Status, ongoing work and issues for further discussions
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
New version 06 has addressed a number of open issues raised by the community and know 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

A special thanks to Ilari Liusvaara, for the many useful observations and comments which have helped to improve the draft
Ongoing: 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.

• Other operations, such as enrollment, using C509 are already treated separately in their own drafts
• Prevents discussion that only concerns the revocation handling from slowing down the main C509 progress
Ongoing: First draft of CBOR OCSP

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

Please note: existing format is lengthy and ill-suited for IoT

Several optimizations possible for a native CBOR format, such as:  
xyz-based CertID
Ongoing: First draft of CBOR OCSP

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

Please note: existing format is lengthy and ill-suited for IoT
Ongoing: First draft of CBOR OCSP

SingleResponse = [ 
    certID : CertID,
    certStatus : CertStatus,
    thisUpdate : Time,
    nextUpdate : *Time,
    singleExtensions : *+[extension]
]

CertStatus = {1: NULL} / {2: RevokedInfo} / {3: NULL}
    ; good / revoked / unknown, semantics from RFC6960

RevokedInfo = ( 
    revocationTime : Time,
    revocationReason : *CRLReason
)

CRLReason = 0..10 ; inclusive range
    ; semantics of integer values from RFC6960, 5.3.1.

Native CBOR OCSP replies could be greatly reduced by not repeating known data
Context: An old issue that has been kept open since the outcome of earlier discussions was to evaluate at a later point.

Some CBOR decoders don’t allow access a sub-part of the encoded CBOR so it can be input into the signature algorithm.

• The commenters proposal was to wrap TBSCertificate in a byte string

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

Potential discussions:

• 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.
Discussions: Integer value for more extensions (#111)

BiometricInfo (RFC 3739)
CT Precertificate SCTs (RFC 6962)
OcspNoCheck (RFC 6960)
QCStatements (RFC 3739, eIDAS eN 319 412)
SMIMECapabilities (RFC 4262)
TLSFeature (RFC 7633)

⇒ Our proposal: to added them to the C509 Extensions Registry, without new cbor-specific encodings of the actual extension values
Discussions

• The above mentioned issues
• Any other topic of interest
Göran Selander, John Mattsson
Ericsson AB
goran.selander@ericsson.com,
john.mattson@ericsson.com

Joel Höglund, RISE
joel.hoglund@ri.se