValue`. This specification assumes 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

The following Concise Data Definition Language (CDDL) defines the CBOR array C509Certificate and the CBOR sequence [RFC8742] TBSCertificate. The member names therefore only have documentary value. Applications not requiring a CBOR item MAY represent C509 certificates with the CBOR sequence ~C509Certificate (unwrapped C509Certificate).

```

C509Certificate = [
    TBSCertificate,
    issuerSignatureValue : any,
]

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

CertificateSerialNumber = ~biguint

Name = [ * RelativeDistinguishedName ] / text / bytes

RelativeDistinguishedName = Attribute / [ 2* Attribute ]

Attribute = ( attributeType: int, attributeValue: text ) //
            ( attributeType: ~oid, attributeValue: bytes ) //
            ( attributeType: pen, attributeValue: bytes )

Time = ~time / null

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

Extensions = [ * Extension ] / int

Extension = ( extensionID: int, extensionValue: any ) //
            ( extensionID: ~oid, ? critical: true,
              extensionValue: bytes ) //
            ( extensionID: pen, ? critical: true,
              extensionValue: bytes )

```

Figure 1: CDDL for C509Certificate.

### 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 is omitted and `subjectPublicKey` consist 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, the octets 0xfe and 0xfd are used instead of 0x02 and 0x03 in the CBOR encoding to represent even and odd y-coordinate, respectively.

### 3.2.2. Encoding of issuerSignatureValue

For ECDSA signatures, the `SEQUENCE` and `INTEGER` type and length fields as well as the any leading 0x00 byte (to indicate that the number is not negative) are omitted. If the two `INTEGER` value fields have different lengths, the shorter `INTEGER` value field is padded with zeroes so that the two fields have the same length. The resulting byte string is encoded as a CBOR byte string.

### 3.3. Encoding of Extensions

This section details the encoding of the `'extensions'` field. The `'extensions'` field is encoded as a CBOR array where each `extensionID` is encoded either as a CBOR int or as an unwrapped CBOR OID tag. If `'extensionID'` is encoded an int (see Section 9.4), the sign is used to encode if the extension is critical and the `'critical'` field is omitted. Critical extensions are encoded with a negative sign and non-critical extensions are encoded with a positive sign.

The `'extnValue'` OCTET STRING value field is encoded as the CBOR byte string `'extensionValue'` except for the extensions specified below. For some extensions, only commonly used parts are supported by the CBOR encoding. If unsupported parts are used, the CBOR encoding cannot be used.

A note on extension ID naming: in existing OID databases most IDs 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). 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 for special encoding in case keyUsage is the only extension present.

```
KeyUsage = int
```

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

```
PolicyMappings = [
  + (issuerDomainPolicy: ~oid, subjectDomainPolicy: ~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 9.8), unwrapped CBOR OID tags [RFC9090], or CBOR PEN tags [RFC9090], where each int or OID / PEN tag encodes a key usage purpose. If the array contains a single KeyPurposeId, the array is omitted.

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

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

InhibitAnyPolicy = 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 9.9 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 9.9). 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 hardwareModuleName type of otherName has been given its own int due to its mandatory use in IEEE 802.1AR. When 'otherName + hardwareModuleName' is used, then [ ~oid, bytes ] is used to contain the pair ( hwType, hwSerialNum ) directly as specified in [RFC4108]. Only the general names in Section 9.9 are supported.

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 the CRL Distribution Points is a sequence of DistributionPointName, where each DistributionPointName only contains uniformResourceIdentifiers, the extension value can be CBOR encoded. extensionValue is encoded as follows:

DistributionPointName = [ 2\* text ] / text  
CRLDistributionPoints = [ + DistributionPointName ]

- \* 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 9.7) 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, certIssuer, and certSerialNumber 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 / null,
    authorityCertIssuer: GeneralNames,
    authorityCertSerialNumber: CertificateSerialNumber
]
AuthorityKeyIdentifier = KeyIdentifierArray / KeyIdentifier
```

- \* Certificate Policies (certificatePolicies). If noticeRef is not used and any explicitText are encoded as UTF8String, the extension value can be CBOR encoded. OIDs registered in Section 9.5 are encoded as an int. The policyQualifierId is encoded as an CBOR int (see Section 9.6) 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 9.9 the extension value can be CBOR encoded. C509 uses the same additions and restrictions as defined in Section 4.2.1.10 of [RFC5280]. Note that the minimum and maximum fields are not used and therefore omitted.

```
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.

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

- \* AS Resources (id-pe-autonomousSysIds). If rdi is not present, the extension value can be CBOR encoded. Each ASId is encoded as an uint. With the exception of the first ASId, the ASId is encoded as the difference to the previous ASId.

```
AsIdsOrRanges = uint / [uint, uint]
ASIdentifiers = [ + AsIdsOrRanges ] / null
```

- \* AS Resources v2 (id-pe-autonomousSysIds-v2). Encoded exactly like autonomousSysIds.

- \* IP Resources (id-pe-ipAddrBlocks). If rdi and SAFI is not present, the extension value can be CBOR encoded. Each AddressPrefix is encoded as a CBOR bytes string (without the unused bits octet) followed by the number of unused bits encoded as a CBOR uint. Each AddressRange is encoded as an array of two CBOR byte strings. The unused bits for min and max are omitted, but the unused bits in max IPAddress is set to ones. With the exception of the first Address, if the byte string has the same length as the previous Address, the Address is encoded as an uint with the the difference to the previous Address. It should be noted that using address differences for compactness prevents encoding an address range larger than  $2^{64} - 1$  corresponding to the cbor integer max value.

```
Address = bytes / uint,
AddressPrefix = (Address, unusedBits: uint)
AddressRange = [Address, Address]
IPAddressOrRange = AddressPrefix / AddressRange
IPAddressChoice = [ + IPAddressOrRange ] / null
IPAddrBlocks = [ AFI: uint, IPAddressChoice ]
```

- \* IP Resources v2 (id-pe-ipAddrBlocks-v2). Encoded exactly like id-pe-ipAddrBlocks.

- \* Signed Certificate Timestamp. If all the SCTs are version 1, and there are no SCT extensions, the extension value can be CBOR encoded. LogIDs are encoded as CBOR byte strings, the timestamp is encoded as a CBOR int (milliseconds since validityNotBefore), and the signature is encoded with an (AlgorithmIdentifier, any) pair in the same way as issuerSignatureAlgorithm and issuerSignatureValue.

```
SignedCertificateTimestamp = (  
  logID: bytes,  
  timestamp: int,  
  sigAlg: AlgorithmIdentifier,  
  sigValue: any,  
)  
SignedCertificateTimestamps = [ + SignedCertificateTimestamp ]
```

### 3.3.1. Example Encoding of Extensions

The examples below use values from Section 9.4, Section 9.8, and Section 9.9:

- \* 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, 6 ].
- \* 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, 6 ], 3, "example.com" ].

### 3.4. COSE Header Parameters

The formatting and processing for c5b, c5c, and c5t, and c5u, defined in Table 1 are similar to x5bag, x5chain, x5t, x5u defined in [RFC9360] except that the certificates are C509 instead of DER encoded X.509 and uses a COSE\_C509 structure instead of COSE\_X509. c5u provides an alternative way to identify an untrusted certificate bag/chain by reference with a URI. The content is a COSE\_C509 item



served with the application/cose-c509-cert media type, see Section 9.13, with corresponding CoAP Content-Format defined in Section 9.17. A stored file format is defined in [RFC9277], with "magic number" TBD8 composed of the reserved CBOR tag 55799 concatenated with the CBOR tag calculated from the CoAP Content-Format value.

The COSE\_C509 structure used in c5b, c5c, and c5u is defined as:

COSE\_C509 = C509Certificate / [ 2\* C509Certificate ]

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 some out-of-band confirmation.

Name	Label	Value Type	Description
c5b	TBD1	COSE_C509	An unordered bag of C509 certificates
c5c	TBD2	COSE_C509	An ordered chain of C509 certificates
c5t	TBD3	COSE_CertHash	Hash of a C509Certificate
c5u	TBD4	uri	URI pointing to a COSE_C509 containing a ordered chain of certificates

Table 1: COSE Header Parameters

Note that certificates can also be identified with a 'kid' header parameter by storing 'kid' and the associated bag or chain in a dictionary.

### 3.5. 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 = [  
  subjectPrivateKeyAlgorithm: AlgorithmIdentifier,  
  subjectPrivateKey: any,  
]
```

The C509PrivateKey item is served with the application/cose-c509-privkey media type, see Section 9.15, with corresponding CoAP Content-Format defined in Section 9.17. A stored file format is defined in [RFC9277], with "magic number" TBD12 composed of the reserved CBOR tag 55799 concatenated with the CBOR tag calculated from the CoAP Content-Format value.

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

The C509PEM item is served with the application/cose-c509-pem media type, see Section 9.16, with corresponding CoAP Content-Format defined in Section 9.17. A stored file format is defined in [RFC9277], with "magic number" TBD13 composed of the reserved CBOR tag 55799 concatenated with the CBOR tag calculated from the CoAP Content-Format value.

Editor's note: Include further details for encoding of subjectPrivateKey.

#### 4. C509 Certificate Signing Request

This section defines the format of a C509 Certificate Signing Request (CSR), also known as a C509 Certificate Request, based on and compatible with RFC 2986 [RFC2986], and reusing the formatting of C509 certificates defined in Section 3. The media type is application/cose-c509-pkcs10, see Section 9.14, with corresponding CoAP Content-Format defined in Section 9.17. The "magic number" TBD9 is composed of the reserved CBOR tag 55799 concatenated with the CBOR tag calculated from the CoAP Content-Format value, see [RFC9277].

Different types of C509 Certificate Requests are defined, see Section 9.2, all using the same CBOR encoding and differing only in what is being signed and what type of C509 certificate is being requested:

- \* The C509 Certificate Request can either be an invertible CBOR re-encoding of a DER encoded RFC 2986 certification request, or it can be natively signed where the signature is calculated over the CBOR encoding instead of the DER encoding.

- \* The requested C509 certificate in the C509 Certificate Request can either be of type 0 or of type 1, see Section 9.1.

Combining these options enables the four instances of `c509CertificateRequestType` defined in Section 9.2. An implementation MAY only support `c509CertificateRequestType = 0`. The most common variants are expected to be:

- \* `c509CertificateRequestType = 0`. This type indicates that the C509 Certificate Request is natively signed, and that the requested certificate format is C509 Type 0. This encoding removes the need for ASN.1 and DER parsing and re-encoding in the requesting party.
- \* `c509CertificateRequestType = 3`. This type indicates that the C509 Certificate Request is CBOR re-encoded RFC 2986 certification requests, and that the requested certificate format is C509 Type 1. This encoding is backwards compatible with legacy RFC 2986 certification requests and X.509 certificates, but enables a reduced transport overhead.

`subjectSignatureAlgorithm` can be a signature algorithm or a non-signature proof-of-possession algorithm, e.g., as defined in [RFC6955]. In the latter case, the signature is replaced by a MAC and requires a public Diffie-Hellman key of the verifier distributed out-of-band. Both kinds are listed in the C509 Signature Algorithms Registry, see Section 9.10. Note that a key agreement key pair may be used with a signature algorithm in a certificate request, see Appendix A.1.3.

Certificate request attributes, i.e. attributes for use with certificate requests providing additional information about the subject of the certificate, are defined in Section 5.4 of [RFC2985]. The attribute `extensionRequest` is supported with a dedicated element. Other certificate request attributes are included using the same Extensions structure as in `extensionsRequest`, both extensions and attributes are listed in the C509 Extensions Registry, see Figure 8. The only other certificate request attribute specified in this document is `challengePassword` which is defined for `utf8String` values and encoded as CBOR text string, except if the text string contains only the symbols `'0''9'` or `'a''f'`, in which case it is encoded as a CBOR byte string.

```
C509CertificateRequest = [  
    TBSCertificateRequest,  
    subjectSignatureValue: any,  
]  
  
; The elements of the following group are used in a CBOR Sequence:  
TBSCertificateRequest = (  
    c509CertificateRequestType: int,  
    subject: Name,  
    subjectPublicKeyAlgorithm: AlgorithmIdentifier,  
    subjectPublicKey: any,  
    extensionsRequest: Extensions,  
    subjectSignatureAlgorithm: AlgorithmIdentifier,  
)  
  
challengePassword: tstr / bstr
```

Figure 2: CDDL for C509CertificateRequest.

After verifying the `subjectSignatureValue`, the CA MAY transform the `C509CertificateRequest` into a [RFC2986] `CertificationRequestInfo` for compatibility with existing procedures and code.

## 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 Certificate Signing Request (CSR) 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].

When a certificate 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 = 1` 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 = 0` 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. 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 CSR, the CA transforms it into a DER encoded RFC 2986 CertificationRequestInfo and use 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 the handshake is 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 a constrained wireless links. The conversion to X.509 within an 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.

## 7. Expected Certificate Sizes

The CBOR encoding of the sample certificate chains given in Appendix A results in the numbers shown in Figure 3 and Figure 4. COSE\_X509 is defined in [RFC9360] and COSE\_C509 is defined in Section 9.12. After RFC 7925 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 will be small, mainly corresponding to making the ASN.1 encoding more compact. CBOR encoding can however significantly compress RFC 7925 profiled certificates. For the example HTTPS certificate chains ([www.ietf.org](http://www.ietf.org) and [tools.ietf.org](http://tools.ietf.org)) both C509 and Brotli perform well complementing

each other. C509 use dedicated information to compress individual certificates, while Brotli can compress duplicate information in the entire chain. Note that C509 certificates of type 0 and 1 have the same size. For Brotli [RFC7932], the Rust crate Brotli 3.3.0 was used with compression level 11 and window size 22.

	COSE_X509	COSE_C509
RFC 7925 profiled IoT Certificate (1)	317	139
ECDSA HTTPS Certificate Chain (2)	2193	1394
RSA HTTPS Certificate Chain (4)	5175	3934

Figure 3: Comparing Sizes of Certificate Chains in COSE. Number of bytes (length of certificate chain).

	X.509	X.509 + Brotli	C509	C509 + Brotli
RFC 7925 Cert (1)	327	324	151	167
RPKI Cert (1)	20991	9134	8660	5668
HTTPS Chain (2)	2204	1455	1414	1063
HTTPS Chain (4)	5190	3244	3958	2845
HTTPS Bag (8)	11578	3979	8882	3519

Figure 4: Comparing Sizes of Certificate Chains with TLS. Number of bytes (length of certificate chain). X.509 and C509 are Certificate messages. X.509 + Brotli and C509 + Brotli are CompressedCertificate messages.

## 8. Security Considerations

The CBOR profiling of X.509 certificates does not change the security assumptions needed when deploying standard X.509 certificates but decreases the number of fields transmitted, which reduces the risk for implementation errors.

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 draft does not reveal any additional information compared to X.509. Because of difference in size, it will be possible to detect that this profile is used. The gateway solution described in Section 6 requires unencrypted certificates and is not recommended.

## 9. IANA Considerations

This document creates several new registries under the new heading "CBOR Encoded X.509 (C509) Parameters". For all items, the 'Reference' field points to this document.

The expert reviewers for the registries defined in this document are expected to ensure that the usage solves a valid use case that could not be solved better in a different way, that it is not going to duplicate one that is already registered, and that the registered point is likely to be used in deployments. They are furthermore expected to check the clarity of purpose and use of the requested code points. Experts should take into account the expected usage of entries when approving point assignment, and 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 encodings, other values in the interval [-256, 255] have a 2 byte encodings, and the remaining values in the interval [-65536, 65535] have 3 byte encodings.

### 9.1. C509 Certificate Types Registry

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

Value	Description
0	Natively Signed C509 Certificate following X.509 v3
1	CBOR re-encoding of X.509 v3 Certificate

Figure 5: C509 Certificate Types

## 9.2. C509 Certificate Request Types Registry

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

Value	Description
0	Natively Signed C509 Certificate Request. Requested certificate is C509 Type 0.
1	Natively Signed C509 Certificate Request. Requested certificate is C509 Type 1.
2	CBOR re-encoding of RFC 2986 certification request. Requested certificate is C509 Type 0.
3	CBOR re-encoding of RFC 2986 certification request. Requested certificate is C509 Type 1.

Figure 6: C509 Certificate Request Types

## 9.3. C509 Attributes Registry

IANA has created a new registry titled "C509 Attributes" under the new heading "CBOR Encoded X.509 (C509) Parameters". The columns 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. For values in the interval  $[0, 23]$  the registration procedure is "IETF Review" and "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	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:
1	Name: Common Name Identifiers: commonName, cn OID: 2.5.4.3 DER: 06 03 55 04 03 Comments:
2	Name: Surname Identifiers: surname, sn OID: 2.5.4.4 DER: 06 03 55 04 04 Comments:
3	Name: Serial Number Identifiers: serialNumber OID: 2.5.4.5 DER: 06 03 55 04 05 Comments:
4	Name: Country Identifiers: countryName, c OID: 2.5.4.6 DER: 06 03 55 04 06 Comments:
5	Name: Locality Identifiers: localityName, locality, l OID: 2.5.4.7 DER: 06 03 55 04 07 Comments:
6	Name: State or Province Identifiers: stateOrProvinceName, st OID: 2.5.4.8 DER: 06 03 55 04 08 Comments:
7	Name: Street Address Identifiers: streetAddress, street OID: 2.5.4.9

	DER:	06 03 55 04 09
	Comments:	
8	Name:	Organization
	Identifiers:	organizationName, o
	OID:	2.5.4.10
	DER:	06 03 55 04 0A
	Comments:	
9	Name:	Organizational Unit
	Identifiers:	organizationalUnitName, ou
	OID:	2.5.4.11
	DER:	06 03 55 04 0B
	Comments:	
10	Name:	Title
	Identifiers:	title
	OID:	2.5.4.12
	DER:	06 03 55 04 0C
	Comments:	
11	Name:	Business Category
	Identifiers:	businessCategory
	OID:	2.5.4.15
	DER:	06 03 55 04 0F
	Comments:	
12	Name:	Postal Code
	Identifiers:	postalCode
	OID:	2.5.4.17
	DER:	06 03 55 04 11
	Comments:	
13	Name:	Given Name
	Identifiers:	givenName
	OID:	2.5.4.42
	DER:	06 03 55 04 2A
	Comments:	
14	Name:	Initials
	Identifiers:	initials
	OID:	2.5.4.43
	DER:	06 03 55 04 2B
	Comments:	
15	Name:	Generation Qualifier
	Identifiers:	generationQualifier
	OID:	2.5.4.44

	DER:	06 03 55 04 2C
	Comments:	
16	Name:	DN Qualifier
	Identifiers:	dnQualifier
	OID:	2.5.4.46
	DER:	06 03 55 04 2E
	Comments:	
17	Name:	Pseudonym
	Identifiers:	pseudonym
	OID:	2.5.4.65
	DER:	06 03 55 04 41
	Comments:	
18	Name:	Organization Identifier
	Identifiers:	organizationIdentifier
	OID:	2.5.4.97
	DER:	06 03 55 04 61
	Comments:	
19	Name:	Inc. Locality
	Identifiers:	jurisdictionOfIncorporationLocalityName
	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:	
20	Name:	Inc. State or Province
	Identifiers:	jurisdictionOfIncorporation StateOrProvinceName
	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:	
21	Name:	Inc. Country
	Identifiers:	jurisdictionOfIncorporationCountryName
	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:	
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:	
24	Name:	Postal Address
	Identifiers:	postalAddress

	OID: 2.5.4.16 DER: 06 03 55 04 10 Comments:
25	Name: Name Identifiers: name OID: 2.5.4.41 DER: 06 03 55 04 29 Comments:
26	Name: Telephone Number Identifiers: telephoneNumber OID: 2.5.4.20 DER: 06 03 55 04 14 Comments:
27	Name: Directory Management Domain Name Identifiers: dmdName OID: 2.5.4.54 DER: 06 03 55 04 36 Comments:
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:
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:
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 00 Comments:

Figure 7: C509 Attributes

#### 9.4. C509 Extensions Registry

IANA has created a new registry titled "C509 Extensions Registry" under the new heading "CBOR Encoded X.509 (C509) Parameters". The columns of the registry are Value, Name, Identifiers, OID, DER, Comments, extensionValue, and Reference, where Value is an positive integer, and the other columns are text strings. The registry also contains CSR attributes for use in Certificate Requests, see Section 4. For values in the interval [1, 23] the registration procedure is "IETF Review" and "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	Extension
1	Name: Subject Key Identifier Identifiers: subjectKeyIdentifier OID: 2.5.29.14 DER: 06 03 55 1D 0E Comments: extensionValue: SubjectKeyIdentifier
2	Name: Key Usage Identifiers: keyUsage OID: 2.5.29.15 DER: 06 03 55 1D 0F Comments: AttributeValue: KeyUsage
3	Name: Subject Alternative Name Identifiers: subjectAltName OID: 2.5.29.17 DER: 06 03 55 1D 11 Comments: extensionValue: SubjectAltName
4	Name: Basic Constraints Identifiers: basicConstraints OID: 2.5.29.19 DER: 06 03 55 1D 13 Comments: extensionValue: BasicConstraints
5	Name: CRL Distribution Points Identifiers: cRLDistributionPoints OID: 2.5.29.31

	DER: 06 03 55 1D 1F Comments: extensionValue: CRLDistributionPoints
6	Name: Certificate Policies Identifiers: certificatePolicies OID: 2.5.29.32 DER: 06 03 55 1D 20 Comments: extensionValue: CertificatePolicies
7	Name: Authority Key Identifier Identifiers: authorityKeyIdentifier OID: 2.5.29.35 DER: 06 03 55 1D 23 Comments: extensionValue: AuthorityKeyIdentifier
8	Name: Extended Key Usage Identifiers: extKeyUsage OID: 2.5.29.37 DER: 06 03 55 1D 25 Comments: 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: extensionValue: AuthorityInfoAccessSyntax
10	Name: Signed Certificate Timestamp List Identifiers: OID: 1.3.6.1.4.1.11129.2.4.2 DER: 06 0A 2B 06 01 04 01 D6 79 02 04 02 Comments: extensionValue: SignedCertificateTimestampList
24	Name: Subject Directory Attributes Identifiers: subjectDirectoryAttributes OID: 2.5.29.9 DER: 06 03 55 1D 09 Comments: extensionValue: SubjectDirectoryAttributes
25	Name: Issuer Alternative Name Identifiers: issuerAltName

	OID: 2.5.29.18 DER: 06 03 55 1D 12 Comments: extensionValue: IssuerAltName
26	Name: Name Constraints Identifiers: nameConstraints OID: 2.5.29.30 DER: 06 03 55 1D 1E Comments: extensionValue: NameConstraints
27	Name: Policy Mappings Identifiers: policyMappings OID: 2.5.29.33 DER: 06 03 55 1D 21 Comments: extensionValue: PolicyMappings
28	Name: Policy Constraints Identifiers: policyConstraints OID: 2.5.29.36 DER: 06 03 55 1D 24 Comments: extensionValue: PolicyConstraints
29	Name: Freshest CRL Identifiers: freshestCRL OID: 2.5.29.46 DER: 06 03 55 1D 2E Comments: extensionValue: FreshestCRL
30	Name: Inhibit anyPolicy Identifiers: inhibitAnyPolicy OID: 2.5.29.54 DER: 06 03 55 1D 36 Comments: 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: extensionValue: SubjectInfoAccessSyntax
32	Name: IP Resources

	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: extensionValue: IPAddrBlocks
33	Name: AS Resources 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: extensionValue: ASIdentifiers
34	Name: IP Resources 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: extensionValue: IPAddrBlocks
35	Name: AS Resources 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: extensionValue: ASIdentifiers
36	Name: Biometric Information Identifiers: id-pe-biometricInfo OID: 1.3.6.1.5.5.7.1.2 DER: 06 08 2B 06 01 05 05 07 01 02 Comments: extensionValue:
37	Name: Precertificate Signing Certificate Identifiers: OID: 1.3.6.1.4.1.11129.2.4.4 DER: 06 0A 2B 06 01 04 01 D6 79 02 04 04 Comments: extensionValue:
38	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: extensionValue:



39	Name: Qualified Certificate Statements Identifiers: id-pe-qcStatements OID: 1.3.6.1.5.5.7.1.3 DER: 06 08 2B 06 01 05 05 07 01 03 Comments: extensionValue:
40	Name: S/MIME Capabilities Identifiers: smimeCapabilities OID: 1.2.840.113549.1.9.15 DER: 06 09 2A 86 48 86 F7 0D 01 09 0F Comments: extensionValue:
41	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: extensionValue:
255	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: CSR Attribute extensionValue: ChallengePassword

Figure 8: C509 Extensions and CSR Attributes

### 9.5. C509 Certificate Policies Registry

IANA has created a new registry titled "C509 Certificate Policies Registry" under the new heading "CBOR Encoded X.509 (C509) Parameters". The columns of the registry are Value, Name, Identifiers, OID, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. For values in the interval  $[-24, 23]$  the registration procedure is "IETF Review" and "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	Certificate Policy
0	Name: Any Policy Identifiers: anyPolicy OID: 2.5.29.32.0 DER: 06 04 55 1D 20 00 Comments:
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:
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:
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:
4	Name: Extended Validation (EV) Identifiers: ev-guidelines OID: 2.23.140.1.1 DER: 06 05 67 81 0C 01 01 Comments:
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:
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:
10	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:
11	Name: Remote SIM Provisioning Role eUICC Identifiers: id-rspRole-euicc OID: 2.23.146.1.2.1.1 DER: 06 07 67 81 12 01 02 01 01 Comments:
12	Name: Remote SIM Provisioning Role eUICC Manufacturer Identifiers: id-rspRole-eum OID: 2.23.146.1.2.1.2 DER: 06 07 67 81 12 01 02 01 02 Comments:
13	Name: Remote SIM Provisioning Role SM-DP+ TLS Identifiers: id-rspRole-dp-tls OID: 2.23.146.1.2.1.3 DER: 06 07 67 81 12 01 02 01 03 Comments:
14	Name: Remote SIM Provisioning Role SM-DP+ Authentication Identifiers: id-rspRole-dp-auth OID: 2.23.146.1.2.1.4 DER: 06 07 67 81 12 01 02 01 04 Comments:
15	Name: Remote SIM Provisioning Role SM-DP+ Profile Binding Identifiers: id-rspRole-dp-pb OID: 2.23.146.1.2.1.5 DER: 06 07 67 81 12 01 02 01 05 Comments:
16	Name: Remote SIM Provisioning Role SM-DS TLS Identifiers: id-rspRole-ds-tls OID: 2.23.146.1.2.1.6 DER: 06 07 67 81 12 01 02 01 06 Comments:
17	Name: Remote SIM Provisioning Role SM-DS Authentication

Identifiers:	id-rspRole-ds-auth
OID:	2.23.146.1.2.1.7
DER:	06 07 67 81 12 01 02 01 07
Comments:	

Figure 9: C509 Certificate Policies

#### 9.6. C509 Policies Qualifiers Registry

IANA has created a new registry titled "C509 Policies Qualifiers Registry" under the new heading "CBOR Encoded X.509 (C509) Parameters". The columns of the registry are Value, Name, Identifiers, OID, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. For values in the interval [-24, 23] the registration procedure is "IETF Review" and "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	Certificate Policy
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:
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:

Figure 10: C509 Policies Qualifiers

#### 9.7. C509 Information Access Registry

IANA has created a new registry titled "C509 Information Access Registry" under the new heading "CBOR Encoded X.509 (C509) Parameters". The columns of the registry are Value, Name, Identifiers, OID, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. For values in the interval [-24, 23] the registration procedure is "IETF Review" and "Expert Review". For all other values the registration procedure is "Expert Review". The initial contents of the registry are:

Value	Information Access
1	Name: OCSF 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:
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:
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:
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:
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 11: C509 Information Accesses

## 9.8. C509 Extended Key Usages Registry

IANA has created a new registry titled "C509 Extended Key Usages Registry" under the new heading "CBOR Encoded X.509 (C509) Parameters". The columns of the registry are Value, Name, Identifiers, OID, DER, Comments, and Reference, where Value is an integer, and the other columns are text strings. For values in the interval  $[-24, 23]$  the registration procedure is "IETF Review" and "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:	
9	Name:	OCSF Signing
	Identifiers:	id-kp-OCSFSigning
	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

Figure 12: C509 Extended Key Usages

### 9.9. C509 General Names Registry

IANA has created a new registry titled "C509 General Names Registry" under the new heading "CBOR Encoded X.509 (C509) Parameters". The columns of the registry are Value, General Name, and Reference, where Value is an integer, and the other columns are text strings. For values in the interval [-24, 23] the registration procedure is "IETF Review" and "Expert Review". For all other values the registration procedure is "Expert Review". The initial contents of the registry are:

Value	General Names
-3	Name: otherName with BundleEID Comments: id-on-bundleEID (1.3.6.1.5.5.7.8.11) 06 08 2B 06 01 05 05 07 08 0B Value: eid-structure from RFC 9171
-2	Name: otherName with Smtputf8Mailbox Comments: id-on-Smtputf8Mailbox (1.3.6.1.5.5.7.8.9) 06 08 2B 06 01 05 05 07 08 09 Value: text
-1	Name: otherName with hardwareModuleName Comments: id-on-hardwareModuleName (1.3.6.1.5.5.7.8.4) 06 08 2B 06 01 05 05 07 08 04



	Value:	[ ~oid, bytes ]
0	Name:	otherName
	Comments:	
	Value:	[ ~oid, bytes ]
1	Name:	rfc822Name
	Comments:	
	Value:	text
2	Name:	dNSName
	Comments:	
	Value:	text
4	Name:	directoryName
	Comments:	
	Value:	Name
6	Name:	uniformResourceIdentifier
	Comments:	
	Value:	text
7	Name:	iPAddress
	Comments:	
	Value:	bytes
8	Name:	registeredID
	Comments:	
	Value:	~oid

Figure 13: C509 General Names

#### 9.10. C509 Signature Algorithms Registry

IANA has created a new registry titled "C509 Signature Algorithms" under the new heading "CBOR Encoded X.509 (C509) Parameters". The registry includes both signature algorithms and non-signature proof-of-possession algorithms. The columns 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. For values in the interval [-24, 23] the registration procedure is "IETF Review" and "Expert Review". For all other values the registration procedure is "Expert Review". The initial contents of the registry are:

Value	X.509 Signature Algorithms
-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: Don't use
-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: Don't use. Compressed signature value
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: Compressed signature value
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: Compressed signature value
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: Compressed signature value
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: Compressed signature value
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: Compressed signature value
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: SHA-256 with 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 public static Diffie-Hellman key.
15	Name: SHA-384 with 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 public static Diffie-Hellman key.
16	Name: SHA-512 with 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 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:	
42	Name: HSS / LMS Identifiers: id-alg-hss-lms-hashsig, id-alg-mts-hashsig OID: 1.2.840.113549.1.9.16.3.17 Parameters: Absent DER: 30 0D 06 0B 2A 86 48 86 F7 0D 01 09 10 03 11 Comments:	
43	Name: XMSS Identifiers: id_alg_xmss OID: 0.4.0.127.0.15.1.1.13.0 Parameters: Absent DER: 30 0B 06 09 04 00 7F 00 0F 01 01 0D 00 Comments:	
44	Name: XMSS^MT Identifiers: id_alg_xmssmt OID: 0.4.0.127.0.15.1.1.14.0 Parameters: Absent DER: 30 0B 06 09 04 00 7F 00 0F 01 01 0E 00 Comments:	
45	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: Compressed signature value	

Figure 14: C509 Signature Algorithms

## 9.11. C509 Public Key Algorithms Registry

IANA has created a new registry titled "C509 Public Key Algorithms" under the new heading "CBOR Encoded X.509 (C509) Parameters". The columns 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. For values in the interval [-24, 23] the registration procedure is "IETF Review" and "Expert Review". For all other values the registration procedure is "Expert Review". The initial contents of the registry are:

Value	X.509 Public Key Algorithms
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: Compressed subjectPublicKey
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: Point compressed subjectPublicKey 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: Point compressed subjectPublicKey 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: Point compressed subjectPublicKey

	Also known as P-521, ansip521r1
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:
10	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:
11	Name: Ed448 (Edwards) Identifiers: id-Ed448, id-EdDSA448 OID: 1.3.101.113 Parameters: Absent DER: 30 05 06 03 2B 65 71 Comments:
16	Name: HSS / LMS Identifiers: id-alg-hss-lms-hashsig, id-alg-mts-hashsig OID: 1.2.840.113549.1.9.16.3.17 Parameters: Absent DER: 30 0D 06 0B 2A 86 48 86 F7 0D 01 09 10 03 11 Comments:
17	Name: XMSS Identifiers: id_alg_xmss OID: 0.4.0.127.0.15.1.1.13.0 Parameters: Absent DER: 30 0B 06 09 04 00 7F 00 0F 01 01 0D 00 Comments:
18	Name: XMSS^MT Identifiers: id_alg_xmssmt OID: 0.4.0.127.0.15.1.1.14.0 Parameters: Absent

	DER:	30 0B 06 09 04 00 7F 00 0F 01 01 0E 00
	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:	Point compressed subjectPublicKey
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:	Point compressed subjectPublicKey
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:	Point compressed subjectPublicKey
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:	Point compressed subjectPublicKey
28	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:	Point compressed subjectPublicKey

Figure 15: C509 Public Key Algorithms

#### 9.11.1. Suitability of different public key algorithms for use within IoT scenarios

The public key algorithms registry Section 9.11 specify a number of algorithms, not all which are suitable for usage with constrained devices. RSA requires large keys and large signature sizes compared to elliptic curve cryptography (ECC), which together with resource-efficient implementations of named elliptic curves (Montgomery, Edwards and Weierstraß curves), make them suitable candidates for IoT public key usage. These curves are represented by ids 111 and 2428 in Section 9.11.

#### 9.12. COSE Header Parameters Registry

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

#### 9.13. 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.

IANA has registered the following media type [RFC6838]:

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

- \* 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: IESG

#### 9.14. Media Type application/cose-c509-pkcs10

When the application/cose-c509-pkcs10 media type is used, the data is a C509CertificateRequest structure.

IANA has registered the following media type [RFC6838]:

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 Certificate 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: IESG

#### 9.15. Media Type application/cose-c509-privkey

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

IANA has registered the following media type [RFC6838]:

Type name: application

Subtype name: cose-c509-privkey

Required parameters: N/A

Optional parameters: usage

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: IESG

#### 9.16. Media Type application/cose-c509-pem

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

IANA has registered the following media type [RFC6838]:

Type name: application

Subtype name: cose-c509-pem

Required parameters: N/A

Optional parameters: usage

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: IESG

#### 9.17. CoAP Content-Formats Registry

IANA is requested to add the media types "application/cose-c509-cert", "application/cose-c509-pkcs10", "application/cose-c509-privkey" and "application/cose-c509-pem" to the "CoAP Content-Formats" registry under the registry group "Constrained RESTful Environments (CoRE) Parameters".

Media Type	Encoding	ID	Reference
application/cose-c509-cert	-	TBD6	[[this document]]
application/cose-c509-pkcs10	-	TBD7	[[this document]]
application/cose-c509-privkey	-	TBD10	[[this document]]
application/cose-c509-pem	-	TBD11	[[this document]]

Figure 16: CoAP Content-Format IDs

### 9.18. TLS Certificate Types Registry

This document registers the following entry in the "TLS Certificate Types" registry under the "Transport Layer Security (TLS) Extensions" heading. The new certificate type can be used with additional TLS certificate compression [RFC8879]. C509 is defined in the same way as X.509, but uses a different value and instead of DER-encoded X.509 certificate, opaque cert\_data<1..2<sup>24</sup>-1> contains a the CBOR sequence ~C509Certificate (an unwrapped C509Certificate).

Editor's Note: The TLS registrations should be discussed and approved by the TLS WG at a later stage. The TLS WG might e.g. want a separate draft in the TLS WG.

Value	Name	Recommended	Comment
TBD5	C509 Certificate	Y	

### 9.19. CBOR Tags Registry

This document registers the following entries in the "CBOR Tags" registry under the "Concise Binary Object Representation (CBOR) Tags" heading.

Tag	X.509 Public Key Algorithms
TDB6	Data Item: COSE_C509 Semantics: An ordered chain of C509 certificates Reference: This document

## 10. References

### 10.1. Normative References

- [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/info/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/info/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/info/rfc2986>>.
- [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/info/rfc4108>>.
- [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/info/rfc5280>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [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/info/rfc8174>>.
- [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/info/rfc8610>>.
- [RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <<https://www.rfc-editor.org/info/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/info/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/info/rfc9052>>.
- [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/info/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/info/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/info/rfc9360>>.
- [SECG] "Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, ver. 2", 2009, <<https://secg.org/sec1-v2.pdf>>.

## 10.2. Informative References

- [CAB-Code] CA/Browser Forum, "CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Code Signing Certificates Version 2.3"", May 2021, <<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 1.7.6"", June 2021, <<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 2.1", February 2021, <<https://www.gsma.com/esim/wp-content/uploads/2021/02/SGP.14-v2.1.pdf>>.

## [I-D.ietf-lake-edhoc]

Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-23, 22 January 2024, <<https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-23>>.

## [I-D.ietf-tls-ctls]

Rescorla, E., Barnes, R., Tschofenig, H., and B. M. Schwartz, "Compact TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-ctls-09, 23 October 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-ctls-09>>.

## [I-D.ietf-uta-tls13-iot-profile]

Tschofenig, H., Fossati, T., and M. Richardson, "TLS/DTLS 1.3 Profiles for the Internet of Things", Work in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-profile-08, 22 October 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-uta-tls13-iot-profile-08>>.

## [IEEE-802.1AR]

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

[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/info/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/info/rfc6955>>.

[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/info/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/info/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/info/rfc7925>>.
- [RFC7932] Alakuijala, J. and Z. Szabadka, "Brotli Compressed Data Format", RFC 7932, DOI 10.17487/RFC7932, July 2016, <<https://www.rfc-editor.org/info/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/info/rfc8446>>.
- [RFC8603] Jenkins, M. and L. Ziegler, "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/info/rfc8603>>.
- [RFC8879] Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", RFC 8879, DOI 10.17487/RFC8879, December 2020, <<https://www.rfc-editor.org/info/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/info/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/info/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/info/rfc9148>>.
- [RFC9190] Preuß 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/info/rfc9190>>.

[RFC9191] Sethi, M., Preuß 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/info/rfc9191>>.

[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., Furuheid, 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](https://doi.org/10.1007/978-3-319-93797-7_14)>.

## Appendix A. Example C509 Certificates

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

Example of [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 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

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

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

```

1,                / version and certificate type /
h'01f50d',        / serialNumber /
"RFC test CA",    / issuer /
1672531200,       / notBefore /
1767225600,       / notAfter /
h'010123456789AB', / subject, EUI-64 /
1,                / subjectPublicKeyAlgorithm /
h'02B1216AB96E5B3B3340F5BDF02E693F16213A04525ED44450
  B1019C2DFD3838AB',
1,                / single extension:
                    non-critical keyUsage
                    digitalSignature /
0,                / signatureAlgorithm /
h'D4320B1D6849E309219D30037E138166F2508247DDDAE76CCE
  EA55053C108E90D551F6D60106F1ABB484CFBE6256C178E4AC
  3314EA19191E8B607DA5AE3BDA16'
```

The size of the CBOR encoding (CBOR sequence) is 139 bytes. The point compressed public key is represented as described in Section 3.2.1.

```

01
43 01 F5 0D
6B 52 46 43 20 74 65 73 74 20 43 41
1A 63 B0 CD 00
1A 69 55 B9 00
47 01 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
00
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
```

## A.1.2. Example: Natively Signed C509 Certificate

The corresponding natively signed C509 certificate in CBOR diagnostic format is identical, except for c509CertificateType, encoding of point compression (see Section 3.2.1), and signatureValue.

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

```

0,
h'01f50d',
"RFC test CA",
1672531200,
1767225600,
h'010123456789AB',
1,
h'02B1216AB96E5B3B3340F5BDF02E693F16213A04525ED44450
B1019C2DFD3838AB',
1,
0,
h'6FC903015259A38C0800A3D0B2969CA21977E8ED6EC344964D
4E1C6B37C8FB541274C3BB81B2F53073C5F101A5AC2A928865
83B6A2679B6E682D2A26945ED0B2'

```

The size of the CBOR encoding (CBOR sequence) is 139 bytes.

```

00
43 01 F5 0D
6B 52 46 43 20 74 65 73 74 20 43 41
1A 63 B0 CD 00
1A 69 55 B9 00
47 01 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
00
58 40 6F C9 03 01 52 59 A3 8C 08 00 A3 D0 B2 96 9C A2 19 77 E8 ED 6E
C3 44 96 4D 4E 1C 6B 37 C8 FB 54 12 74 C3 BB 81 B2 F5 30 73 C5 F1 01
A5 AC 2A 92 88 65 83 B6 A2 67 9B 6E 68 2D 2A 26 94 5E D0 B2

```

#### A.1.3. C509 for Diffie-Hellman keys

The two previous examples illustrate the common key usage `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 certificate request.

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

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

issuerPublicKeyAlgorithm :

1 (EC Public Key (Weierstraß) with secp256r1)

issuerPublicKey :

h'02AE4CDB01F614DEFC7121285FDC7F5C6D1D42C95647F061BA0080DF678867845E'

issuerPrivateKey :

h'DC66B3415456D649429B53223DF7532B942D6B0E0842C30BCA4C0ACF91547BB2'

subjectPrivateKey :

h'D718111F3F9BD91B92FF6877F386BDBFCEA7154268FD7F2FB56EE17D99EA16D4'

#### 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: 1.3.6.1.4.1.6175.10.1
        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):/

```

1,
h'7E7661D7B54E4632',
[
  -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'03C8B421F11C25E47E3AC57123BF2D9FDC494F028BC351CC80C03F150BF50CFF95',
[
  4, -2,
  1, h'96600D8716BF7FD0E752D0AC760777AD665D02A0',
  7, h'68D16551F951BFC82A431D0D9F08BC2D205B1160',
  -2, 5,
  3, [-1, [h'2B06010401B01F0A01', h'01020304']] / subjectAltName w. hardwareMod
uleName /
],
0,
h'C0D81996D2507D693F3C48EAA5EE9491BDA6DB214099D98117C63B361374CD86A7
74989F4C321A5CF25D832A4D336A08AD67DF20F1506421188A0ADE6D349236'

```

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

```

01 48 7E 76 61 D7 B5 4E 46 32 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 03 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 B0 1F 0A 01 44 01 02 03
04 00 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](http://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 30 37 32 39 30 30 30 30 30 5a 17 0d
32 31 30 37 32 39 31 32 30 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):/

```

1,
h'047FA1E31928EE403BA0B83A395673FC',
[
  -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'03963ECDD84DCD1B93A1CF432D1A7217D6C63BDE3355A02F8CFB5AD8994CD44E20',
[
  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",
      "http://crl4.digicert.com/CloudflareIncECCCA-3.crl"],
  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,
10, [
  h'F65C942FD1773022145418083094568EE34D131933BFDF0C2F200BCC4EF164E3',
  77922190,
  0,
  h'F8D1B4A93D2F0D4C4176DFB488BCC73B86443D7DE00E6AC8174D8948A8843668
  29FF5A34068A240C69502788E8EE25AB7ED2CBCF686ECE7B5F96B431A90702FA',
  h'5CDC4392FEE6AB4544B15E9AD456E61037FBD5FA47DCA17394B25EE6F6C70ECA',
  77922238,
  0,
  h'E891C197BFB0E3D30CB6CEE60D94C3C75FD1175336931108D89812D4D29D81D0
  A159D16C4647D1483757FCD6CE4E75EC7B5EF657EFE028F8E5CC4792682DAC43'
]
],
0,
h'BD63CF4F7E5CFE6C29385EA71CFBFC1E3F7B1CD07251A221F77769C0F471DFEA
B5C06CC45854FA30B28288B1D3BB9A6661ED5031725B1A8202E0DA5B59F95402'

```

The size of the CBOR encoding (CBOR sequence) is 783 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):/

```
1,
h'A6A55C870E39B40E',
[
  -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',
  10, [
```

```

h'F65C942FD1773022145418083094568EE34D131933BFDF0C2F200BCC4EF164E3',
1715,
0,
h'8CF54852CE5635433911CF10CDB91F52B33639223AD138A41DECA6FEDE1FE90F
BCA2254366C19A2691C47A00B5B653ABBD44C2F8BAAEF4D2DAF2527CE6454995',
h'5CDC4392FEE6AB4544B15E9AD456E61037FBD5FA47DCA17394B25EE6F6C70ECA',
2012,
0,
h'A5E0906E63E91D4FDDEFFF0352B91E50896007564B448A3828F596DC6B28726D
FC91EAED02168866054EE18A2E5346C4CC51FEB3FA10A91D2EDBF99125F86CE6'
]
],
23,
h'14043FA0BED2EE3FA86E3A1F788EA04C35530F11061FFF60A16D0B83E9D92ADB
B33F9DB3D7E0594C19A8E419A50CA770727763D5FE64510AD27AD650A58A9238
ECCB2F0F5AC064584D5C06B9736368278B8934DC79C71D3AFD345F8314415849
80682980398A867269CC7937CEE397F7DCF39588ED81032900D2A2C7BAABD63A
8ECA090BD9FB39264BFF03D88E2D3F6B21CA8A7DD85FFB94BA83DE9CFC158D61
FA672DB0C7DB3D250A414A85D37F4946373CF4B175D052F3DDC766F14BFDAA00
EDBF47EED01EC7BE4F646FC31FD72FE03D2F265AF4D7EE2819B7AFD303CF552
F40534A08A3E194158C8A8E05171840915AECA57775FA18F7D577D531CCCC72D'

```

The size of the CBOR encoding (CBOR sequence) is 1245 bytes.

#### Acknowledgments

The authors want to thank Henk Birkholz, Carsten Bormann, Russ Housley, Olle Johansson, Benjamin Kaduk, Lijun Liao, Ilari Liusvaara, Laurence Lundblade, Francesca Palombini, Thomas Peterson, Michael Richardson, Stefan Santesson, Jim Schaad, Brian Sipos, Fraser Tweedale, and Rene Struik for reviewing and commenting on intermediate versions of the draft and helping with GitHub.

#### Authors' Addresses

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

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

Shahid Raza  
RISE AB  
Email: shahid.raza@ri.se



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

Martin Furuhed  
Nexus Group  
Email: martin.furuhed@nexusgroup.com