

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2020

S. Raza
J. Hoeglund
RISE AB
G. Selander
J. Mattsson
Ericsson AB
M. Furuhed
Nexus Group
March 09, 2020

CBOR Profile of X.509 Certificates
draft-raza-ace-cbor-certificates-04

Abstract

This document specifies a CBOR encoding and profiling of X.509 public key certificate suitable for Internet of Things (IoT) deployments. The full X.509 public key certificate format and commonly used ASN.1 DER encoding is overly verbose for constrained IoT environments. Profiling together with CBOR encoding reduces the certificate size significantly with associated known performance benefits.

The CBOR certificates are compatible with the existing X.509 standard, enabling the use of profiled and compressed X.509 certificates without modifications in the existing X.509 standard.

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 September 10, 2020.

Copyright Notice

Copyright (c) 2020 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Terminology	3
3.	CBOR Encoding	4
4.	Deployment settings	6
5.	Expected Certificate Sizes	7
6.	Native CBOR Certificates	7
7.	Security Considerations	8
8.	Privacy Considerations	8
9.	IANA Considerations	8
9.1.	CBOR Certificate Types Registry	8
9.2.	CBOR Certificate Signature Algorithms Registry	9
9.3.	CBOR Certificate Public Key Algorithms Registry	9
10.	References	10
10.1.	Normative References	10
10.2.	Informative References	11
Appendix A.	Example CBOR Certificates	12
A.1.	Example X.509 Certificate	12
A.2.	Example CBOR Certificate Compression	13
A.3.	Example Native CBOR Certificate	13
Appendix B.	X.509 Certificate Profile, ASN.1	14
Authors' Addresses	15

[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 [I-D.ietf-cbor-sequence].

This document specifies the CBOR certificate profile, which is a CBOR based encoding and compression of the X.509 certificate format. The profile is based on previous work on profiling of X.509 certificates for Internet of Things deployments [RFC7925] [X.509-IoT] which retains backwards compatibility with X.509, and can be applied for lightweight certificate based authentication with e.g. TLS [RFC8446], DTLS [I-D.ietf-tls-dtls13], or EDHOC [I-D.selander-ace-cose-ecdhe]. The same profile can be used for "native" CBOR encoded certificates, which further optimizes the performance in constrained environments but are not backwards compatible with X.509, see [Section 6](#).

Other work has looked at reducing size of X.509 certificates. The purpose of this document is to stimulate a discussion on CBOR based certificates.

2. Terminology

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. The CBOR certificate can be a native CBOR certificate, in which case the signature is calculated on the CBOR encoded data, or a CBOR compressed X.509 certificates in which case the signature is calculated on the DER encoded ASN.1 data in the X.509 certificate. In both cases the certificate content is adhering to the restrictions given by [\[RFC7925\]](#). The corresponding ASN.1 schema is given in [Appendix A](#).

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 there CBOR encodings are listed below. Combining these different components reduces the certificate size significantly, something that is not possible with general purpose compressions algorithms, see Figure 1.

CBOR certificates are defined in terms of [RFC 7925](#) 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.
- 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 Distinguished Name is encoded as CBOR map, but if only CN is present the value can be encoded as a single text value.
- o validity. The 'notBefore' and 'notAfter' UTCTime fields are encoded as as UnixTime in unsigned integer format.
- o subject. The 'subject' field is restricted to specifying the value of the common name. By [RFC 7925](#) an IoT subject is identified by either an EUI-64 for clients, or by a FQDN for servers. An EUI-64 mapped from a 48-bit MAC address is encoded as a CBOR byte string of length 6. Other EUI-64 is ncoded as a CBOR byte string of length 8. A FQDN is encoded as a CBOR text string.

- o `subjectPublicKeyInfo`. If the 'algorithm' field is the default (`id-ecPublicKey` and `prime256v1`), it is omitted in the CBOR encoding., otherwise it is included in the `subjectPublicKeyInfo_algorithm` field encoded as a int, (see [Section 9](#)). The 'subjectPublicKey' is encoded as 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 [RFC 7925](#) is encoded as specified below, where a critical extensions are encoded with a negative sign.

I.e. non-critical `keyUsage` `keyAgreement` is encoded as 5, critical `basicConstraints` `cA` is encoded as -3, and non-critical `extKeyUsage` `id-kp-codeSigning` + `id-kp-OCSPSigning` is encoded as 22.

If `subjectAltName` is present, the value is placed at the end of the array encoded as a byte or text string following the encoding rules for the subject field. If the array contains a single int, extensions is encoded as the int instead of an array.

`subjectAltName` = 1

`basicConstraints` = 2 + `cA`

`keyUsage` = 3 + `digitalSignature`
+ 2 * `keyAgreement` + 4 * `keyCertSign`

`extKeyUsage` = 10 + `id-kp-serverAuth` + 2 * `id-kp-clientAuth`
+ 4 * `id-kp-codeSigning` + 8 * `id-kp-OCSPSigning`

- o `signatureAlgorithm`. If the 'signatureAlgorithm' field is the default (`ecdsa-with-SHA256`) it is omitted in the CBOR encoding, otherwise it is included in the `signatureAlgorithm` field encoded as an CBOR int (see [Section 9](#)).
- 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 native 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 native CBOR certificate (type = 0) or a CBOR compressed X.509 certificates (type = 1), see [Section 9](#).

The Concise Data Definition Language (CDDL) for CBOR certificate is:

```
certificate = (  
  type : int,  
  serialNumber : bytes,  
  issuer : { + int => bytes } / text,  
  validity_notBefore: uint,  
  validity_notAfter: uint,  
  subject : text / bytes  
  subjectPublicKey : bytes  
  extensions : [ *4 int, ? text / bytes ] / int,  
  signatureValue : bytes,  
  ? ( signatureAlgorithm : int,  
      subjectPublicKeyInfo_algorithm : int )  
)
```

The signatureValue for native CBOR certificates is calculated over the CBOR sequence:

```
(  
  type : int,  
  serialNumber : bytes,  
  issuer : { + int => bytes } / text,  
  validity_notBefore: uint,  
  validity_notAfter: uint,  
  subject : text / bytes  
  subjectPublicKey : bytes  
  extensions : [ *4 int, ? text / bytes ] / int,  
  ? ( signatureAlgorithm : int,  
      subjectPublicKeyInfo_algorithm : int )  
)
```

TODO - Specify exactly how issuer is encoded into a map / text and back again.

[4.](#) 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 the currently used DTLS v1.2 protocol, 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, 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 DTLS v1.3, because certificates are encrypted, the proposed encoding needs to be done fully end-to-end, through adding the encoding/decoding functionality to the server. This corresponds to the proposed native mode, a new certificate compression scheme. The required changes on the server side are in line with recent protocols utilizing cbor encoding for communication with resource constrained devices [[RFC8613](#)].

5. Expected Certificate Sizes

The CBOR encoding of the sample certificate given in [Appendix A](#) results in the numbers shown in Figure 1. 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 will be small, mainly corresponding to making the ASN.1 encoding 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	136

Figure 1: Comparing Sizes of Certificates (bytes)

6. Native CBOR Certificates

Further performance improvements can be achieved with the use of native CBOR certificates. In this case the signature is calculated over the CBOR encoded structure rather than the ASN.1 encoded

structure. This removes entirely the need for ASN.1 and reduces the processing in the authenticating devices.

This solution applies when the devices are only required to authenticate with a set of native CBOR certificate compatible servers, which may become a preferred approach for future deployments. The mapping between X.509 and CBOR certificates enables a migration path between the backwards compatible format and the fully optimized format. This motivates introducing a type flag to indicate if the certificate should be restored to X.509 or kept cbor encoded.

7. 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 current version of the format hardcodes the signature algorithm which does not allow for crypto agility. A COSE crypto algorithm can be specified with small overhead, and this change is proposed for a future version of the draft.

8. Privacy Considerations

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 4](#) requires unencrypted certificates.

9. IANA Considerations

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	Reference
0	Native CBOR Certificate.	[[this document]]
1	CBOR Compressed X.509 Certificate	[[this document]]

Figure 2: CBOR Certificate Types

9.2. 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	Reference
0	ecdsa-with-SHA384	[[this document]]
1	ecdsa-with-SHA512	[[this document]]
2	id-ecdsa-with-shake128	[[this document]]
3	id-ecdsa-with-shake256	[[this document]]
4	id-Ed25519	[[this document]]
5	id-Ed448	[[this document]]

Figure 3: CBOR Certificate Signature Algorithms

9.3. 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	Reference
0	id-ecPublicKey + prime384v1	[[this document]]
1	id-ecPublicKey + prime512v1	[[this document]]
2	id-X25519	[[this document]]
3	id-X448	[[this document]]
4	id-Ed25519	[[this document]]
5	id-Ed448	[[this document]]

Figure 4: CBOR Certificate Public Key Algorithms

10. References

10.1. Normative References

- [I-D.ietf-cbor-sequence]
Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", [draft-ietf-cbor-sequence-02](#) (work in progress), September 2019.
- [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>>.
- [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>>.

10.2. Informative References

- [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-dtls13-34](#) (work in progress), November 2019.
- [I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", [draft-selander-ace-cose-ecdhe-14](#) (work in progress), September 2019.
- [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>>.
- [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>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", [RFC 8613](#), DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/info/rfc8613>>.
- [SECG] "Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, ver. 2", 2009, <<https://secg.org/sec1-v2.pdf>>.
- [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>.

[Appendix A](#). Example CBOR Certificates

[A.1](#). Example X.509 Certificate

Example [RFC 7925](#) 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
14301206035504030C0B5246432074657374204341301E170D32303031303130
30303030305A170D3231303230323030303030305A30223120301E0603550403
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'128269',
  "RFC test CA",
  1577836800,
  1612224000,
  h'0123456789AB',
  h'02ae4cdb01f614defc7121285fdc7f5c6d1d42c95647f061ba
    0080df678867845e',
  5,
  h'373873EF8781B88297EF235C1FACCF62DA4E44740DC2A2E6A3
    C6C882A3238D9C3AD9353BA788683B06BB48FECA16EA711717
    34C675C5332B2AF1CB733810A1FC'
)

```

The CBOR encoding (CBOR sequence) of the CBOR certificate is 136 bytes

```

01431282696B52464320746573742043411A5E0BE1001A601896004601234567
89AB582102AE4CDB01F614DEFC7121285FDC7F5C6D1D42C95647F061BA0080DF
678867845E055840373873EF8781B88297EF235C1FACCF62DA4E44740DC2A2E6
A3C6C882A3238D9C3AD9353BA788683B06BB48FECA16EA71171734C675C5332B
2AF1CB733810A1FC

```

[A.3.](#) Example Native CBOR Certificate

The corresponding native CBOR certificate in CBOR diagnostic format is equal except for type and signatureValue


```
(
  0,
  h'128269',
  "RFC test CA",
  1577836800,
  1612224000,
  h'0123456789AB',
  h'02ae4cdb01f614defc7121285fdc7f5c6d1d42c95647f061
    ba0080df678867845e',
  5,
  h'7F10A063DA8DB2FD49414440CDF85070AC22A266C7F1DFB1
    577D9A35A295A8742E794258B76968C097F85542322A0796
    0199C13CC0220A9BC729EF2ECA638CFE'
)
```

The CBOR encoding (CBOR sequence) of the CBOR certificate is 136 bytes

```
00431282696B52464320746573742043411A5E0BE1001A601896004601234567
89AB582102AE4CDB01F614DEFC7121285FDC7F5C6D1D42C95647F061BA0080DF
678867845E0558407F10A063DA8DB2FD49414440CDF85070AC22A266C7F1DFB1
577D9A35A295A8742E794258B76968C097F85542322A07960199C13CC0220A9B
C729EF2ECA638CFE
```

[Appendix B](#). X.509 Certificate Profile, ASN.1

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