Workgroup: Network Working Group
Published: 17 September 2020
Intended Status: Informational
Expires: 21 March 2021
Authors: J. Schaad
August Cellars

CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates

Abstract

The CBOR Signing And Encrypted Message (COSE) structure uses references to keys in general. For some algorithms, additional properties are defined which carry parts of 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

Copyright Notice

Table of Contents

1. Introduction

In the process of writing [RFC8152] the working group discussed X.509 certificates [RFC5280] ad decided that no use cases wher prestented 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 in the device of a factory-installed certificate. The [I-D.selander-ace-cose-ecdhe] 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. A third 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.

Example COSE messages for the various header parameters defined below can be found at https://github.com/cose-wg/Examples.

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 of X.509 certificates: it is perfectly reasonable for a constrained device to be provisioned with a certificate which it can then provide 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 use a public key for that thumbprint, but without performing the certificate evaluation or even having the entire certificate.

Certificates obtained from any of these methods MUST still be validated. 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 to certificates of a chain length of one as well as longer chains. If the application cannot establish trust in the certificate, that certificate cannot be used.

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. 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 be used as a signal to modify the set of trust anchors. 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 bstr.

* If multiple certificates are conveyed, a CBOR array of bstrs is used, with each certificate being in its own bstr.

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 attribute.

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 be used as a signal to modify the set of trust anchors. 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 bstr.

- If multiple certificates are conveyed, a CBOR array of bstrs is used, with each certificate being in its own bstr.

x5t: This header parameter provides the ability to identify an X. 509 certificate by a hash value. The attribute is an array of two elements. The first element is an algorithm identifier which is an integer or a string containing the hash algorithm identifier. The algorithm is registered in the "COSE Algorithms" registry The second element is a binary string containing the hash value.

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.

RFC Editor please remove the following 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]. 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, the attribute 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 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. In particular, self-signed certificates MUST NOT be trusted without an out-of-band confirmation.

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.

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 ]
```

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+AES192KW, ECDH-SS+AES256KW	Thumbprint for the senders X.509 certificate
x5u-sender	TBD	uri	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+AES192KW, ECDH-SS+AES256KW	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+AES192KW, ECDH-SS+AES256KW	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, 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).

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>.

[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-dtls13-38, 29 May 2020, <https://tools.ietf.org/html/draft-ietf-tls-dtls13-38>.

[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.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-selander-ace-cose-ecdhe-14, 11 September 2019, <https://tools.ietf.org/html/draft-selander-ace-cose-ecdhe-14>.

[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>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[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-02, 23 May 2018, <https://tools.ietf.org/html/draft-richardson-enrollment-roadmap-02>.

Author's Address

Jim Schaad
August Cellars

Email: ietf@augustcellars.com