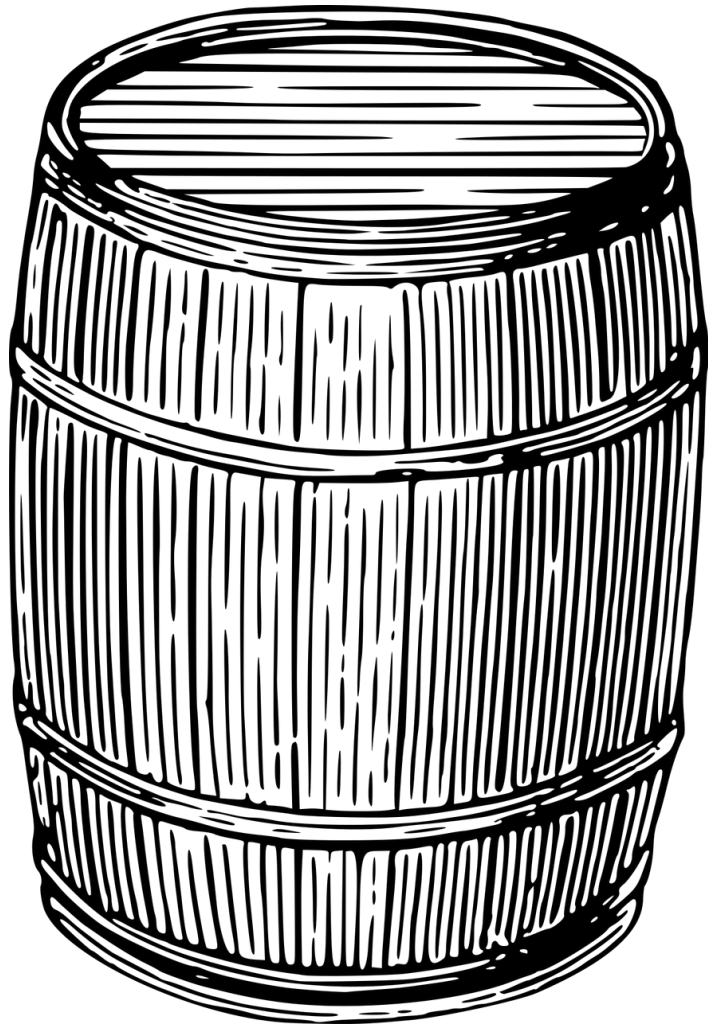


CBOR Encoding of X.509 Certificates (CBOR Certificates)

draft-mattsson-cose-cbor-cert-compress-03

COSE, IETF 109, John Preuß Mattsson

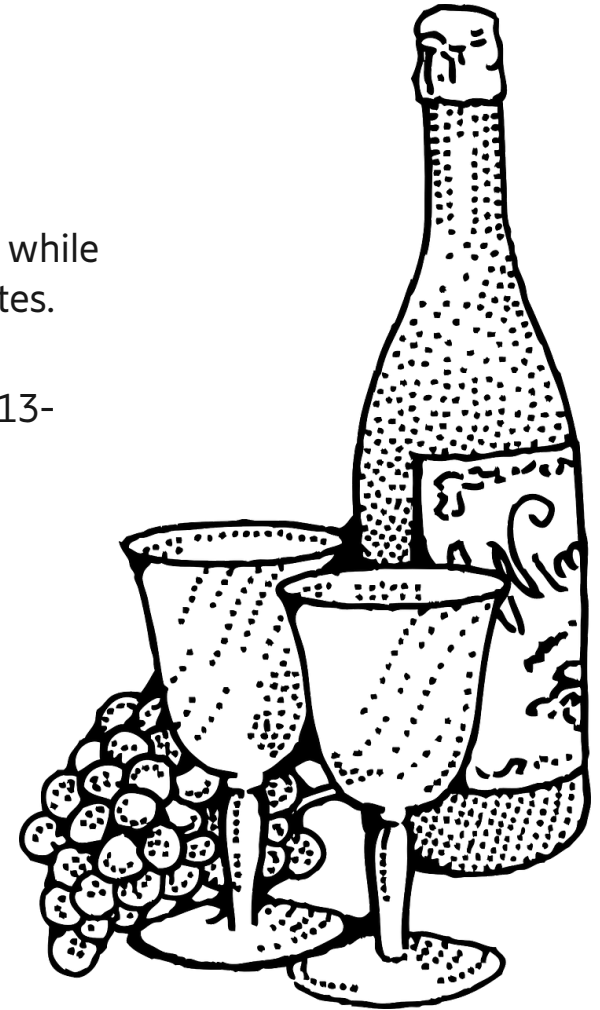


High level Discussion



CBOR Certificates Scope

- High level goal: *"Make sweet CBOR wine from sour ASN.1 grapes"*
- Began as optimal CBOR encoding of a small subset of RFC 7925. Now the discussed scope is more general. Need to support a larger subset of RFC 5280 while still achieving close to optimal encoding for the most constrained IoT certificates.
- Certificate profiles that have been mentioned are RFC 7925, draft-ietf-uta-tls13-iot-profile, IEEE 802.1AR, but also other parts of RFC 5280 like GeneralizedTime and COSE related EKUs, etc.
- Best approach is probably to define CBOR encoding for a quite large subset of RFC 5280 so it works with existing and future X.509 profiles.
 - We believe that this can be done while keeping the size of constrained IoT certificates close to the current size (example certificate is 138 bytes).
- Make registrations so that CBOR certificates can be used in COSE and TLS.



CBOR Certificates Process



To discuss encodings we first need to decide on what to support:

- Proposal to do compressed X.509 certs and natively signed certs.
- ASN.1 encodings include e.g. BER, DER, CER, PER, CPER, XER, CXER, EXER, OER, JER, GSER, SER, LWER, MBER.
 - Proposal to only support compression of DER encoded certificates.
- IETF PKIX (RFC 5280) is itself a profile of ITU-T X.509. Due to the popularity of PKIX, X.509 nowadays usually refers to IETF PKIX.
 - Proposal to do encoding for actively used parts of RFC 5280, as well as future additions.
 - Separate encoding specification from profile specification
 - Details discussed on following slides.

ASN.1



CDDL



```
Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING }

TBSCertificate ::= SEQUENCE {
    version             [0] EXPLICIT Version DEFAULT v1,
    serialNumber        INTEGER,
    signature            AlgorithmIdentifier,
    issuer              Name,
    validity            Validity,
    subject             Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID      [1] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
    subjectUniqueID     [2] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
    extensions          [3] EXPLICIT Extensions OPTIONAL
    -- If present, version MUST be v3
}
```

```
; This defines a CBOR Sequence:
CBORCertificate = [
    TBSCertificate,
    issuerSignatureValue : bytes,
]

TBSCertificate = (
    cborCertificateType : int,
    certificateSerialNumber : bytes,
    issuerSignatureAlgorithm : Algorithm,
    issuer : Name,
    validityNotBefore : bytes,
    validityNotAfter : bytes,
    subject : Name,
    subjectPublicKeyAlgorithm : Algorithm,
    subjectPublicKey : bytes,
    extensions : [ * Extension ] / int,
)
```

■ = removed
■ = split
■ = added

- Proposed high level format is an RFC 8742 CBOR Sequence (Saves 1 byte compared to CBOR array)
 - Proposed to only include 1 signature algorithm. Proposal to keep the field in TBSCertificate that is signed.
 - Proposed to not support issuerUniqueID, subjectUniqueID
 - Proposed to introduce 'type' to differentiate between compressed X.509 and natively signed.
 - Proposed to split validity and subjectPublicKeyInfo into two elements (saves 2 bytes compared to arrays)



List of Changes

Changes from -01 to -02 to -03



Changes from -01 to -02

- Changed terminology to "natively signed"
- Completely changed the encoding of issuer and subject. The new encoding supports encoding of sequences of sets of attribute types and values. The encoding handle any attribute type encoded as utf8string and printableString.
- Moved signatureAlgorithm so it comes before signatureValue. Made the algorithm identifiers explicit (no default) and split them into two items. This increases size with 2 bytes.
- Extension text moved to separate paragraph.
- Split CDDL into certificate and tbsCertificate similar to RFC 5280 to not have to duplicate CDDL.
- New section of compliance requirements.

Changes from -02 to -03

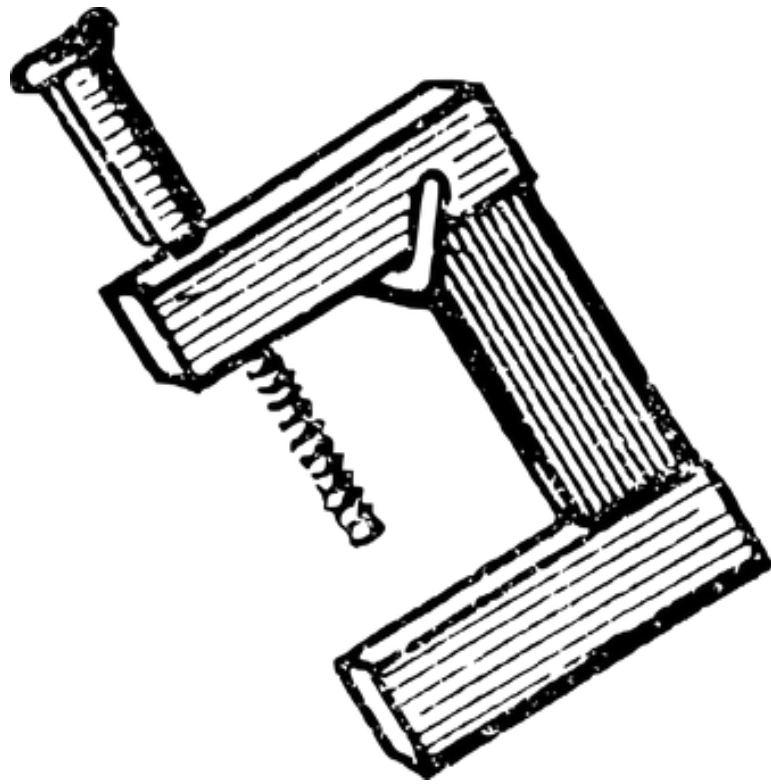
- Added GeneralizedTime as suggested by Jim. We also addresses an issue in that the old encoding could not encode leap seconds (which X.509 can).
- Changed from TLS compression algorithms to TLS certificate Type as suggested by Ilari. This also enables CBOR certificates to be used with a general compression algorithm in TLS.
- Added support for a lot more algorithms. Registered values for most algorithms without parameters.
- More strict text on several parts regarding encoding.

Changes from -01 to -02 to -03



Changes from -02 to -03

- Tried to reduce the number of negative ints.
- Update title based on comment from Jim
- Updated abstract and intro so it is up to date with the document.
- Added text that the string types teletexString, universalString, and bmpString are not supported.
- Allowed raw relative OIDs for algorithms and extensions.
- Added more detail on how ECDSA signatureValue is encoded
- Updated CDDL to follow RFC 8742 recommendations
- Fixed so that an empty subject can be encoded.
- Completely rewrote extension encoding as the old description was underspecified, had several problems, and several limitations.
- Added IANA registries for extensions, EKU, and subjectAltname. Added registration procedures for the IANA registries.
- Added c5bag and c5t as well.
- Specified how EUI-64 is compressed to 6 or 8 bytes
- The encoding of the tools.ietf.org certificate was added as an example.
- Added acknowledgements



Low level Discussion

version, issuerUniqueID, subjectUniqueID, type



— version:

- RFC 5280: Version MUST be 3 (value is 2).
- Draft -03 assumes v3 and does not explicitly encode it.

— issuerUniqueID, subjectUniqueID:

- issuerUniqueID and subjectUniqueID are seldom used.
- Certificate extensions in version 3 replace the functionality of these members.
- Draft -03 does not support issuerUniqueID, subjectUniqueID.

— cborCertificateType:

- Draft -03 defines 'type' to differentiate between compressed X.509 and natively signed.
- Can be used for other purposes in the future if needed.

```
cborCertificateType : int,
```

| Value | Description |
|-------|-----------------------------------|
| 0 | Natively Signed CBOR Certificate |
| 1 | CBOR Compressed X.509 Certificate |

serialNumber, signatureValue



— serialNumber:

- RFC 5280: serialNumber MUST be a positive integer. Certificate users MUST be able to handle serialNumber values up to 20 octets. Conforming CAs MUST NOT use serialNumber values longer than 20 octets.
- Draft -03: encode as a CBOR byte string. Encoded as in a DER encoded INTEGER value field.
- CBOR unsigned integers are not suitable as they only support 8-, 16-, 32-, and 64-bit integers.
- CBOR bignum could be used, but is not as widely supported as bytes, and would increase size with 1 byte.

```
certificateSerialNumber : bytes,
```

— signatureValue:

- Draft -03: encode as a CBOR byte string. Encoded as in a DER encoded BIT STRING value field (assuming zero unused bits). Special handling of ECDSA.
- **ECDSA signature (r, s) is encoded as length $2 * (\text{ceil}(\log_2(n) / 8)$, where n is the curve order. Is that correct also for Wei25519/W-25519 and Wei448/W-448?**
 - Cf. draft-lwig-curve-representations/
NIST SP 800-186

```
issuerSignatureValue : bytes,
```


subjectPublicKeyInfo, signatureAlgorithm, signature



— subjectPublicKey:

- Draft -03: split up into algorithm and subjectPublicKey instead of encoding as CBOR array.
- Draft -03: encode subjectPublicKey as CBOR byte string. Encoded as in a DER encoded BIT STRING value field (assuming zero unused bits). Special handling of id-ecPublicKey.
- id-ecPublicKey is point compressed as specified in SECG.
- **Support point compressed X.509 certs in future version?**

```
subjectPublicKey : bytes,
```

— signatureAlgorithm, signature:

- Duplicate. Draft -03 only includes one of these fields and calls it issuerSignatureAlgorithm.

— issuerSignatureAlgorithm and subjectPublicKeyAlgorithm:

- RFC 5280: AlgorithmIdentifier is defined as OID + parameters of type ANY.
- Most algorithms omit parameters. rsaEncryption Public Keys and PKCS #1 v1.5 Signatures has parameters = NULL. Old RSASSA-PSS algorithms and old ECC public keys use parameters.
- Draft -03: no support for parameters. RSASSA-PSS is only supported with SHAKE.
- Draft -03: encode signature and public key algorithms as ints from different registries.
- id-ecPublicKey only supported with named curves.
- **Should the draft allow raw DER encoded relative OIDs?**
- **Should any algorithms with parameters be supported?**

```
issuerSignatureAlgorithm : Algorithm,  
subjectPublicKeyAlgorithm : Algorithm,  
Algorithm = int / relativeOID  
relativeOID = bytes
```


issuerSignatureAlgorithm and subjectPublicKeyAlgorithm



— issuerSignatureAlgorithm and subjectPublicKeyAlgorithm:

- Which algorithms should values initially be registered for? Draft -03 registers algs below but to differentiate between 1 and 2 byte identifiers. Wei25519/W-25519 and Wei448/W-448 should also be added.
- Weak signature algorithms with SHA-1 are still used in root certificates.

Public Key Algorithms

=====

rsaEncryption
id-ecPublicKey + secp256r1
id-ecPublicKey + secp384r1
id-ecPublicKey + secp521r1
id-X25519
id-X448
id-Ed25519
id-Ed448
id-alg-hss-lms-hashsig
id-alg-xmss
id-alg-xmssmt

Signature Algorithms

=====

sha-1WithRSAEncryption

sha224WithRSAEncryption

sha256WithRSAEncryption

sha384WithRSAEncryption

sha512WithRSAEncryption

id-rsassa-pkcs1-v1_5-with-sha3-224

id-rsassa-pkcs1-v1_5-with-sha3-256

id-rsassa-pkcs1-v1_5-with-sha3-384

id-rsassa-pkcs1-v1_5-with-sha3-512

id-RSASSA-PSS-SHAKE128

id-RSASSA-PSS-SHAKE256

ecdsa-with-SHA1

ecdsa-with-SHA224

ecdsa-with-SHA256

ecdsa-with-SHA384

ecdsa-with-SHA512

id-ecdsa-with-sha3-224

id-ecdsa-with-sha3-256

id-ecdsa-with-sha3-384

id-ecdsa-with-sha3-512

id-ecdsa-with-shake128

id-ecdsa-with-shake256

id-Ed25519

id-Ed448

id-alg-hss-lms-hashsig

id-alg-xmss

id-alg-xmssmt

issuer, subject



— issuer, subject:

- RFC 5280: both issuer and subject are defined as:
SEQUENCE OF SET OF { type AttributeType, value AttributeValue }
- Draft -03: encode this as a CBOR array of CBOR arrays of int keys and text values.
 - Draft -03: omit the outer array when there is a single RDN.
 - Draft -03: omit the map when there is a single utf8string encoded id-at-commonName.
 - Draft -03: encode as bytes instead of text when there is a single EUI-64.
 - **What is the difference between EUI-48, MAC-48, and 48-bit MAC address?**
 - Unspecified how a EUI-64 is encoded in X.509 (xx-xx-xx-FF-FE-xx-xx-xx, xx-xx-xx-FF-FF-xx-xx-xx, something else?). **EUI-64 encoding of MAC-48 deprecated?**
 - Are EUI-64 are created from 48-bit MAC addresses? Having a fixed MAC address goes against the current trend of MAC Address randomization for privacy.

```
issuer : Name
subject : Name
Name = [ * RelativeDistinguishedName ] / RelativeDistinguishedName
RelativeDistinguishedName = [ + Attribute ] / text / bytes
Attribute = ( attributeType : int, attributeValue : text )
```


- **AttributeType, AttributeValue:**

- RFC 5280 defines a large number of attribute types and character string types.
- Draft -03: only register values for the attribute types which MUST and SHOULD be supported, according to RFC 5280. Draft -03 encodes attribute type as absolute value of CBOR int.
- **Are there any other attribute types that should be supported?**
- RFC 5280: MUST use either the PrintableString or UTF8String. Draft -03: only support PrintableString and UTF8String. **Any other character string type needed?**
- Draft -03: encode character string type (PrintableString or UTF8String) as sign of CBOR int.
- Each character in a PrintableString only contains 6.2 bits of information. **Compress?**

| Value | X.509 Attribute Type |
|-------|------------------------------|
| 1 | id-at-commonName |
| 2 | id-at-surname |
| 3 | id-at-serialNumber |
| 4 | id-at-countryName |
| 5 | id-at-localityName |
| 6 | id-at-stateOrProvinceName |
| 7 | id-at-organizationName |
| 8 | id-at-organizationalUnitName |
| 9 | id-at-title |
| 10 | id-at-givenName |
| 11 | id-at-initials |
| 12 | id-at-generationQualifier |
| 13 | id-at-dnQualifier |
| 14 | id-at-pseudonym |

validity



- **validity:**

- Draft -03: split up into notBefore and notAfter instead of encoding as CBOR array.
- Draft -03: support both UTCTime and GeneralizedTime. Encoded as a byte string, which is interpreted as an unsigned integer n in network byte order, using the following invertible encoding (Horner's method with different bases). UTCTime and GeneralizedTime are encoded as a byte strings of length 4 and 5 respectively.
 - UTCTime: $n = SS + 61 * (MM + 60 * (HH + 24 * (dd + 32 * (mm + 13 * yy))))$
 - GeneralizedTime: $n = SS + 61 * (MM + 60 * (HH + 24 * (dd + 32 * (mm + 13 * yyyy))))$
- Decoding can be done by a succession of modulo and subtraction operations. I.e. $SS = n \bmod 61$, $MM = ((n - SS) / 61) \bmod 60$, etc.
- Encoding is strictly increasing so it is easy to check if compressed value < current time. Cannot directly calculate distance between encoded values without decoding.

`validityNotBefore: bytes,
validityNotAfter: bytes,`

- Unix time could be used, but unix time does not handle leap seconds, unix time functions are not available on all platforms, many existing time functions use local time, and unix time is tricky to implement incorrectly due to complication leap year definitions.
- **Comments?**

extensions



— extensions:

- Based on the discussion on the list, the restricted set of extensions in RFC 7925 is insufficient.
- Draft -03: encode extensions as an [* (int, ? any)] array.
- Draft -03: allow raw DER encoded relativeOIDs. A lot of extensions defined. Some private.
- **Which extensions should be supported? Which extended key usage should be supported? Which types of Subject Alt Names should be supported?**
- Draft -03 defines CBOR encoding for the RFC 7925 extensions and try to minimize their encoding. Which other extensions should be given CBOR encoding? Can we define a general DER-> CBOR encoding to use when encoding size is not critical?

```
extensions : [ * Extension ] / int
Extension = ExtensionReg // ExtensionRaw
ExtensionReg = (
    extensionType : int,
    ? extensionValue : any, ; optionality and type known from extensionType
)
ExtensionRaw = (
    extensionID : relativeOID,
    ? critical : bool,
    ? extensionValue : bytes,
)
```


Extensions



- Limited amount of positive one-byte integers [1,23]. Need to determine which extensions that should be allocated in that range. **Should we reserve integers for future extensions?**

| Value | X.509 Extension Type | extensionValue |
|-------|-------------------------------------|-----------------|
| 1 | id-ce-basicConstraints (cA = false) | int |
| 2 | id-ce-basicConstraints (cA = true) | |
| 3 | id-ce-basicConstraints (cA = true) | |
| 4 | id-ce-keyUsage | |
| 5 | id-ce-keyUsage + 1 | int |
| 6 | id-ce-keyUsage + 16 | |
| 7 | id-ce-keyUsage + 17 | |
| 8 | id-ce-keyUsage + 32 | |
| 9 | id-ce-keyUsage + 33 | [] / int / rOID |
| 10 | id-ce-keyUsage + 48 | |
| 11 | id-ce-keyUsage + 49 | |
| 12 | id-ce-extKeyUsage | |
| 13 | id-ce-subjectAltName | [] / text |
| 14 | id-ce-authorityKeyIdentifier | bytes |
| 15 | id-ce-subjectKeyIdentifier | bytes |
| 16 | id-ce-certificatePolicies | bytes |
| 17 | id-ce-cRLDistributionPoints | bytes |
| 18 | id-pe-authorityInfoAccess | bytes |
| 19 | SCT List (1.3.6.1.4.1.11129.2.4.2) | bytes |
| 248 | id-ce-nameConstraints | bytes |
| 249 | id-ce-policyConstraints | bytes |
| 250 | id-ce-inhibitAnyPolicy | bytes |
| 251 | id-ce-authorityKeyIdentifier | bytes |
| 252 | id-ce-policyMappings | bytes |
| 253 | id-ce-issuerAltName | bytes |
| 254 | id-ce-subjectDirectoryAttributes | bytes |
| 255 | id-ce-freshestCRL | bytes |
| 256 | id-pe-subjectInfoAccess | bytes |

| Value | Extended Key Usage |
|-------|-----------------------|
| 0 | anyExtendedKeyUsage |
| 1 | id-kp-serverAuth |
| 2 | id-kp-clientAuth |
| 3 | id-kp-codeSigning |
| 4 | id-kp-emailProtection |
| 5 | id-kp-timeStamping |
| 6 | id-kp-OCSPSigning |

| Value | Subject Alternative Name | |
|-------|---------------------------|-------|
| 0 | rfc822Name | text |
| 1 | dNSName | text |
| 2 | directoryName | Name |
| 3 | uniformResourceIdentifier | text |
| 4 | iPAddress | bytes |

■ = RFC 7925, 1-byte

■ = Other, 1-byte



Use in
COSE and TLS

COSE and TLS



— COSE:

- Draft -03 registers 'c5bag', 'c5chain', 'c5u', and 'c5t' similar to 'x5bag', 'x5chain', 'x5u', and 'x5t'

— TLS

- Draft -03 registers CBOR certificates as a new TLS certificate type.
- This allows CBOR certificates to be used with certificate compression (for constrained IoT certificates, the current general certificate compression mechanisms increase size. For non-IoT certificates with many character strings, certificate compression is expected to decrease size).

| Name | Label | Value Type | Description |
|---------|-------|----------------|---------------------------------------|
| c5bag | TBD1 | COSE_CBOR_Cert | An ordered chain of CBOR certificates |
| c5chain | TBD2 | COSE_CBOR_Cert | An ordered chain of CBOR certificates |
| c5t | TBD3 | COSE_CertHash | Hash of an CBOR certificate |
| c5u | TBD4 | uri | URI pointing to a CBOR certificate |

| Value | Name | Recommended | Comment |
|-------|------------------|-------------|---------|
| TBD3 | CBOR Certificate | Y | |


Example Encoding of a HTTPS Certificate



- The DER encoding of the tools.ietf.org certificate is 1647 bytes.
- The CBOR encoding (CBOR sequence) of the CBOR certificate is 1374 bytes (16% reduction).
- Non-RFC 7925 extension values are DER encoded byte strings.

 = RFC 7925

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

```
1,
h'A6A55C870E39B40E',
0,
[
  [4, "US"],
  [6, "Arizona"],
  [5, "Scottsdale"],
  [7, "Starfield Technologies, Inc."],
  [8, "http://certs.starfieldtech.com/repository/"],
  [1, "Starfield Secure Certificate Authority - G2"]
],
h'2D3EE7F6',
h'2F98B716',
[
  [8, "Domain Control Validated"],
  [-1, "/*.tools.ietf.org"]
],
0,
h'3082010A0282010100B1E137E8EB82D689FADBF5C24B77F02C4ADE726E3E
CFF3E728AF37D8B67BDDF37EAE6E977FF7CA694ECCD006DF5D279B3B12E7E6
1E242BDD79D8530126ED284FC98694834EC8E1142E85B3AFD46EDD6946AF41
[
  -1,
 12, [ 1, 2 ],
  -4, 5,
  17, h'30343032a030a02e862c687474703a2f2f63726c2e7374617266
  16, h'305A304E060B6086480186FD6E01071701303F303D06082B0601
  18,
  h'3074302A06082B06010505073001861E687474703A2F2F6F6373702E7374
  65706F7369746F72792F73666967322E637274',
  14, h'30168014254581685026383D3B2D2CBECD6AD9B63DB36663',
 13, [ 1, "/*.tools.ietf.org", 1, "tools.ietf.org" ],
  15, h'0414AD8AB41C0751D7928907B0B784622F36557A5F4D',
  19,
  h'0481F400F2007700F65C942FD1773022145418083094568EE34D131933BF
  C19A2691C47A00B5B653ABBD44C2F8BAAEF4D2DAF2527CE64549950077005C
```


How to progress until next meeting



- Reviews
- Implementations
- Discussion on the list