

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: December 2, 2021

K. Den Hartog, Ed.  
Mattr  
May 31, 2021

Pairing Friendly Curves representations in JOSE and COSE  
draft-denhartog-pairing-curves-jose-cose-00

Abstract

This specification defines representations enabling the Standards for Pairing Friendly Curves to be used for JSON Object Signing and Encryption (JOSE) and CBOR Object Signing and Encryption (COSE) messages.

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 December 2, 2021.

Copyright Notice

Copyright (c) 2021 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
1.1. Notational Conventions . . . . .	3
2. Barreto Naehrig (BN) Curves . . . . .	3
2.1. BN256 Curve . . . . .	3
2.1.1. Bn256G1 Curve JOSE Key Representation . . . . .	3
2.1.2. Bn256G1 Curve COSE Key Representation . . . . .	3
2.1.3. Bn256G2 Curve JOSE Key Representation . . . . .	3
2.1.4. Bn256G2 Curve COSE Key Representation . . . . .	4
2.2. BN462 Curve . . . . .	4
2.2.1. Bn462G1 Curve JOSE Key Representation . . . . .	4
2.2.2. Bn462G1 Curve COSE Key Representation . . . . .	4
2.2.3. Bn462G2 Curve JOSE Key Representation . . . . .	4
2.2.4. Bn462G2 Curve COSE Key Representation . . . . .	5
3. Barreto Lynn Scott (BLS) Curves . . . . .	5
3.1. BLS12-381 Curve . . . . .	5
3.1.1. Bls12381G1 Curve JOSE Key Representation . . . . .	5
3.1.2. Bls12381G1 Curve COSE Key Representation . . . . .	5
3.1.3. Bls12381G2 Curve JOSE Key Representation . . . . .	5
3.1.4. Bls12381G2 Curve COSE Key Representation . . . . .	6
3.2. BLS48-581 Curve . . . . .	6
3.2.1. Bls48581G1 Curve JOSE Key Representation . . . . .	6
3.2.2. Bls48581G1 Curve COSE Key Representation . . . . .	6
3.2.3. Bls48581G2 Curve JOSE Key Representation . . . . .	6
3.2.4. Bls48581G2 Curve COSE Key Representation . . . . .	7
4. IANA Considerations . . . . .	7
4.1. JSON Web Key Elliptic Curve Registrations . . . . .	7
4.2. COSE Elliptic Curve Registrations . . . . .	8
5. Security Considerations . . . . .	10
6. Privacy Considerations . . . . .	10
7. Acknowledgements . . . . .	10
8. Normative References . . . . .	11
Author's Address . . . . .	11

## 1. Introduction

This specification defines algorithm encodings and representations enabling the Standard Pairing Friendly Curves to be used for JSON Object Signing and Encryption (JOSE) [RFC7517] [RFC7517] and CBOR Object Signing and Encryption (COSE) [RFC8152] [RFC8152] messages. The elliptic curve and associated algorithm are registered in appropriate IANA JOSE and COSE registries.

### 1.1. 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 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Barreto Naehrig (BN) Curves

### 2.1. BN256 Curve

#### 2.1.1. Bn256G1 Curve JOSE Key Representation

The pairing friendly elliptic curve "BN256" which uses the largest prime-order subgroup of  $E(\text{GF}(p))$  is represented in a JSON Web Key (JWK) [RFC7517] [RFC7517] using these values: o "kty": "OKP" o "crv": "Bn256G1" plus "x" value to represent the curve point for the public key. The "x" value MUST be encoded using [SEC1] [SEC1] format and MUST be base64url encoded without padding as defined in [RFC7515] [RFC7515] Appendix C.

#### 2.1.2. Bn256G1 Curve COSE Key Representation

It is represented in a COSE\_Key [RFC8152] [RFC8152] using these values:

- o "kty" (1): "OKP" (1)
- o "crv" (-1): "Bn256G1" (TBD - requested assignment 9)

plus "x" (-2) values to represent the curve point for the key. The "x" value MUST be encoded using [SEC1] [SEC1] point compression format.

#### 2.1.3. Bn256G2 Curve JOSE Key Representation

The pairing friendly elliptic curve "BN256" which uses an r-order subgroup of  $E'(\text{GF}(p^2))$  is represented in a JSON Web Key (JWK) [RFC7517] [RFC7517] using these values:

- o "kty": "OKP"
- o "crv": "Bn256G2"

plus "x" value to represent the curve point for the public key. The "x" value MUST be encoded using [SEC1] [SEC1] and MUST be base64url encoded without padding as defined in [RFC7515] [RFC7515] Appendix C.

#### 2.1.4. Bn256G2 Curve COSE Key Representation

It is represented in a COSE\_Key [RFC8152] [RFC8152] using these values:

- o "kty" (1): "OKP" (1)
- o "crv" (-1): "Bn256G2" (TBD - requested assignment 10)

plus "x" (-2) values to represent the curve point for the key. The "x" value MUST be encoded using [SEC1] [SEC1] point compression format.

#### 2.2. BN462 Curve

##### 2.2.1. Bn462G1 Curve JOSE Key Representation

The pairing friendly elliptic curve "BN462" which uses the largest prime-order subgroup of  $E(\text{GF}(p))$  is represented in a JSON Web Key (JWK) [RFC7517] [RFC7517] using these values:

- o "kty": "OKP"
- o "crv": "Bn462G1"

plus "x" value to represent the curve point for the public key. The "x" value MUST be encoded using [SEC1] [SEC1] and MUST be base64url encoded without padding as defined in [RFC7515] [RFC7515] Appendix C.

##### 2.2.2. Bn462G1 Curve COSE Key Representation

It is represented in a COSE\_Key [RFC8152] [RFC8152] using these values:

- o "kty" (1): "OKP" (1)
- o "crv" (-1): "Bn462G1" (TBD - requested assignment 11)

plus "x" (-2) values to represent the curve point for the key.

##### 2.2.3. Bn462G2 Curve JOSE Key Representation

The pairing friendly elliptic curve "BN462" which uses an r-order subgroup of  $E'(\text{GF}(p^2))$  is represented in a JSON Web Key (JWK) [RFC7517] [RFC7517] using these values:

- o "kty": "OKP"
- o "crv": "Bn462G2"

plus "x" value to represent the curve point for the public key. The "x" value MUST be encoded using [SEC1] [SEC1] and MUST be base64url encoded without padding as defined in [RFC7515] [RFC7515] Appendix C.

#### 2.2.4. Bn462G2 Curve COSE Key Representation

It is represented in a COSE\_Key [RFC8152] [RFC8152] using these values:

- o "kty" (1): "OKP" (1)
- o "crv" (-1): "Bn462G2" (TBD - requested assignment 12)

plus "x" (-2) values to represent the curve point for the key.

### 3. Barreto Lynn Scott (BLS) Curves

#### 3.1. BLS12-381 Curve

##### 3.1.1. Bls12381G1 Curve JOSE Key Representation

The pairing friendly elliptic curve "BLS12-381" which uses the largest prime-order subgroup of  $E(\text{GF}(p))$  is represented in a JSON Web Key (JWK) [RFC7517] [RFC7517] using these values:

- o "kty": "OKP"
- o "crv": "Bls12381G1"

plus "x" value to represent the curve point for the public key. The "x" value MUST be encoded using the Z-Cash serialization defined in [DPFC09] [DPFC09] and MUST be base64url encoded without padding as defined in [RFC7515] [RFC7515] Appendix C.

##### 3.1.2. Bls12381G1 Curve COSE Key Representation

It is represented in a COSE\_Key [RFC8152] [RFC8152] using these values:

- o "kty" (1): "OKP" (1)
- o "crv" (-1): "Bls12381G1" (TBD - requested assignment 13)

plus "x" (-2) values to represent the curve point for the key.

##### 3.1.3. Bls12381G2 Curve JOSE Key Representation

The pairing friendly elliptic curve "BLS12-381" which uses an  $r$ -order subgroup of  $E'(\text{GF}(p^2))$  is represented in a JSON Web Key (JWK) [RFC7517] [RFC7517] using these values:

- o "kty": "OKP"
- o "crv": "Bls12381G2"

plus "x" value to represent the curve point for the public key. The "x" value MUST be encoded using the Z-Cash serialization defined in [DPFC09] [DPFC09] and MUST be base64url encoded without padding as defined in [RFC7515] [RFC7515] Appendix C.

#### 3.1.4. Bls12381G2 Curve COSE Key Representation

It is represented in a COSE\_Key [RFC8152] [RFC8152] using these values:

- o "kty" (1): "OKP" (1)
- o "crv" (-1): "Bls12381G2" (TBD - requested assignment 14)

plus "x" (-2) values to represent the curve point for the key.

### 3.2. BLS48-581 Curve

#### 3.2.1. Bls48581G1 Curve JOSE Key Representation

The pairing friendly elliptic curve "BLS48-581" which uses the largest prime-order subgroup of  $E(\text{GF}(p))$  is represented in a JSON Web Key (JWK) [RFC7517] [RFC7517] using these values:

- o "kty": "OKP"
- o "crv": "Bls48581G1"

plus "x" value to represent the curve point for the public key. The "x" value MUST be encoded using [SEC1] [SEC1] and MUST be base64url encoded without padding as defined in [RFC7515] [RFC7515] Appendix C.

#### 3.2.2. Bls48581G1 Curve COSE Key Representation

It is represented in a COSE\_Key [RFC8152] [RFC8152] using these values:

- o "kty" (1): "OKP" (1)
- o "crv" (-1): "Bls48581G1" (TBD - requested assignment 15)

plus "x" (-2) values to represent the curve point for the key.

#### 3.2.3. Bls48581G2 Curve JOSE Key Representation

The pairing friendly elliptic curve "BLS48-581" which uses the largest prime-order subgroup of  $E'(\text{GF}(p^8))$  is represented in a JSON Web Key (JWK) [RFC7517] [RFC7517] using these values:

- o "kty": "OKP"
- o "crv": "Bls48581G2"

plus "x" value to represent the curve point for the public key. The "x" value MUST be encoded using [SEC1] [SEC1] and MUST be base64url encoded without padding as defined in [RFC7515] [RFC7515] Appendix C.

#### 3.2.4. Bls48581G2 Curve COSE Key Representation

It is represented in a COSE\_Key [RFC8152] [RFC8152] using these values:

- o "kty" (1): "OKP" (1)
- o "crv" (-1): "Bls48581G1" (TBD - requested assignment 16)

plus "x" (-2) values to represent the curve point for the key.

### 4. IANA Considerations

#### 4.1. JSON Web Key Elliptic Curve Registrations

This section registers the following value in the IANA "JSON Web Key Elliptic Curve" registry [IANA.JOSE.Curves].

- o Curve Name: Bn256G1
- o Curve Description: 256 bit Barreto-Naehrig pairing friendly curve using the largest prime-order subgroup of  $E(\text{GF}(p))$
- o JOSE Implementation Requirements: Prohibited
- o Change Controller: IESG
- o Specification Document(s): Section 2.1 of [[ this specification ]]

- o Curve Name: Bn256G2
- o Curve Description: 256 bit Barreto-Naehrig pairing friendly curve using an r-order subgroup of  $E'(\text{GF}(p^2))$
- o JOSE Implementation Requirements: Prohibited
- o Change Controller: IESG
- o Specification Document(s): Section 2.3 of [[ this specification ]]

- o Curve Name: Bn462G1
- o Curve Description: 462 bit Barreto-Naehrig pairing friendly curve using the largest prime-order subgroup of  $E(\text{GF}(p))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 2.5 of [[ this specification ]]

- o Curve Name: Bn462G2
- o Curve Description: 462 bit Barreto-Naehrig pairing friendly curve using an r-order subgroup of  $E'(\text{GF}(p^2))$

- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 2.7 of [[ this specification ]]
- o Curve Name: Bls12381G1
- o Curve Description: 381 bit with an embedding degree of 12 Barreto-Lynn-Scott pairing friendly curve using an r-order subgroup of  $E(\text{GF}(p))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.1 of [[ this specification ]]
- o Curve Name: Bls12381G2
- o Curve Description: 381 bit with an embedding degree of 12 Barreto-Lynn-Scott pairing friendly curve using an r-order subgroup of  $E'(\text{GF}(p^2))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.3 of [[ this specification ]]
- o Curve Name: Bls48581G1
- o Curve Description: 581 bit with an embedding degree of 48 Barreto-Lynn-Scott pairing friendly curve using an r-order subgroup of  $E(\text{GF}(p))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.5 of [[ this specification ]]
- o Curve Name: Bls48581G1
- o Curve Description: 581 bit with an embedding degree of 48 Barreto-Lynn-Scott pairing friendly curve using an r-order subgroup of  $E'(\text{GF}(p^8))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.7 of [[ this specification ]]

#### 4.2. COSE Elliptic Curve Registrations

This section registers the following value in the IANA "JSON Web Key Elliptic Curve" registry [IANA.JOSE.Curves].

- o Curve Name: Bn256G1
- o Value: TBD (requested assignment 9)
- o Key Type: OKP
- o Curve Description: 256 bit Barreto-Naehrig pairing friendly curve using the largest prime-order subgroup of  $E(\text{GF}(p))$
- o JOSE Implementation Requirements: Prohibited
- o Change Controller: IESG



- o Specification Document(s): Section 2.2 of [[ this specification ]]
- o Recommended: No (Prohibited)
  
- o Curve Name: Bn256G2
- o Value: TBD (requested assignment 10)
- o Key Type: OKP
- o Curve Description: 256 bit Barreto-Naehrig pairing friendly curve using an r-order subgroup of  $E'(\text{GF}(p^2))$
- o JOSE Implementation Requirements: Prohibited
- o Change Controller: IESG
- o Specification Document(s): Section 2.4 of [[ this specification ]]
- o Recommended: No (Prohibited)
  
- o Curve Name: Bn462G1
- o Value: TBD (requested assignment 11)
- o Key Type: OKP
- o Curve Description: 462 bit Barreto-Naehrig pairing friendly curve using the largest prime-order subgroup of  $E(\text{GF}(p))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 2.6 of [[ this specification ]]
- o Recommended: No
  
- o Curve Name: Bn462G2
- o Value: TBD (requested assignment 12)
- o Key Type: OKP
- o Curve Description: 462 bit Barreto-Naehrig pairing friendly curve using an r-order subgroup of  $E'(\text{GF}(p^2))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 2.8 of [[ this specification ]]
- o Recommended: No
  
- o Curve Name: Bls12381G1
- o Value: TBD (requested assignment 13)
- o Key Type: OKP
- o Curve Description: 381 bit with an embedding degree of 12 Barreto-Lynn-Scott pairing friendly curve using an r-order subgroup of  $E(\text{GF}(p))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.2 of [[ this specification ]]
- o Recommended: Yes
  
- o Curve Name: Bls12381G2
- o Value: TBD (requested assignment 14)
- o Key Type: OKP

- o Curve Description: 381 bit with an embedding degree of 12 Barreto-Lynn-Scott pairing friendly curve using an r-order subgroup of  $E'(\text{GF}(p^2))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.4 of [[ this specification ]]
- o Recommended: Yes
  
- o Curve Name: Bls48581G1
- o Value: TBD (requested assignment 15)
- o Key Type: OKP
- o Curve Description: 581 bit with an embedding degree of 48 Barreto-Lynn-Scott pairing friendly curve using an r-order subgroup of  $E(\text{GF}(p))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.6 of [[ this specification ]]
- o Recommended: No
  
- o Curve Name: Bls48581G1
- o Value: TBD (requested assignment 16)
- o Key Type: OKP
- o Curve Description: 581 bit with an embedding degree of 48 Barreto-Lynn-Scott pairing friendly curve using an r-order subgroup of  $E'(\text{GF}(p^8))$
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.8 of [[ this specification ]]
- o Recommended: No

## 5. Security Considerations

See [DPFC09] [DPFC09] for additional details about security considerations of the curves used. Implementers should also consider section 9 of [RFC7517] [RFC7517] when implementing this work.

## 6. Privacy Considerations

To be added.

## 7. Acknowledgements

The authors of this draft would like to acknowledge the following individuals for the significant contribution of ideas and time spent reviewing this document:

## 8. Normative References

- [DPFC09] IRTF CFRG, "Pairing-Friendly Curves", November 2020, <<https://tools.ietf.org/html/draft-irtf-cfrg-pairing-friendly-curves-09>>.
- [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>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/info/rfc7517>>.
- [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>>.
- [SEC1] Standards for Efficient Cryptography Group (SECG), "SEC 1: Elliptic Curve Cryptography", 2009, <<https://www.secg.org/sec1-v2.pdf>>.

## Author's Address

Kyle Den Hartog (editor)  
Mattr

Email: [kyle.denhartog@mattr.global](mailto:kyle.denhartog@mattr.global)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 14 July 2022

J. Preuß Mattsson  
G. Selander  
Ericsson AB  
S. Raza  
J. Höglund  
RISE AB  
M. Furuhed  
Nexus Group  
10 January 2022

CBOR Encoded X.509 Certificates (C509 Certificates)  
draft-ietf-cose-cbor-encoded-cert-03

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%. 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 COSE headers, a C509 TLS certificate type, and a C509 file format.

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 14 July 2022.

## Copyright Notice

Copyright (c) 2022 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 . . . . .	4
3. C509 Certificate . . . . .	5
3.1. Message Fields . . . . .	5
3.2. Encoding of subjectPublicKey and issuerSignatureValue . .	9
3.3. Encoding of Extensions . . . . .	9
4. C509 Certificate Signing Request . . . . .	14
5. C509 Certificate Revocation List . . . . .	15
6. C509 Online Certificate Status Protocol . . . . .	16
7. C509 Processing and Certificate Issuance . . . . .	16
8. Legacy Considerations . . . . .	17
9. Expected Certificate Sizes . . . . .	17
10. Security Considerations . . . . .	18
11. IANA Considerations . . . . .	19
11.1. C509 Certificate Types Registry . . . . .	19
11.2. C509 Attributes Registry . . . . .	20
11.3. C509 Extensions Registry . . . . .	23
11.4. C509 Certificate Policies Registry . . . . .	27
11.5. C509 Policies Qualifiers Registry . . . . .	29
11.6. C509 Information Access Registry . . . . .	30
11.7. C509 Extended Key Usages Registry . . . . .	32
11.8. C509 General Names Registry . . . . .	33
11.9. C509 Signature Algorithms Registry . . . . .	35
11.10. C509 Public Key Algorithms Registry . . . . .	38
11.11. COSE Header Parameters Registry . . . . .	41
11.12. TLS Certificate Types Registry . . . . .	42
11.13. CBOR Tags Registry . . . . .	43
12. References . . . . .	43
12.1. Normative References . . . . .	43
12.2. Informative References . . . . .	44
Appendix A. Example C509 Certificates . . . . .	47
A.1. Example RFC 7925 profiled X.509 Certificate . . . . .	47

A.2. Example IEEE 802.1AR profiled X.509 Certificate . . . . .	50
A.3. Example CAB Baseline ECDSA HTTPS X.509 Certificate . . . . .	50
A.4. Example CAB Baseline RSA HTTPS X.509 Certificate . . . . .	53
Acknowledgments . . . . .	56
Authors' Addresses . . . . .	56

## 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 [I-D.ietf-emu-eap-tls13] [I-D.ietf-emu-eaptls-cert] 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.

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 [I-D.ietf-tls-dtls13], COSE [RFC8152], 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. 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 specifies COSE headers for use of the C509 certificates with COSE, see Section 11.11. The document also specifies a TLS certificate type for use of the C509 certificates with TLS and QUIC (with or without additional TLS certificate compression), see Section 11.12.

## 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 [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, 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 5.

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 11.1.



- \* `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 a (CBOR int, CBOR text string) pair or as a (unwrapped CBOR OID, 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 have 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 a CBOR text string. 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 of length 8 instead. 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 of length 6.
- \* `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 11.10) 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 11.3) followed by an optional CBOR item of any type or an unwrapped CBOR OID tag [RFC9090] followed by a 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 11.9) 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 )

Time = ~time / null

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

Extensions = [ * Extension ] / int

Extension = ( extensionID: int, extensionValue: any ) //
            ( extensionID: ~oid, ? 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), uncompressed keys are point compressed as defined in Section 2.3.3 of [SECG]. If a DER encoded certificate with a point compressed public key of type id-ecPublicKey is CBOR encoded, 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 shortest 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 as either a CBOR int or an unwrapped CBOR OID tag. If 'extensionID' is encoded an int (see Section 11.3), 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.

CBOR encoding of the following extension values are 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 11.7 or unwrapped CBOR OID tags [RFC9090] where each int or OID tag encodes a key usage purpose. If the array contains a single KeyPurposeId, the array is omitted.

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

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

```
InhibitAnyPolicy = uint
```

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 11.8 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 11.8). 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 identify the pair ( hwType, hwSerialEntries ) directly as specified in [RFC4108]. Only the general names in Section 11.8 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 an CBOR ints (see Section 11.6) or unwrapped CBOR OID tags [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,  
  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 11.4 are encoded as an int. The policyQualifierId is encoded as an CBOR int (see Section 11.5) 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 contains general names registered in Section 11.8 the extension value can be CBOR encoded.

```
GeneralSubtree = [ GeneralName, minimum: uint, ? maximum: uint ]  
NameConstraints = [  
  permittedSubtrees: GeneralSubtree,  
  excludedSubtrees: GeneralSubtree,  
]
```

- \* 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 (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-ipAddrBlocks-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 ASId, the Address is encoded as an uint with the the difference to the previous Address.

```
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 and 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 11.3, Section 11.7, and Section 11.8:

- \* A critical basicConstraints ('cA' = true) without pathLenConstraint is encoded as the two CBOR ints -4, -1.
- \* A non-critical keyUsage with digitalSignature and keyAgreement asserted is encoded as the two CBOR ints 2, 17 ( $2^0 + 2^4 = 17$ ).
- \* 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, 17, 8, [ 3, 6 ], 3, "example.com" ].

## 4. C509 Certificate Signing Request

The section defines the C509 Certificate Signing Request (CSR) format based on and compatible with RFC 2986 [RFC2986] reusing the formatting for C509 certificates defined in Section 3. There are currently two c509CertificateSigningRequestType values defined, c509CertificateSigningRequestType = 0 requests a c509CertificateType = 0 and c509CertificateSigningRequestType = 1 requests a c509CertificateType = 1. subjectProofOfPossessionAlgorithm can be a C509 signature algorithm or a non-signature Proof-of-Possession Algorithm as defined in e.g. RFC 6955. CSR attributes other than extensionRequest are not supported.

```
C509CertificateSigningRequest = [  
    TBSCertificateSigningRequest,  
    subjectProofOfPossessionValue: any,  
]  
  
; The elements of the following group are used in a CBOR Sequence:  
TBSCertificateSigningRequest = (  
    c509CertificateSigningRequestType: int,  
    subject: Name,  
    subjectPublicKeyAlgorithm: AlgorithmIdentifier,  
    subjectPublicKey: any,  
    extensionsRequest : Extensions,  
    subjectProofOfPossessionAlgorithm: AlgorithmIdentifier,  
)
```

Figure 2: CDDL for C509CertificateSigningRequest.

After verifying the `subjectProofOfPossessionValue`, the CA MAY transform the `C509CertificateSigningRequest` into a RFC 2985 `CertificationRequestInfo` for compatibility with existing procedures and code.

## 5. C509 Certificate Revocation List

The section defines the C509 Certificate Revocation List (CRL) format based on and compatible with [RFC5280] reusing the formatting for C509 certificates defined in Section 3.

```
C509CertificateRevocationList = [  
  TBSCertificateRevocationList,  
  issuerSignatureValue : any,  
]  
  
; The elements of the following group are used in a CBOR Sequence:  
TBSCertificateSigningRequest = (  
  C509CertificateRevocationListType: int,  
  issuer: Name,  
  thisUpdate: Time,  
  nextUpdate: Time,  
  revokedCertificates: RevokedCertificates,  
  crlExtensions: Extensions,  
  issuerSignatureAlgorithm: AlgorithmIdentifier,  
)  
  
RevokedCertificates = [  
  userCertificate: CertificateSerialNumber,  
  revocationDate: Time,  
  crlEntryExtensions: Extensions,  
]
```

Figure 3: CDDL for C509CertificateRevocationList.

## 6. C509 Online Certificate Status Protocol

TODO

## 7. 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 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 of 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 CA may be needed.

## 8. Legacy Considerations

C509 certificates can be deployed with legacy X.509 certificates and CA infrastructure. In order to verify the signature, the C509 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 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.

## 9. Expected Certificate Sizes

The CBOR encoding of the sample certificate chains given in Appendix A results in the numbers shown in Figure 4 and Figure 5. COSE\_X509 is defined in [I-D.ietf-cose-x509] and COSE\_C509 is defined in Section 11.11. 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) 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 4: Comparing Sizes of Certificate Chains in COSE. Number of bytes (length of certificate chain).

	X509	X509 + 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 5: Comparing Sizes of Certificate Chains with TLS. Number of bytes (length of certificate chain). X509 and C509 are Certificate messages. X509 + Brotli and C509 + Brotli are CompressedCertificate messages.

## 10. 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 8 requires unencrypted certificates and is not recommended.

## 11. IANA Considerations

This document creates several new registries under the new heading "C509 Certificate". 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.

### 11.1. C509 Certificate Types Registry

IANA has created a new registry titled "C509 Certificate Types" under the new heading "C509 Certificate". 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 6: C509 Certificate Types

## 11.2. C509 Attributes Registry

IANA has created a new registry titled "C509 Attributes" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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". 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 Comments:

	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:	
21	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:	

Figure 7: C509 Attributes

### 11.3. C509 Extensions Registry

IANA has created a new registry titled "C509 Extensions Registry" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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. For values in the interval [1, 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	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: SignedCertificateTimestamps
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: 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: 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: 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: 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

Figure 8: C509 Extensions

## 11.4. C509 Certificate Policies Registry

IANA has created a new registry titled "C509 Certificate Policies Registry" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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	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

#### 11.5. C509 Policies Qualifiers Registry

IANA has created a new registry titled "C509 Policies Qualifiers Registry" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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	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



#### 11.6. C509 Information Access Registry

IANA has created a new registry titled "C509 Information Access Registry" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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

## 11.7. C509 Extended Key Usages Registry

IANA has created a new registry titled "C509 Extended Key Usages Registry" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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	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-OCSPSigning OID: 1.3.6.1.5.5.7.3.9 DER: 06 08 2B 06 01 05 05 07 03 09 Comments: RFC 5280
10	Name: Kerberos PKINIT Client Auth Identifiers: id-pkinit-KPClientAuth OID: 1.3.6.1.5.2.3.4 DER: 06 07 2B 06 01 05 02 03 04 Comments: RFC 4556
11	Name: Kerberos PKINIT KDC Identifiers: id-pkinit-KPKdc OID: 1.3.6.1.5.2.3.5 DER: 06 07 2B 06 01 05 02 03 05 Comments: RFC 4556
12	Name: SSH Client Identifiers: id-kp-secureShellClient OID: 1.3.6.1.5.5.7.3.21 DER: 06 08 2B 06 01 05 05 07 03 15 Comments: RFC 6187
13	Name: Kerberos PKINIT KDC Identifiers: id-pkinit-KPKdc OID: 1.3.6.1.5.5.7.3.22 DER: 06 08 2B 06 01 05 05 07 03 16 Comments: RFC 6187

Figure 12: C509 Extended Key Usages

### 11.8. C509 General Names Registry

IANA has created a new registry titled "C509 General Names Registry" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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
-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

## 11.9. C509 Signature Algorithms Registry

IANA has created a new registry titled "C509 Signature Algorithms" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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:
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 <sup>MT</sup> 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:

Figure 14: C509 Signature Algorithms

## 11.10. C509 Public Key Algorithms Registry

IANA has created a new registry titled "C509 Public Key Algorithms" under the new heading "CBOR Encoded X509 Certificates (C509 Certificates)". 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 <sup>MT</sup> 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 13 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 13 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 13 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 13 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

Figure 15: C509 Public Key Algorithms

#### 11.11. COSE Header Parameters Registry

EDITORS NOTE: The text should be moved a section and not be in the IANA Section.

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 for c5b, c5c, and c5t, and c5u are similar to x5bag, x5chain, x5t, x5u defined in [I-D.ietf-cose-x509] 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/cbor content format. The COSE\_C509 structure used in c5b, c5c, and c5u is defined as:

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

As the contents of c5bag, c5chain, 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.

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

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

#### 11.12. 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 X509, 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. When COSE WG has adopted work on C509 certificates, it could perhaps be presented in the TLS WG. The TLS WG might e.g. want a separate draft in the TLS WG.

Value	Name	Recommended	Comment
TBD5	C509 Certificate	Y	

### 11.13. 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

## 12. References

### 12.1. Normative References

- [I-D.ietf-cose-x509]  
Schaad, J., "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates", Work in Progress, Internet-Draft, draft-ietf-cose-x509-08, 14 December 2020, <<https://www.ietf.org/internet-drafts/draft-ietf-cose-x509-08.txt>>.
- [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>>.
- [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>>.

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

## 12.2. 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/>>.
- [ChorMe] Bormann, C., "CBOR Playground", May 2018, <<http://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-emu-eap-tls13]

Mattsson, J. and M. Sethi, "Using EAP-TLS with TLS 1.3 (EAP-TLS 1.3)", Work in Progress, Internet-Draft, draft-ietf-emu-eap-tls13-21, 20 October 2021, <<https://www.ietf.org/internet-drafts/draft-ietf-emu-eap-tls13-21.txt>>.

## [I-D.ietf-emu-eaptls-cert]

Sethi, M., Mattsson, J., and S. Turner, "Handling Large Certificates and Long Certificate Chains in TLS-based EAP Methods", Work in Progress, Internet-Draft, draft-ietf-emu-eaptls-cert-08, 20 November 2020, <<https://www.ietf.org/archive/id/draft-ietf-emu-eaptls-cert-08.txt>>.

## [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-12, 20 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-lake-edhoc-12.txt>>.

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

Rescorla, E., Barnes, R., and H. Tschofenig, "Compact TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-ctls-04, 25 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-ctls-04.txt>>.

## [I-D.ietf-tls-dtls13]

Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <<https://www.ietf.org/internet-drafts/draft-ietf-tls-dtls13-43.txt>>.

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

Tschofenig, H. and T. Fossati, "TLS/DTLS 1.3 Profiles for the Internet of Things", Work in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-profile-03, 25 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-uta-tls13-iot-profile-03.txt>>.



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

[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 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: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  
30:44:02:20:44:5d:79:8c:90:e7:f5:00:dc:74:7a:65:4c:ec:  
6c:fa:6f:03:72:76:e1:4e:52:ed:07:fc:16:29:4c:84:66:0d:  
02:20:5a:33:98:5d:fb:d4:bf:dd:6d:4a:cf:38:04:c3:d4:6e:  
bf:3b:7f:a6:26:40:67:4f:c0:35:4f:a0:56:db:ae:a6

The DER encoding of the above certificate is 314 bytes.

```

30 82 01 36 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 30 30 31 30 31 30 30 30 30 30
5A 17 0D 32 31 30 32 30 32 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 47 00 30 44 02 20 44 5D 79 8C 90
E7 F5 00 DC 74 7A 65 4C EC 6C FA 6F 03 72 76 E1 4E 52 ED 07 FC 16 29
4C 84 66 0D 02 20 5A 33 98 5D FB D4 BF DD 6D 4A CF 38 04 C3 D4 6E BF
3B 7F A6 26 40 67 4F C0 35 4F A0 56 DB AE A6

```

#### 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,
h'01f50d',
"RFC test CA",
1577836800,
1612224000,
h'0123456789AB',
1,
h'02B1216AB96E5B3B3340F5BDF02E693F16213A04525ED44450
B1019C2DFD3838AB',
1,
0,
h'445D798C90E7F500DC747A654CEC6CFA6F037276E14E52ED07
FC16294C84660D5A33985DFBD4BFDD6D4ACF3804C3D46EBF3B
7FA62640674FC0354FA056DBAEA6'

```

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

```

01
43 01 F5 0D
6B 52 46 43 20 74 65 73 74 20 43 41
1A 5E 0B E1 00
1A 60 18 96 00
46 01 23 45 67 89 AB
01
58 21 02 B1 21 6A B9 6E 5B 3B 33 40 F5 BD F0 2E 69 3F 16 21 3A 04 52
5E D4 44 50 B1 01 9C 2D FD 38 38 AB
01
00
58 40 44 5D 79 8C 90 E7 F5 00 DC 74 7A 65 4C EC 6C FA 6F 03 72 76 E1
4E 52 ED 07 FC 16 29 4C 84 66 0D 5A 33 98 5D FB D4 BF DD 6D 4A CF 38
04 C3 D4 6E BF 3B 7F A6 26 40 67 4F C0 35 4F A0 56 DB AE A6

```

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

The corresponding natively signed C509 certificate in CBOR diagnostic format is identical, except for `c509CertificateType` and `signatureValue`.

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

```

0,
h'01f50d',
"RFC test CA",
1577836800,
1612224000,
h'0123456789AB',
1,
h'02B1216AB96E5B3B3340F5BDF02E693F16213A04525ED44450
B1019C2DFD3838AB',
1,
0,
h'B27A0B781455F71B68290F6C2EC9A897F18FDE9B6C59575953
BC67268AB0E4DDE99D273E04E4715383AB2257C6AAA35284E5
ED18BDB91247E9F2C433136480B9'

```

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

```

00
43 01 F5 0D
6B 52 46 43 20 74 65 73 74 20 43 41
1A 5E 0B E1 00
1A 60 18 96 00
46 01 23 45 67 89 AB
01
58 21 02 B1 21 6A B9 6E 5B 3B 33 40 F5 BD F0 2E 69 3F 16 21 3A 04 52
5E D4 44 50 B1 01 9C 2D FD 38 38 AB
01
00
58 40 B2 7A 0B 78 14 55 F7 1B 68 29 0F 6C 2E C9 A8 97 F1 8F DE 9B 6C
59 57 59 53 BC 67 26 8A B0 E4 DD E9 9D 27 3E 04 E4 71 53 83 AB 22 57
C6 AA A3 52 84 E5 ED 18 BD B9 12 47 E9 F2 C4 33 13 64 80 B9

```

#### A.1.3. 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 [RFC8152]. 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

EDITOR'S NOTE: To do

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

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

30 82 04 b5 30 82 04 5a a0 03 02 01 02 02 10 04 7f a1 e3 19 28 ee 40  
3b a0 b8 3a 39 56 73 fc 30 0a 06 08 2a 86 48 ce 3d 04 03 02 30 4a 31  
0b 30 09 06 03 55 04 06 13 02 55 53 31 19 30 17 06 03 55 04 0a 13 10  
43 6c 6f 75 64 66 6c 61 72 65 2c 20 49 6e 63 2e 31 20 30 1e 06 03 55  
04 03 13 17 43 6c 6f 75 64 66 6c 61 72 65 20 49 6e 63 20 45 43 43 20  
43 41 2d 33 30 1e 17 0d 32 30 30 37 32 39 30 30 30 30 30 30 5a 17 0d  
32 31 30 37 32 39 31 32 30 30 30 30 5a 30 6d 31 0b 30 09 06 03 55 04  
06 13 02 55 53 31 0b 30 09 06 03 55 04 08 13 02 43 41 31 16 30 14 06  
03 55 04 07 13 0d 53 61 6e 20 46 72 61 6e 63 69 73 63 6f 31 19 30 17  
06 03 55 04 0a 13 10 43 6c 6f 75 64 66 6c 61 72 65 2c 20 49 6e 63 2e  
31 1e 30 1c 06 03 55 04 03 13 15 73 6e 69 2e 63 6c 6f 75 64 66 6c 61  
72 65 73 7c 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'
]
l,
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
  FC91EAE02168866054EE18A2E5346C4CC51FEB3FA10A91D2EDBF99125F86CE6'
]
],
23,
h' 14043FA0BED2EE3FA86E3A1F788EA04C35530F11061FFF60A16D0B83E9D92ADB
  B33F9DB3D7E0594C19A8E419A50CA770727763D5FE64510AD27AD650A58A9238
  ECCB2F0F5AC064584D5C06B9736368278B8934DC79C71D3AFD345F8314415849
  80682980398A867269CC7937CEE397F7DCF39588ED81032900D2A2C7BAABD63A
  8ECA090BD9FB39264BFF03D88E2D3F6B21CA8A7DD85FFB94BA83DE9CFC158D61
  FA672DB0C7DB3D250A414A85D37F4946373CF4B175D052F3DDC766F14BFDAA00
  EDBFE47EED01EC7BE4F646FC31FD72FE03D2F265AF4D7EE2819B7AFD303CF552
  F40534A08A3E194158C8A8E05171840915AECA57775FA18F7D577D531CCC72D'

```

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, Ilari Liusvaara, Laurence Lundblade, Francesca Palombinini, Thomas Peterson, Michael Richardson, Maik Reichert, Stefan Santesson, Jim Schaad, 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

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 16 June 2021

J. Schaad  
August Cellars  
13 December 2020

CBOR Object Signing and Encryption (COSE): Header parameters for  
carrying and referencing X.509 certificates  
draft-ietf-cose-x509-08

## Abstract

The CBOR Signing And Encrypted Message (COSE) structure uses references to keys in general. For some algorithms, additional properties are defined which carry parameters relating to keys as needed. The COSE Key structure is used for transporting keys outside of COSE messages. This document extends the way that keys can be identified and transported by providing attributes that refer to or contain X.509 certificates.

## Contributing to this document

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

The source for this draft is being maintained in GitHub. Suggested changes should be submitted as pull requests at <https://github.com/cose-wg/X509>. Instructions are on that page as well. Editorial changes can be managed in GitHub, but any substantial issues need to be discussed on the COSE mailing list.

## 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 16 June 2021.

## 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
1.1. Requirements Terminology . . . . .	3
2. X.509 COSE Header Parameters . . . . .	3
3. X.509 certificates and static-static ECDH . . . . .	7
4. IANA Considerations . . . . .	8
4.1. COSE Header Parameter Registry . . . . .	8
4.2. COSE Header Algorithm Parameter Registry . . . . .	8
5. Security Considerations . . . . .	9
6. References . . . . .	9
6.1. Normative References . . . . .	9
6.2. Informative References . . . . .	10
Author's Address . . . . .	11

## 1. Introduction

In the process of writing [RFC8152], the working group discussed X.509 certificates [RFC5280] and decided that no use cases were presented that showed a need to support certificates. Since that time, a number of cases have been defined in which X.509 certificate support is necessary, and by implication, applications will need a documented and consistent way to handle such certificates. This document defines a set of attributes that will allow applications to transport and refer to X.509 certificates in a consistent manner.

In some of these cases, a constrained device is being deployed in the context of an existing X.509 PKI: for example, in the 6TiSCH environment, [I-D.richardson-enrollment-roadmap] describes a device enrollment solution that relies on the presence of a factory-installed certificate on the device. The [I-D.ietf-lake-edhoc] draft was also written with the idea that long term certificates could be used to provide for authentication of devices, and uses them to establish session keys. Another possible scenario is the use of COSE

as the basis for a secure messaging application. This scenario assumes the presence of long term keys and a central authentication authority. Basing such an application on public key certificates allows it to make use of well established key management disciplines.

### 1.1. Requirements 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.

## 2. X.509 COSE Header Parameters

The use of X.509 certificates allows for an existing trust infrastructure to be used with COSE. This includes the full suite of enrollment protocols, trust anchors, trust chaining and revocation checking that have been defined over time by the IETF and other organizations. The key structures that have been defined in COSE currently do not support all of these properties although some may be found in COSE Web Tokens (CWT) [RFC8392].

It is not necessarily expected that constrained devices themselves will evaluate and process X.509 certificates: it is perfectly reasonable for a constrained device to be provisioned with a certificate that it subsequently provides to a relying party - along with a signature or encrypted message - on the assumption that the relying party is not a constrained device, and is capable of performing the required certificate evaluation and processing. It is also reasonable that a constrained device would have the hash of a certificate associated with a public key and be configured to use a public key for that thumbprint, but without performing the certificate evaluation or even having the entire certificate. In any case, there still needs to be an entity that is responsible for handling the possible certificate revocation.

Parties that intend to rely on the assertions made by a certificate obtained from any of these methods still need to validate it. This validation can be done according to the PKIX rules in [RFC5280] or by using a different trust structure, such as a trusted certificate distributor for self-signed certificates. The PKIX validation includes matching against the trust anchors configured for the application. These rules apply when the validation succeeds in a single step as well as when certificate chains need to be built. If the application cannot establish trust in the certificate, the public key contained in the certificate cannot be used for cryptographic operations.

The header parameters defined in this document are:

**x5bag:** This header parameter contains a bag of X.509 certificates. The set of certificates in this header parameter is unordered and may contain self-signed certificates. Note that there could be duplicating certificates. The certificate bag can contain certificates which are completely extraneous to the message. (An example of this would be where a signed message is being used to transport a certificate containing a key agreement key.) As the certificates are unordered, the party evaluating the signature will need to be capable of building the certificate path as necessary. That party will also have to take into account that the bag may not contain the full set of certificates needed to build any particular chain.

The trust mechanism MUST process any certificates in this parameter 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. As the contents of this header parameter are untrusted input, the header parameter can be in either the protected or unprotected header bucket.

This header parameter allows for a single X.509 certificate or a bag of X.509 certificates to be carried in the message.

- \* If a single certificate is conveyed, it is placed in a CBOR byte string.
- \* If multiple certificates are conveyed, a CBOR array of byte strings is used, with each certificate being in its own byte string.

**x5chain:** This header parameter contains an ordered array of X.509 certificates. The certificates are to be ordered starting with the certificate containing the end-entity key followed by the certificate which signed it and so on. There is no requirement for the entire chain to be present in the element if there is reason to believe that the relying party already has, or can locate the missing certificates. This means that the relying party is still required to do path building, but that a candidate path is proposed in this header parameter.



The trust mechanism MUST process any certificates in this parameter 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. As the contents of this header parameter are untrusted input, the header parameter can be in either the protected or unprotected header bucket.

This header parameter allows for a single X.509 certificate or a chain of X.509 certificates to be carried in the message.

- \* If a single certificate is conveyed, it is placed in a CBOR byte string.
- \* If multiple certificates are conveyed, a CBOR array of byte strings is used, with each certificate being in its own byte string.

x5t: This header parameter provides the ability to identify an X.509 certificate by a hash value (a thumbprint). The 'x5t' header parameter can be represented as an array of two elements. The first element is an algorithm identifier which is an integer or a string containing the hash algorithm identifier corresponding to either the Value (integer) or Name (string) column of the algorithm registered in the "COSE Algorithms" registry. The second element is a binary string containing the hash value computed over the DER encoded certificate.

As this header parameter does not provide any trust, the header parameter can be in either a protected or unprotected header bucket.

For interoperability, applications which use this header parameter MUST support the hash algorithm 'SHA-256', but can use other hash algorithms. This requirement allows for different implementations to be configured to use an interoperable algorithm, but does not preclude the use (by prior agreement) of other algorithms.

RFC Editor please remove the following two paragraphs:

During AD review, a question was raised about how effective the previous statement is in terms of dealing with a MTI algorithm. There needs to be some type of arrangement between the parties to agree that a specific hash algorithm is going to be used in computing the thumbprint. Making it a MUST use would make that true, but it then means that agility is going to be very difficult.

The worry is that while SHA-256 may be mandatory, if a sender supports SHA-256 but only sends SHA-512 then the recipient which only does SHA-256 would not be able to use the thumbprint. In that case both applications would conform to the specification, but still not be able to inter-operate.

x5u: This header parameter provides the ability to identify an X.509 certificate by a URI [RFC3986]. It contains a CBOR text string. The referenced resource can be any of the following media types:

- \* application/pkix-cert [RFC2585]
- \* application/pkcs7-mime; smime-type="certs-only" [RFC8551]

As this header parameter implies a trust relationship between the party generating the x5u parameter and the party hosting the referred-to resource, this header parameter MUST be in the protected attribute bucket.

The URI provided MUST provide integrity protection and server authentication. For example, an HTTP or CoAP GET request to retrieve a certificate MUST use TLS [RFC8446] or DTLS [I-D.ietf-tls-dtls13]. If the retrieved certificate does not chain to an existing trust anchor, the certificate MUST NOT be trusted unless the server is configured as trusted to provide new trust anchors or if an out-of-band confirmation can be received for trusting the retrieved certificate.

The header parameters are used in the following locations:

- \* COSE\_Signature and COSE\_Sign1 objects: in these objects they identify the certificate to be used for validating the signature.
- \* COSE\_recipient objects: in this location they identify the certificate for the recipient of the message.

The labels assigned to each header parameter can be found in the following table.

Name	Label	Value Type	Description
x5bag	TBD4	COSE_X509	An unordered bag of X.509 certificates
x5chain	TBD3	COSE_X509	An ordered chain of X.509 certificates
x5t	TBD1	COSE_CertHash	Hash of an X.509 certificate
x5u	TBD2	uri	URI pointing to an X.509 certificate

Table 1: X.509 COSE Header Parameters

Below is an equivalent CDDL [RFC8610] description of the text above.

```
COSE_X509 = bstr / [ 2*certs: bstr ]
COSE_CertHash = [ hashAlg: (int / tstr), hashValue: bstr ]
```

The content of the bstr are the bytes of a DER encoded certificate.

### 3. X.509 certificates and static-static ECDH

The header parameters defined in the previous section are used to identify the recipient certificates for the ECDH key agreement algorithms. In this section we define the algorithm specific parameters that are used for identifying or transporting the sender's key for static-static key agreement algorithms.

These attributes are defined analogously to those in the previous section. There is no definition for the certificate bag, as the same attribute would be used for both the sender and recipient certificates.

**x5chain-sender:** This header parameter contains the chain of certificates starting with the sender's key exchange certificate. The structure is the same as 'x5chain'.

**x5t-sender:** This header parameter contains the hash value for the sender's key exchange certificate. The structure is the same as 'x5t'.

**x5u-sender:** This header parameter contains a URI for the sender's

key exchange certificate. The structure and processing are the same as 'x5u'.

Name	Label	Type	Algorithm	Description
x5t-sender	TBD	COSE_CertHash	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Thumbprint for the senders X.509 certificate
x5u-sender	TBD	uri	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	URI for the senders X.509 certificate
x5chain-sender	TBD	COSE_X509	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	static key X.509 certificate chain

Table 2: Static ECDH Algorithm Values

#### 4. IANA Considerations

##### 4.1. COSE Header Parameter Registry

IANA is requested to register the new COSE Header parameters in Table 1 in the "COSE Header Parameters" registry. The "Value Registry" field is empty for all of the items. For each item, the 'Reference' field points to this document.

##### 4.2. COSE Header Algorithm Parameter Registry

IANA is requested to register the new COSE Header Algorithm parameters in Table 2 in the "COSE Header Algorithm Parameters" registry. For each item, the 'Reference' field points to this document.

## 5. Security Considerations

Establishing trust in a certificate is a vital part of processing. A major component of establishing trust is determining what the set of trust anchors are for the process. A new self-signed certificate appearing on the client cannot be a trigger to modify the set of trust anchors, because a well defined trust-establishment process is required. One common way for a new trust anchor to be added (or removed) from a device is by doing a new firmware upgrade.

In constrained systems, there is a trade-off between the order of checking the signature and checking the certificate for validity. Validating certificates can require that network resources be accessed in order to get revocation information or retrieve certificates during path building. The resulting network access can consume power and network bandwidth. On the other hand, if the certificates are validated after the signature is validated, an oracle can potentially be built based on detecting the network resources which is only done if the signature validation passes. In any event, both the signature and certificate validation **MUST** be completed successfully before acting on any requests.

Before using the key in a certificate, the key **MUST** be checked against the algorithm to be used and any algorithm specific checks need to be made. These checks can include validating that points are on curves for elliptical curve algorithms, and that sizes of RSA keys are of an acceptable size. The use of unvalidated keys can lead either to loss of security or excessive consumption of resources (for example using a 200K RSA key).

When processing x5u header parameter the security considerations of [RFC3986] and specifically those defined in Section 7.1 also apply.

Regardless of the source, certification path validation is an important part of establishing trust in a certificate. Section 6 of [RFC5280] provides guidance for the path validation. The security considerations of [RFC5280] are also important for the correct usage of this document.

The security of the algorithm used for 'x5t' does not affect the security of the system as this header parameter selects which certificate that is already present on the system should be used, but it does not provide any trust.

## 6. References

### 6.1. 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>>.
- [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>>.
- [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>>.

## 6.2. Informative References

- [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>>.
- [I-D.ietf-tls-dtls13]  
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-39, 2 November 2020, <<https://tools.ietf.org/html/draft-ietf-tls-dtls13-39>>.
- [RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <<https://www.rfc-editor.org/info/rfc8551>>.
- [RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, DOI 10.17487/RFC2585, May 1999, <<https://www.rfc-editor.org/info/rfc2585>>.
- [I-D.ietf-lake-edhoc]  
Selandier, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-02, 2 November 2020, <<https://tools.ietf.org/html/draft-ietf-lake-edhoc-02>>.

- [RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <<https://www.rfc-editor.org/info/rfc8392>>.
- [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>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [I-D.richardson-enrollment-roadmap]  
Richardson, M., "Device Enrollment in IETF protocols -- A Roadmap", Work in Progress, Internet-Draft, draft-richardson-enrollment-roadmap-03, 7 October 2020, <<https://tools.ietf.org/html/draft-richardson-enrollment-roadmap-03>>.

## Author's Address

Jim Schaad  
August Cellars

Email: [ietf@augustcellars.com](mailto:ietf@augustcellars.com)