CBOR Encoded X.509 Certificates (C509)

draft-ietf-cose-cbor-encoded-cert-06

IETF 117, 2023-07-24
G. Selander, J. Mattsson (Ericsson AB),
Joel Höglund, S. Raza (RISE),
M. Furuhed (Nexus Group)
Brief status update

New version -06 address a number of open issues raised by the community and known TODOs.

- New IANA section for Media Type application/cose-c509
- New and updated COSE Header Parameters
- Support for uncompressed ECC keys also in native C509 certificates
- Additional EC curves
- Several minor bug-fixes, and reference updates to now completed RFCs

Issues tracked at
https://github.com/cose-wg/CBOR-certificates/

Special thanks to Ilari Liusvaara, for many useful observations and comments which have helped to improve the draft
Overall trade-offs and discussion themes

- How much to include in main draft vs creating more drafts
- Compactness / saving bytes
- Convenient to parse and process
- Generality, how to encode as many relevant X.509 certificates as possible
CBOR Sequence parsing in libraries (#76)

Context: This is an old issue that has been kept open since the outcome of the earlier discussion was to evaluate at a later point.

A reviewer proposed to wrap TBSCertificate in a byte string, because some CBOR decoders don’t allow access a sub-part of the encoded CBOR so it can be input into the signature algorithm.

Our proposal: to leave as is, trust modern parsers to handle it.

Comments?
Certificate chain optimizations (#82)

Context: another old issue which has been kept open to allow further discussions

CBOR certs could provide optimizations for self-issued certificates as well as for certs that are sent in cert chains.

Q: Should CBOR certs provide optimizations for self-issued certs or chains?

• Potentially large savings.
• Added complexity, makes CBOR compression two pass
• Could be handled through COSE headers + Brotli

⇒ Our proposal: to keep as is, to keep the implementations simple, avoiding two-pass
Compression of extensions, certificates, or chains (#98, similar to #86)

The TLS size examples in the -01 draft shows that even after C509 encoding, Brotli can still compress a certificate chain quite much. Similar for RPKI certs.

Should COSE define the use of Brotli?

- Brotli, or some other general compression mechanism could be used in COSE.
  - Likely after the signature has been calculated, i.e. not in individual extensions.
- C509 could specify a compressed cert type which take a C509 cert and produce a compressed C509.
- COSE_C509 could specify a compressed chain cert type which take a COSE_C509 cert and produce a compressed COSE_C509.
More extensions having integer values (#111)

<table>
<thead>
<tr>
<th>Extension</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiometricInfo</td>
<td>(RFC 3739)</td>
</tr>
<tr>
<td>CT Precertificate SCTs</td>
<td>(RFC 6962)</td>
</tr>
<tr>
<td>OcspNoCheck</td>
<td>(RFC 6960)</td>
</tr>
<tr>
<td>QCStatements</td>
<td>(RFC 3739, eIDAS eN 319 412)</td>
</tr>
<tr>
<td>SMIMECapabilities</td>
<td>(RFC 4262)</td>
</tr>
<tr>
<td>TLSFeature</td>
<td>(RFC 7633)</td>
</tr>
</tbody>
</table>

⇒ Our proposal: to add them to the C509 Extensions Registry, without new cbor-specific encodings of the actual extension values
CRL + OCSP break-out proposal

We propose to break out the sections on messages for revocation lists (CRL) and Online Certificate Status Protocol messages (OCSP) to form a separate draft.

- Prevents discussion that only concerns the revocation handling from slowing down the main C509 progress

Comments?
First draft of CBOR OCSP Request (not optimized)

C509OCSPRequest = [ 
    TBSRequest, 
    optionalSignature : *any, 
]

TBSRequest = [ 
    version : uint .default 1, 
    requestorName : *GeneralName, 
    requestList : [+Request], 
    requestExtensions : *+[extension] 
]

Request = [ 
    reqCert : CertID, 
    singleRequestExtensions : *+[extension] 
]

CertID = [ 
    hashAlgorithm : AlgorithmIdentifier, 
    issuerNameHash : bytes, -- Hash of issuer's DN 
    issuerKeyHash : bytes, -- Hash of issuer's public key 
    serialNumber : CertificateSerialNumber 
]

extension = TBD

Several optimizations are possible for a native CBOR format, in particular CertID
C509OCSPResponse = [  
  responseStatus : C509OCSPResponseStatus,
  responseBytes : *BasicOCSPResponse
]

OCSPResponseStatus = 0..6 ; inclusive range  
; semantics of integer values as in rfc6960, 4.2.1. ASN.1 Specification of the OCSP Response

BasicOCSPResponse  = [  
  tbsResponseData : ResponseData,
  signatureAlgorithm : AlgorithmIdentifier,
  signature : any,
  certs : *[C509Certificate]
]

ResponseData = [  
  version : uint .default 1,
  responderID : ResponderID,
  producedAt : Time,
  responses : [+SingleResponse],
  responseExtensions : *+[extension]
]

ResponderID = Name / KeyHash

Name = TBD

KeyHash = bytes -- SHA-1 hash of responder's public key (excluding the tag and length fields)  
; OBSOLETE, but needed if we want to fully recreate RFC6960 style messages
Native CBOR OCSP responses could be greatly reduced by not repeating known data.
Next steps

- Handle remaining issues
- Incorporate/remove revocation data formats
- Make ready for WGLC