Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

S. Raza
J. Hoeglund
RISE AB
G. Selander
J. Mattsson
Ericsson AB
M. Furuhed
Nexus Group

November 02, 2020

# CBOR Profile of X.509 Certificates

draft-mattsson-cose-cbor-cert-compress-02

## Abstract

This document specifies a CBOR encoding/compression of RFC 7925 profiled certificates. By using the fact that the certificates are profiled, the CBOR certificate compression algorithms can in many cases compress RFC 7925 profiled certificates with over 50%. This document also specifies COSE headers for CBOR encoded certificates as well as the use of the CBOR certificate compression algorithm with TLS Certificate Compression in TLS 1.3 and DTLS 1.3.

## Status of This Memo



## Copyright Notice

Table of Contents



## 1. Introduction

One of the challenges with deploying a Public Key Infrastructure (PKI) for the Internet of Things (IoT) is the size and encoding of X.509 public key certificates [RFC5280], since those are not optimized for constrained environments [RFC7228]. More compact

certificate representations are desirable. Due to the current PKI usage of X.509 certificates, keeping X.509 compatibility is necessary at least for a transition period. However, the use of a more compact encoding with the Concise Binary Object Representation (CBOR) [RFC7049] reduces the certificate size significantly which has known





performance benefits in terms of decreased communication overhead, power consumption, latency, storage, etc.

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. [RFC8610] 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 [RFC7049], [RFC8610], and [RFC8742].

RFC 7925 [RFC7925] specifies a certificate profile for Internet of Things deployments which can be applied for lightweight certificate based authentication with e.g. TLS [RFC8446], DTLS [I-D.ietf-tls-dtls13], COSE [RFC8152], or EDHOC [I-D.ietf-lake-edhoc]. This document specifies the CBOR encoding/ compression of [RFC7925] profiled X.509 certificates based on [X.509-IoT]. Two variants are defined using exactly the same CBOR encoding and differing only in what is being signed:

- o The CBOR compressed X.509 certificate, which can be decompressed into a certificate that can be verified by code compatible with [RFC7925].

- o The "natively signed" CBOR encoded certificate, which further

optimizes the performance in constrained environments but is not backwards compatible with [RFC7925], see Section 7.

Other work has looked at reducing the size of X.509 certificates. The purpose of this document is to stimulate a discussion on CBOR based certificates: what field values (in particular for 'issuer'/'subject') are relevant for constrained IoT applications, what is the potential savings that can be expected with the proposed encoding, and what is the right trade-off between compactness and generality. The current version specifies a certificate encoding which can support large parts of [RFC5280], and at the same time can





maintain a small message size for certificates compatible with [RFC7925].

This document specifies COSE headers for use of the CBOR certificate encoding with COSE. The document also specifies the CBOR certificate compression algorithm for use as TLS Certificate Compression with TLS 1.3 and DTLS 1.3.

## 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 [RFC7228].

## 3. CBOR Encoding

This section specifies the content and encoding for CBOR certificates, with the overall objective to produce a very compact representation of the certificate profile defined in [RFC7925]. The CBOR certificate can be either a CBOR compressed 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 CBOR certificate, in which case the signature is calculated directly on the CBOR encoded data (see Section 7). In both cases the certificate content is adhering to the restrictions given by [RFC7925]. The

corresponding ASN.1 schema is given in Appendix A.

## 3.1. Message Fields

The encoding and compression has several components including: ASN.1 DER and base64 encoding are replaced with CBOR encoding, static fields are elided, and elliptic curve points are compressed. The X.509 fields and their CBOR encodings are listed below. Combining these different components reduces the certificate size significantly, which is not possible with general purpose compressions algorithms, see Figure 1.

CBOR certificates are defined in terms of [RFC7925] profiled X.509 certificates:

o version. The 'version' field is known (fixed to v3), and is omitted in the CBOR encoding.





o serialNumber. The 'serialNumber' field is encoded as a CBOR byte string. This allows encoding of all lengths with minimal overhead.

o signatureAlgorithm. The 'signatureAlgorithm' field is encoded as a CBOR int (see Section 9).

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

o issuer. In the general case, the sequence of 'RelativeDistinguishedName' is encoded as CBOR array of CBOR maps, where each AttributeTypeAndValue is encoded as a (CBOR int, CBOR byte string) pair. Each AttributeType is encoded as a CBOR int (see Figure 3), where the sign is used to represent the character string type; positive for printableString, negative for utf8String. If only a single 'RelativeDistinguishedName' is present, the array is omitted and issuer is encoded as a CBOR map. If a RelativeDistinguishedName contains a single AttributeTypeAndValue containing an utf8String encoded 'common name', the AttributeValue is encoded as a CBOR text string. If

the utf8String encoded 'common name' contains an EUI-64 mapped from a 48-bit MAC address it is encoded as a CBOR byte string of length 6. Other EUI-64 is encoded as a CBOR byte string of length 8.

o validity. The 'notBefore' and 'notAfter' UTCTime fields are ASCII string of the form "yymmddHHMMSSZ". They are encoded as the unsigned integers using the following invertible encoding (Horner's method with different bases). The resulting integer n always fit in a 32 bit usigned integer.

n = SS + 60 * (MM + 60 * (HH + 24 * (dd + 32 * (mm + 13 * yy))))

Decoding can be done by a succession of modulo and substraction operations. I.e. SS = n mod 60, MM = ((n - SS) / 60) mod 60, etc.

o subject. The 'subject' is encoded exactly like issuer.

o subjectPublicKeyInfo. The 'algorithm' field is encoded as a CBOR int (see Section 9). The 'subjectPublicKey' field is encoded as a CBOR byte string. Public keys of type id-ecPublicKey are point compressed as defined in Section 2.3.3 of [SECG].

o extensions. The 'extensions' field is encoded as a CBOR array where each extension is represented with an int. The extensions





mandated to be supported by [RFC7925] is encodeded as specified in Section 3.2.

o signatureValue. Since the signature algorithm and resulting signature length are known, padding and extra length fields which are present in the ASN.1 encoding are omitted and the 'signatureValue' field is encoded as a CBOR byte string. For natively signed CBOR certificates the signatureValue is calculated over the certificate CBOR sequence excluding the signatureValue.

In addition to the above fields present in X.509, the CBOR encoding introduces an additional field:

o type. A CBOR int used to indicate the type of CBOR certificate. Currently, type can be a natively signed CBOR certificate (type =

0) or a CBOR compressed X.509 certificates (type = 1), see Section 9.

The following Concise Data Definition Language (CDDL) defines certificate and tbsCertificate as groups, which are encoded as CBOR Sequences [RFC8742]. The member names therefore only have documentary value.

```
certificate = (
   tbsCertificate,
   signatureValue : bytes,
)

tbsCertificate = (
   type : int,
   serialNumber : bytes,
   signatureAlgorithm : int,
   issuer : [ 2* DistinguishedName ] / DistinguishedName,
   validity_notBefore: uint,
   validity_notAfter: uint,
   subject : [ 2* DistinguishedName ] / DistinguishedName,
   subjectPublicKeyInfo_algorithm : int,
   subjectPublicKeyInfo_subjectPublicKey : bytes,
   extensions : [ 2* extension ] / extension,
)

DistinguishedName = { + int => bytes } / text / bytes

extension = (int, ? text / bytes)
```





## 3.2. Encoding of Extensions

NOTE: The discussions in the COSE WG seems to indicate that a much larger set of extensions should be supported. This will likely result in a completly different encoding than the one below, which is very [RFC7925] focused.

This section details the encoding of the 'extensions' field. Each

extension is represented with an int. Critical extensions are encoded with a negative sign. The boolean values (digitalSignature, keyAgreement, etc.) are set to 0 or 1 according to their value in the DER encoding. If the array contains a single int, 'extensions' is encoded as the int instead of an array. pathLenConstraint is limited to a max value of 10. If subjectAltName is present, the value is placed after the int the end of the array encoded as a byte or text string following the encoding rules for the subject field.

```
subjectAltName = 1

basicConstraints = 2 + pathLenConstraint

keyUsage = 12 + digitalSignature
         + 2 * keyAgreement + 4 * keyCertSign

extKeyUsage = 19 + id-kp-serverAuth + 2 * id-kp-clientAuth
            + 4 * id-kp-codeSigning + 8 * id-kp-OCSPSigning
```

Consequently:

- o A non-critical subjectAltName is encoded as 1. A critical subjectAltName is encoded as -1.
- o A critical basicConstraints (cA = 1) without pathLenConstraint is encoded as -2.
- o A non-critical keyUsage (digitalSignature = 0, keyAgreement = 1, keyCertSign = 0) is encoded as 14 (= 12 + 2).
- o A non-criticical extKeyUsage (id-kp-serverAuth = 0, id-kp-clientAuth = 0, id-kp-codeSigning = 1, id-kp-OCSPSigning = 1) is encoded as 31 (= 19 + 4 + 8).

Thus, a critical basicConstraints (cA = 1) followed by a non-critical keyUsage (digitalSignature = 0, keyAgreement = 1, keyCertSign = 0) is encoded as [-2, 14]. A single critical subjectAltName (dNSName = "for.example") is encoded as [-1, "for.example"].





## 4. Compliance Requirements for Constrained IoT

For general purpose applications, the normative requirements of [RFC5280] applies. This section describes the mandatory to implement algorithms and OIDs for constrained IoT application; the values of the OIDs including certificate fields and extensions, time format, attributes in distinguished names, etc.

TODO: Write this section

## 5. Deployment settings

CBOR certificates can be deployed with legacy X.509 certificates and CA infrastructure. In order to verify the signature, the CBOR certificate is used to recreate the original X.509 data structure to be able 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 CBOR 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 CBOR 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 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 CBOR certificate and does not perform the conversion to X.509. This option is viable when client authentication can be asserted by other means.

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

## 6. Expected Certificate Sizes

The CBOR encoding of the sample certificate given in Appendix A results in the numbers shown in Figure 1. After [RFC7925] profiling, most duplicated information has been removed, and the remaining text strings are minimal in size. Therefore the further size reduction reached with general compression mechanisms will be small, mainly corresponding to making the ASN.1 endcoding more compact. The zlib number was calculated with zlib-flate.

```
zlib-flate -compress < cert.der > cert.compressed
```

| | RFC 7925 | zlib | CBOR Certificate |
|---|---|---|---|
| Certificate Size | 314 | 295 | 138 |

Figure 1: Comparing Sizes of Certificates (bytes)

## 7. Natively Signed CBOR Certificates

The difference between CBOR compressed X.509 certificate and natively signed CBOR certificate is that the signature is calculated over the CBOR encoding of the CBOR sequence tbsCertficate rather than the DER encoded ASN.1 data. This removes entirely the need for ASN.1 DER and base64 encoding which reduces the processing in the authenticating devices, and avoids known complexities with these encodings.

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

CBOR compressed X.509 certificates provides an intermediate step between [RFC7925] profiled X.509 certificates and natively signed CBOR certificates: An implementation of CBOR compressed X.509 certificates contains both the CBOR encoding of the X.509 certificate and the signature operations sufficient for natively signed CBOR certificates.

## 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.

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 5 requires unencrypted certificates and

is not recommended.





## 9. IANA Considerations

For all items, the 'Reference' field points to this document.

### 9.1. CBOR Certificate Types Registry

IANA has created a new registry titled "CBOR Certificate Types" under the new heading "CBOR Certificate". The registration procedure is "Expert Review". The columns of the registry are Value, Description, and Reference, where Value is an integer and the other columns are text strings. The initial contents of the registry are:

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

Figure 2: CBOR Certificate Types

### 9.2. CBOR Attribute Type Registry

IANA has created a new registry titled "CBOR Attribute Type Registry" under the new heading "CBOR Certificate". The registration procedure is "Expert Review". The columns of the registry are Value, X.509 Attribute Type, and Reference, where Value is an integer and the other columns are text strings. The initial contents of the registry are:

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

Figure 3: CBOR Attribute Type Registry

### 9.3. CBOR Certificate Signature Algorithms Registry

IANA has created a new registry titled "CBOR Certificate Signature Algorithms" under the new heading "CBOR Certificate". The registration procedure is "Expert Review". The columns of the registry are Value, X.509 Algorithm, and Reference, where Value is an integer and the other columns are text strings. The initial contents of the registry are:

| Value | X.509 Signature Algorithm |
|---|---|

| | |
|---|---|
| 0 | ecdsa-with-SHA256 |
| 1 | ecdsa-with-SHA384 |
| 2 | ecdsa-with-SHA512 |
| 3 | id-ecdsa-with-shake128 |
| 4 | id-ecdsa-with-shake256 |
| 5 | id-Ed25519 |
| 6 | id-Ed448 |

Figure 4: CBOR Certificate Signature Algorithms





## 9.4. CBOR Certificate Public Key Algorithms Registry

IANA has created a new registry titled "CBOR Certificate Public Key Algorithms" under the new heading "CBOR Certificate". The registration procedure is "Expert Review". The columns of the registry are Value, X.509 Algorithm, and Reference, where Value is an integer and the other columns are text strings. The initial contents of the registry are:

| Value | X.509 Public Key Algorithm |
|---|---|
| 0 | id-ecPublicKey + prime256v1 |
| 1 | id-ecPublicKey + prime384v1 |
| 2 | id-ecPublicKey + prime512v1 |
| 3 | id-X25519 |
| 4 | id-X448 |
| 5 | id-Ed25519 |
| 6 | id-Ed448 |

Figure 5: CBOR Certificate Public Key Algorithms

## 9.5. COSE Header Parameters Registry

This document registers the following entries in the "COSE Header Parameters" registry under the "CBOR Object Signing and Encryption (COSE)" heading. The formatting and processing are the same as the corresponding x5chain and x5u defined in [I-D.ietf-cose-x509] except that the certificates are CBOR encoded instead of DER encoded.

| Name | Label | Value Type | Description |
|---|---|---|---|
| CBORchain | TBD1 | COSE_CBOR_Cert | An ordered chain of CBOR certificates |
| CBORu | TBD2 | uri | URI pointing to a CBOR certificate |

## 9.6. TLS Certificate Compression Algorithm IDs Registry

This document registers the following entry in the "Certificate Compression Algorithm IDs" registry under the "Transport Layer Security (TLS) Extensions" heading.





| Algorithm Number | Description |
|---|---|
| TBD3 | CBOR Certificate |

# 10. References

## 10.1. Normative References

[I-D.ietf-tls-certificate-compression]
Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", draft-ietf-tls-certificate-compression-10 (work in progress), January 2020.

[I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version

1.3", draft-ietf-tls-dtls13-38 (work in progress), May 2020.

[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>.

[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>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[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>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.





[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>.

[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>.

[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>.

## 10.2. Informative References

[I-D.ietf-cose-x509]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates", draft-ietf-cose-x509-07 (work in progress), September 2020.

[I-D.ietf-lake-edhoc]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-ietf-lake-edhoc-01 (work in progress), August 2020.

[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>.

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

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





# Appendix A. Example CBOR Certificates

## A.1. Example 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 2020 GMT
            Not After : Feb  2 00:00:00 2021 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:ae:4c:db:01:f6:14:de:fc:71:21:28:5f:dc:7f:
                    5c:6d:1d:42:c9:56:47:f0:61:ba:00:80:df:67:88:
                    67:84:5e:e9:a6:9f:d4:89:31:49:da:e3:d3:b1:54:
                    16:d7:53:2c:38:71:52:b8:0b:0d:f3:e1:af:40:8a:
                    95:d3:07:1e:58
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage:
                Digital Signature
    Signature Algorithm: ecdsa-with-SHA256
         30:44:02:20:37:38:73:ef:87:81:b8:82:97:ef:23:5c:1f:ac:
         cf:62:da:4e:44:74:0d:c2:a2:e6:a3:c6:c8:82:a3:23:8d:9c:
         02:20:3a:d9:35:3b:a7:88:68:3b:06:bb:48:fe:ca:16:ea:71:
         17:17:34:c6:75:c5:33:2b:2a:f1:cb:73:38:10:a1:fc
```

The DER encoding of the above certificate is 314 bytes.

```
308201363081DEA003020102020301F50D300A06082A8648CE3D040302301631
143012060355040030C0B52464320746573742043413011E170D32303031303130
30303030305A170D323130323032303030303030305A30223120301E0603550403
0C1730312D32332D34352D46462D46452D36372D38392D41423059301306072A
8648CE3D020106082A8648CE3D03010703420004AE4CDB01F614DEFC7121285F
DC7F5C6D1D42C95647F061BA0080DF678867845EE9A69FD4893149DAE3D3B154
16D7532C387152B80B0DF3E1AF408A95D3071E58A30F300D300B0603551D0F04
0403020780300A06082A8648CE3D04030203470030440220373873EF8781B882
97EF235C1FACCF62DA4E44740DC2A2E6A3C6C882A3238D9C02203AD9353BA788
683B06BB48FECA16EA71171734C675C5332B2AF1CB733810A1FC
```

## A.2. Example CBOR Certificate Compression

The CBOR certificate compression of the X.509 in CBOR diagnostic format is:

```
(
  1,
  h'01f50d',
  0,
  "RFC test CA",
  721699200,
  760492800,
  h'0123456789AB',
  0,
  h'02ae4cdb01f614defc7121285fdc7f5c6d1d42c95647f061ba
    0080df678867845e',
  5,
  h'373873EF8781B88297EF235C1FACCF62DA4E44740DC2A2E6A3
    C6C882A3238D9C3AD9353BA788683B06BB48FECA16EA711717
    34C675C5332B2AF1CB733810A1FC'
)
```

The CBOR encoding (CBOR sequence) of the CBOR certificate is 138 bytes.

```
014301F50D006B52464320746573742043411A2B0441801A2D54330046012345
6789AB00582102AE4CDB01F614DEFC7121285FDC7F5C6D1D42C95647F061BA00
80DF678867845E055840373873EF8781B88297EF235C1FACCF62DA4E44740DC2
A2E6A3C6C882A3238D9C3AD9353BA788683B06BB48FECA16EA71171734C675C5
332B2AF1CB733810A1FC
```

## A.3. Example: Natively Signed CBOR Certificate

The corresponding natively signed CBOR certificate in CBOR diagnostic format is identical except for type and signatureValue.

```
(
  0,
  h'01f50d',
  0,
  "RFC test CA",
  721699200,
  760492800,
  h'0123456789AB',
  0,
  h'02ae4cdb01f614defc7121285fdc7f5c6d1d42c95647f061
    ba0080df678867845e',
  5,
  h'7F10A063DA8DB2FD49414440CDF85070AC22A266C7F1DFB1
    577D9A35A295A8742E794258B76968C097F85542322A0796
    0199C13CC0220A9BC729EF2ECA638CFE'
)
```

The CBOR encoding (CBOR sequence) of the CBOR certificate is 138 bytes.

004301F50D006B5246432074657374204341 1A2B0441801A2D54330046012345
6789AB00582102AE4CDB01F614DEFC7121285FDC7F5C6D1D42C95647F061BA00
80DF678867845E0558407F10A063DA8DB2FD49414440CDF85070AC22A266C7F1
DFB1577D9A35A295A8742E794258B76968C097F85542322A07960199C13CC022
0A9BC729EF2ECA638CFE

## Appendix B.  X.509 Certificate Profile, ASN.1

NOTE: The ASN.1 below is not up to date with the rest of the specification

TODO - This ASN.1 profile should probably be in a document that updates RFC 7925.

```
IOTCertificate DEFINITIONS EXPLICIT TAGS ::= BEGIN

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

```
   }

   TBSCertificate  ::= SEQUENCE {
     version           [0] INTEGER {v3(2)},
     serialNumber          INTEGER (1..MAX),
     signature             AlgorithmIdentifier,
     issuer                Name,
     validity              Validity,
```





```
     subject               Name,
     subjectPublicKeyInfo  SubjectPublicKeyInfo,
     extensions        [3] Extensions OPTIONAL
   }

   Name  ::= SEQUENCE SIZE (1) OF DistinguishedName

   DistinguishedName  ::= SET SIZE (1) OF CommonName

   CommonName  ::= SEQUENCE {
     type              OBJECT IDENTIFIER (id-at-commonName),
     value             UTF8String
   }

   Validity  ::= SEQUENCE {
     notBefore         UTCTime,
     notAfter          UTCTime
   }

   SubjectPublicKeyInfo  ::= SEQUENCE {
     algorithm         AlgorithmIdentifier,
     subjectPublicKey  BIT STRING
   }

   AlgorithmIdentifier  ::=  SEQUENCE  {
     algorithm         OBJECT IDENTIFIER,
     parameters        ANY DEFINED BY algorithm OPTIONAL  }
   }

   Extensions  ::= SEQUENCE SIZE (1..MAX) OF Extension

   Extension  ::= SEQUENCE {
     extnId            OBJECT IDENTIFIER,
```

```
     critical           BOOLEAN DEFAULT FALSE,
     extnValue          OCTET STRING
    }

   id-at-commonName    OBJECT IDENTIFIER   ::=
            {joint-iso-itu-t(2) ds(5) attributeType(4) 3}

   END
```

## Authors' Addresses


Shahid Raza
RISE AB

Email: shahid.raza@ri.se







Joel Hoeglund
RISE AB

Email: joel.hoglund@ri.se

Goeran Selander
Ericsson AB

Email: goran.selander@ericsson.com

John Preuss Mattsson
Ericsson AB

Email: john.mattsson@ericsson.com

Martin Furuhed
Nexus Group

Email: martin.furuhed@nexusgroup.com