ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 26, 2019

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

June 24, 2019

CBOR Profile of X.509 Certificates
draft-raza-ace-cbor-certificates-02

Abstract

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

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

Status of This Memo

Table of Contents

1. Introduction

One of the challenges with deploying a Public Key Infrastructure (PKI) for the Internet of Things (IoT) is the size and encoding of X.509 public key certificates, since those are not optimized for constrained environments [RFC7228]. More compact certificate representations are desirable. Due to the current PKI usage of X.509 certificates, keeping X.509 compatibility is necessary at least for a transition period. However, the use of a more compact encoding with the Concise Binary Object Representation (CBOR) [I-D.ietf-cbor-7049bis] reduces the certificate size significantly which has known performance benefits in terms of decreased communication overhead, power consumption, latency, storage, etc.

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

CBOR data items are encoded to or decoded from byte strings using a type-length-value encoding scheme, where the three highest order bits of the initial byte contain information about the major type. CBOR supports several different types of data items, in addition to integers (int, uint), simple values (e.g. null), byte strings (bstr), and text strings (tstr), CBOR also supports arrays of data items and maps of pairs of data items. For a complete specification and examples, see [I-D.ietf-cbor-7049bis] and [RFC8610].

This document specifies the CBOR certificate profile, which is a CBOR based encoding and compression of the X.509 certificate format. The profile is based on previous work on profiling of X.509 certificates for Internet of Things deployments [X.509-IoT] which retains backwards compatibility with X.509, and can be applied for lightweight certificate based authentication with e.g. DTLS [RFC6347] or EDHOC [I-D.selander-ace-cose-ecdhe]. The same profile can be used for "native" CBOR encoded certificates, which further optimizes the performance in constrained environments but are not backwards compatible with X.509, see Section 6.

Other work has looked at reducing size of X.509 certificates. The purpose of this document is to stimulate a discussion on CBOR based certificates. Further optimizations of this profile are known and will be included in future versions.

o Terminology {#terminology}

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification makes use of the terminology in [RFC7228].

2. X.509 Certificate Profile

This profile is inspired by [RFC7925] and mandates further restrictions to enable reduction of certificate size. In this section we list the required fields in an X.509 certificate needed by devices in IoT deployments. The corresponding ASN.1 schema is given in Appendix B.

In order to comply with this certificate profile, the following restrictions MUST be applied:

- Version number. The X.509 standard has not moved beyond version 3 since 2008. With the introduction of certificate extensions new certificate fields can be added without breaking the format, making version changes less likely. Therefore this profile fixes the version number to 3.

- Serial number. The serial number together with the identity of the CA is the unique identifier of a certificate. The serial number MUST be an unsigned integer.

- Signature algorithm. For the CBOR profile, the signature algorithm is by default assumed to be ECDSA with SHA256.

- Issuer. Used to identify the issuing CA through a sequence of name-value pairs. This profile is restricting this to one pair, common name and associated string value. The common name MUST uniquely identify the CA. Other fields MUST NOT be used.

- Validity. The following representation MUST be used: UTCTime-format, YYMMDDhhmmss. This is the most compact format allowed by the X.509 standard.

- Subject. The subject section has the same format as the issuer, identifying the receiver of the public key through a sequence of name-value pairs. This sequence is in the profile restricted to a single pair, subject name and associated (unique) value. For an IoT-device, the MAC-derived EUI-64 serves this purpose well.

- Subject public key info. For the IoT devices, elliptic curve cryptography based algorithms have clear advantages. For the IoT profile the public key algorithm is by default assumed to be prime256v1.

- Issuer Unique ID and Subject Unique ID. These fields are optional in X.509 and MUST NOT be used with the CBOR profile.

- Extensions. Extensions consist of three parts; an OID, a boolean telling if it is critical or not, and the value. To maintain forward compatibility, the CBOR profile does not restrict the use of extensions. By the X.509-standard, any device must be able to process eight extensions types. Since only four of them are critical for IoT, this profile is making the other four optional. Still mandatory to be understood are:

  * Key Usage

  * Subject Alternative Name

  * Basic Constraints

  * Extended Key Usage

- Certificate signature algorithm. This field duplicates the info present in the signature algorithm field. By default assumed to be ECDSA with SHA256.

- Certificate Signature. The field corresponds to the signature done by the CA private key. For the CBOR profile, this is restricted to ECDSA type signatures with a signature length of 64 bits.

3. CBOR Encoding

This section specifies the CBOR certificates, which are the result of the CBOR encoding and lossless compression of the X.509 certificate profile of the previous section. The CDDL representation is given in Appendix A.

The encoding and compression has several components including: ASN.1 and base64 encoding is replaced with CBOR encoding, static fields are elided, and compression of elliptic curve points. The field encodings and associated savings are listed below. Combining these different components reduces the certificate size significantly, see Figure 1.

- Version number. The version number field is omitted in the encoding. This saves 5 bytes.

- Serial number. The serial number is encoded as an unsigned integer. Encoding overhead is reduced by one byte.

- Signature algorithm. If the signature algorithm is the default it is omitted in the encoding, otherwise encoded as a one byte COSE identifier. This saves 11 or 12 bytes.

- Issuer. Since the profile only allows the common name type, the common name type specifier is omitted. In total, the issuer field encoding overhead goes from 13 bytes to one byte.

- Validity. The time is encoded as UnixTime in integer format. The validity is represented as a 'not before'-'not after' pair of integer. This reduces the size from 32 to 11 bytes.

- Subject. An IoT subject is identified by a EUI-64, in turn based on a 48bit unique MAC id. This is encoded using only 7 bytes using CBOR. This is a reduction down from 36 bytes for the corresponding ASN.1 encoding.

- Subject public key info. If the algorithm identifier is the default, it is omitted, otherwise encoded as a one byte COSE identifier. For the allowed ECC type keys, one of the public key ECC curve point elements can be calculated from the other, hence only one of the curve points is needed (point compression, see [PointCompression]). These actions together, for the default algorithm, reduce size from 91 to 35 bytes.

- Extensions. Minor savings are achieved by the compact CBOR encoding. In addition, the relevant X.509 extension OIDs always start with 0x551D, hence these two bytes can be omitted.

- Certificate signature algorithm. This algorithm field is always the same as the above signature algorithm, and is omitted in the encoding.

- Signature. Since the signature algorithm and resulting signature length are known, padding and extra length fields which are present in the ASN.1 encoding are omitted. The overhead for encoding the 64-bit signature value is reduced from 11 to 2 bytes.

4. Deployment settings

CBOR certificates can be deployed with legacy X.509 certificates and CA infrastructure. In order to verify the signature, the CBOR certificate is used to recreate the original X.509 data structure to be able to verify the signature.

For the currently used DTLS v1.2 protocol, where the handshake is sent unencrypted, the actual encoding and compression can be done at different locations depending on the deployment setting. For example, the mapping between CBOR certificate and standard X.509 certificate can take place in a 6LoWPAN border gateway which allows the server side to stay unmodified. This case gives the advantage of the low overhead of a CBOR certificate over a constrained wireless

links. The conversion to X.509 within an IoT device will incur a computational overhead, however, this is negligible compared to the reduced communication overhead.

For the setting with constrained server and server-only authentication, the server only needs to be provisioned with the CBOR certificate and does not perform the conversion to X.509. This option is viable when client authentication can be asserted by other means.

For DTLS v1.3, because certificates are encrypted, the proposed encoding needs to be done fully end-to-end, through adding the encoding/decoding functionality to the server. A new certificate format or new certificate compression scheme needs to be added. While that requires changes on the server side, we believe it to be in line with other proposals utilizing cbor encoding for communication with resource constrained devices.

5. Expected Certificate Sizes

The profiling size saving mainly comes from enforcing removal of issuer and subject info fields besides the common name. The encoding savings are presented above in Section 3, for a sample certificate given in Appendix C resulting in the numbers shown in Figure 1.

After profiling, all duplicated information has been removed, and remaining text strings are minimal in size. Therefore no further size reduction can be reached with general compression mechanisms. (In practice the size might even grow slightly due to the compression encoding information, as illustrated in the table below.)

	X.509 Profiled	CBOR Encoded
Certificate Size	313	144

	X.509 Profiled	CBOR Encoded	Zlib
Certificate Size	313	144	319

Figure 1: Comparing Sizes of Certificates (bytes)

6. Native CBOR Certificates

Further performance improvements can be achieved with the use of native CBOR certificates. In this case the signature is calculated over the CBOR encoded structure rather than the ASN.1 encoded structure. This removes entirely the need for ASN.1 and reduces the processing in the authenticating devices.

This solution applies when the devices are only required to authenticate with a set of native CBOR certificate compatible servers, which may become a preferred approach for future deployments. The mapping between X.509 and CBOR certificates enables a migration path between the backwards compatible format and the fully optimized format.

7. Security Considerations

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

Conversion between the certificate formats can be made in constant time to reduce risk of information leakage through side channels.

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

8. Privacy Considerations

The mechanism in this draft does not reveal any additional information compared to X.509.

Because of difference in size, it will be possible to detect that this profile is used.

The gateway solution described in Section 4 requires unencrypted certificates.

9. IANA Considerations

This document registers a compression algorithm in the registry entitled "Certificate Compression Algorithm IDs", under the "Transport Layer Security (TLS) Extensions" heading (see [I-D.ietf-tls-certificate-compression]).

Algorithm Number	Description
TBD	cbor-iot

10. References

10.1. Normative References

[I-D.ietf-cbor-7049bis]
Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", draft-ietf-cbor-7049bis-05 (work in progress), January 2019.

[I-D.ietf-tls-certificate-compression]
Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", draft-ietf-tls-certificate-compression-05 (work in progress), April 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

10.2. Informative References

[I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-cose-ecdhe-13 (work in progress), March 2019.

[PointCompression]
Miller, V., "Use of Elliptic Curves in Cryptography.", Springer, Cham. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 218., 1986, <https://doi.org/10.1007/3-540-39799-X_31>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[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>.

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

Appendix A. CBOR Certificate, CDDL

```
   certificate = [
     serial_number : uint,
     issuer : text,
     validity : [notBefore: int, notAfter: int],
     subject : text / bytes
     public_key : bytes
     ? extensions : [+ extension],
     signature : bytes
     ? signature_alg + public_key_info : bytes
   ]

   extension = [
     oid : int,
     ? critical : bool,
     value : bytes
   ]
```

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

```
   IOTCertificate DEFINITIONS EXPLICIT TAGS ::= BEGIN

   Certificate  ::= SEQUENCE {
     tbsCertificate       TBSCertificate,
```

```
     signatureAlgorithm   SignatureIdentifier,
     signature            BIT STRING
   }

   TBSCertificate  ::= SEQUENCE {
     version       \[0\] INTEGER {v3(2)},
     serialNumber       INTEGER (1..MAX),
     signature       SignatureIdentifier,
     issuer       Name,
     validity       Validity,
     subject       Name,
     subjectPublicKeyInfo       SubjectPublicKeyInfo,
     extensions       \[3\] Extensions OPTIONAL
   }

   SignatureIdentifier ::= SEQUENCE {
     algorithm       OBJECT IDENTIFIER (ecdsa-with-SHA256)
   Name  ::= SEQUENCE SIZE (1) OF DistinguishedName
   DistinguishedName  ::= SET SIZE (1) OF CommonName
   CommonName  ::= SEQUENCE {
     type       OBJECT IDENTIFIER (id-at-commonName),
     value       UTF8String
         -- For a CA, value is CA name, else EUI-64 in format
         -- "01-23-54-FF-FE-AB-CD-EF"
   }

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

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

   AlgorithmIdentifier ::= SEQUENCE {
     algorithm        OBJECT IDENTIFIER (id-ecPublicKey),
     parameters       OBJECT IDENTIFIER (prime256v1)
   }
     Extensions  ::= SEQUENCE SIZE (1..MAX) OF Extension

   Extension  ::= SEQUENCE {
     extnId          OBJECT IDENTIFIER,
     critical        BOOLEAN DEFAULT FALSE,
     extnValue       OCTET STRING
    }
```

```
   ansi-X9-62          OBJECT IDENTIFIER   ::=
            {iso(1) member-body(2) us(840) 10045}

   id-ecPublicKey      OBJECT IDENTIFIER   ::=
            {ansi-X9-62 keyType(2) 1}

   prime256v1          OBJECT IDENTIFIER   ::=
            {ansi-X9-62 curves(3) prime(1) 7}

   ecdsa-with-SHA256   OBJECT IDENTIFIER   ::=
            {ansi-X9-62 signatures(4) ecdsa-with-SHA2(3) 2}

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

   END
```

Appendix C. Certificate Example

This section shows an example of an X.509 profiled certificate before CBOR encoding.

```
   Certificate:
       Data:
           Version: 3 (0x2)
           Serial Number: DEC (HEX)
       Signature Algorithm: ecdsa-with-SHA256
           Issuer: <23 byte issuer ID>
           Validity
               Not Before: <not_before_ts>
               Not After : <not_after_ts>
           Subject: <23 byte UID>
           Subject Public Key Info:
               Public Key Algorithm: id-ecPublicKey
                   Public-Key: (256 bit)
                   pub:
                       .. .. ..
                   ASN1 OID: prime256v1
                   NIST CURVE: P-256
           X509v3 extensions:
               X509v3 Basic Constraints: critical
                   CA:FALSE
               X509v3 Key Usage:
                   Digital Signature, Key Encipherment
       Signature Algorithm: ecdsa-with-SHA256
            .. .. ...
```

Authors' Addresses

Shahid Raza
RISE AB

Email: shahid.raza@ri.se

Joel Hoeglund
RISE AB

Email: joel.hoglund@ri.se

Goeran Selander
Ericsson AB

Email: goran.selander@ericsson.com

John Mattsson
Ericsson AB

Email: john.mattsson@ericsson.com

Martin Furuhed
Nexus Group

Email: martin.furuhed@nexusgroup.com