COSE
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2025

H. Tschofenig
H-BRS
B. Moran
Arm Limited
H. Birkholz
Fraunhofer SIT
2 March 2025

CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing Chains of CBOR Web Tokens (CWTs)
draft-tschofenig-cose-cwt-chain-02

Abstract

The CBOR Object Signing and Encryption (COSE) message structure uses references to keys and defines header parameters to carry chains of X.509 certificates.

This specification extends this functionality to CBOR Web Tokens (CWTs).

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 3 September 2025.

Copyright Notice

Table of Contents

1. Introduction

The CBOR Object Signing and Encryption (COSE) message structure uses references to keys and defines header parameters to carry chains of X.509 certificates. The header parameters for conveying X.509 certificate chains in a COSE payload are defined in [RFC9360].

This document is inspired by RFC 9360 and defines header parameters to convey chains of CBOR Web Tokens (CWTs) [RFC8392]. The use of chains of CWTs allows a trust infrastructure established by CWTs to be used with COSE. The Concise Binary Object Representation (CBOR) key structures [RFC8949] that have been defined in COSE support the use of X.509 certificates. This specification applies the well-proven concepts to CWTs. These chains of CWTs allow path validation similarly to what a X.509 certificate-based Public Key Infrastructure (PKI) provides. Since [RFC8747] does not define the semantics of path validation for CWTs, new terminology is introduced.

This document is structured as follows: After introducing some terms, we describe path validation for CWTs. Then, we define new header parameters.

2. Terminology and Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The following terms are useful for readers of this document:

* End Entity: user of CWT and/or end user system that is the subject of a CWT;

* CA: certification authority; RFC 8747 calls this entity the "issuer" and describes it as "the party that creates the CWT and binds the claims about the subject to the proof-of-possession key". In an OAuth-based system, this entity often corresponds to an authorization server.

* CA CWT: A CWT that is self-issued whereby the same name appears in the subject and issuer claims.

* RA: registration authority, i.e., an optional system to which a CA delegates certain management functions; while often used in PKI deployments it is a role that has not found usage in systems using CWTs.

* CRL issuer: a system that generates and signs Certificate Revocation Lists (CRLs); The term CRL is used generically to also refer to status lists [I-D.ietf-oauth-status-list].

* repository: a system or collection of distributed systems that stores CWTs and CRLs and serves as a means of distributing these CWTs and CRLs to end entities. These repositories may be append-only databases, in the style of [I-D.ietf-keytrans-architecture].

* Trust Anchor: As defined in [RFC6024] and [RFC9019], a Trust Anchor "represents an authoritative entity via a public key and associated data. The public key is used to verify digital signatures, and the associated data is used to constrain the types of information for which the trust anchor is authoritative." The trust anchor may be a CWT, a raw public key, or other structure, as appropriate.

* Subject Public Key (Info): The "confirmation" claim, defined in [RFC8747], used to carry the public key and the algorithm with which the key is used.

3. CWT Path Validation

The goal of path validation is to verify the binding between a subject name and the public key, as represented in the target CWT, based on the public key of the trust anchor. In most cases, the target CWT will be an end entity CWT. Verifying the binding between the name and subject public key requires obtaining a sequence of certificates that support that binding. For path validation to work CWTs that have a minimum number of claims, namely:

* Subject

* Issuer

* Confirmation

Valid paths begin with CWTs issued by a trust anchor and the trust anchor is an input to the algorithm. The algorithm in Section 6 of [RFC5280] requires the public key of the CA, the CA's name, and any constraints upon the set of paths that may be validated using this key.

The path validation algorithm verifies that a prospective certification path (a sequence of n CWTs) satisfies the following conditions:

(a) for all x in {1, ..., n-1}, the subject of CWT x is the issuer of CWT x+1;

(b) CWT 1 is issued by the trust anchor;

(c) CWT n is the CWT to be validated (i.e., the target CWT); and

Note: When the trust anchor is provided in the form of a self-signed CWT, this self-signed CWT is not included as part of the prospective certification path.

As a variation to the algorithm presented in Section 6 of [RFC5280], there is no strict requirement for a CWT being valid in terms of its lifetime (as indicated by the "Expiration Time" and the "Not Before" claims) since CWTs may not necessarily carry these claims and validatity may be determined via different means, which are outside the scope of this algorithm.

Path validation is an important part of establishing trust in a CWT and when applying path validation, as defined in Section 6 of[RFC5280], to CWTs the reader needs to treat them as certificates. It is important to keep in mind that many of the advanced features

available with an X.509 certificate-based PKI are, at the time of writing, not available with CWTs. The authors do, however, believe that differences will decrease over time as CWT-based deployments scale.

4. CWT COSE Header Parameters

Parties that intend to rely on the assertions made by a CWTs obtained from any of these methods still need to validate it. This validation can be done according to the PKIX rules specified in [RFC5280] or by using a different trust structure, such as a trusted distributor for self-signed CWTs. 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 CWT chains need to be built. If the application cannot establish trust in the CWT, the public key contained in the CWT cannot be used for cryptographic operations.

The header parameters defined in this document are as follows:

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

The trust mechanism MUST process any CWT in this parameter as untrusted input. The presence of a self-signed CWT 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. Sending the header parameter in the unprotected header bucket allows an intermediary to remove or add CWT.

The end entity CWT MUST be integrity protected by COSE. This can, for example, be done by sending the header parameter in the protected header, sending an 'cwt-bag' in the unprotected header combined with an 'cwt-t' in the protected header, or including the end entity CWT in the external_aad.

This header parameter allows for a single CWT or a bag of CWT to be carried in the message.

* If a single CWT is conveyed, it is placed in a CBOR byte string.

* If multiple CWTs are conveyed, a CBOR array of byte strings is used, with each CWT being in its own byte string.

cwt-chain: This header parameter contains an ordered array of CWTs. The CWTs are to be ordered starting with the CWT containing the end entity key followed by the CWT that 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 CWT. 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 CWT in this parameter as untrusted input. The presence of a self-signed CWT 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. Sending the header parameter in the unprotected header bucket allows an intermediary to remove or add CWT.

The end entity CWT MUST be integrity protected by COSE. This can, for example, be done by sending the header parameter in the protected header, sending an 'cwt-chain' in the unprotected header combined with an 'cwt-t' in the protected header, or including the end entity CWT in the external_aad.

This header parameter allows for a single CWT or a chain of CWTs to be carried in the message.

* If a single CWT is conveyed, it is placed in a CBOR byte string.

* If multiple CWTs are conveyed, a CBOR array of byte strings is used, with each CWT being in its own byte string.

cwt-t: This header parameter identifies the end entity CWT by a hash value (a thumbprint). The 'cwt-t' header parameter is represented as an array of two elements. The first element is an algorithm identifier that is an integer or a string containing the hash algorithm identifier corresponding to the Value column (integer or text string) of the algorithm registered in the "COSE Algorithms" registry (see [COSE-IANA]). The second element is a binary string containing the hash value computed over the CWT.

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

The identification of the end entity CWT MUST be integrity protected by COSE. This can be done by sending the header parameter in the protected header or including the end entity CWT in the external_aad.

The 'cwt-t' header parameter can be used alone or together with the 'cwt-bag', 'cwt-chain', or 'cwt-u' header parameters to provide integrity protection of the end entity CWT.

For interoperability, applications that 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.

Note: For conveying the thumbprint of a public key alone, see {{RFC9679}}.

cwt-u: This header parameter provides the ability to identify a CWT by a URI [RFC3986]. It contains a CBOR text string. The referenced resource can be any of the following media types:

* application/cwt {{RFC8392}}

* application/cwt usage=chain (see {{chain}})

When the application/cwt media type is used, the data is a encoded according to RFC 8392. If the parameter "usage" is set to "chain", this sequence indicates a CWT chain.

The end entity CWT MUST be integrity protected by COSE. This can, for example, be done by sending the 'cwt-u' in the unprotected or protected header combined with an 'cwt-t' in the protected header, or including the end entity CWT in the external_aad. As the end entity CWT is integrity protected by COSE, the URI does not need to provide any protection.

If a retrieved CWT does not chain to an existing trust anchor, that CWT MUST NOT be trusted unless the URI provides integrity protection and server authentication and 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 CWT. If an HTTP or Constrained Application Protocol (CoAP) GET request is used to retrieve a CWT, a standardized security protocol should be used. Examples of such security protocols include TLS {{RFC8446}}, DTLS {{RFC9147}}, or Object Security for Constrained RESTful Environments (OSCORE) {{RFC8613}} should be used.

The header parameters are used in the following locations:

COSE_Signature and COSE_Sign1 objects: In these objects, the parameters identify the CWT to be used for validating the signature.

COSE_recipient objects: In this location, the parameters identify the CWT for the recipient of the message.

The labels assigned to each header parameter can be found in Figure 1.

Name	Label	Value Type	Description
cwt-bag	TBD1	COSE_CWT	An unordered bag of CWTs
cwt-chain	TBD2	COSE_CWT	An ordered chain of CWTs
cwt-t	TBD3	COSE_CWTHash	Hash of a CWT
cwt-u	TBD4	uri	URI pointing to a CWT

Figure 1: CWT COSE Header Parameters.

Below is an equivalent Concise Data Definition Language (CDDL) description (see [RFC8610]) of the text above.

```
COSE_CWT = CWT-Messages / [ 2*CWT-Messages ]
COSE_CWTHash = [ hashAlg: (int / tstr), hashValue: bstr ]
```

The contents of "bstr" are the bytes of a CWT.

5. CWTs and Static-Static ECDH

The header parameters defined in the previous section are used to identify the recipient CWT. 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 CWT bag, as the same parameter would be used for both the sender and recipient.

cwt-chain-sender: This header parameter contains the chain of CWT starting with the sender's key exchange CWT. The structure is the same as 'cwt-chain'.

cwt-t-sender: This header parameter contains the hash value for the sender's key exchange CWT. The structure is the same as 'cwt-t'.

cwt-u-sender: This header parameter contains a URI for the sender's key exchange CWT. The structure and processing are the same as 'cwt-u'.

Name	Label	Type	Algorithm	Description
cwt-t-sender	TBD5	COSE_CWTHash	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Thumbprint for the sender's CWT
cwt-u-sender	TBD6	uri	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	URI for the sender's CWT
cwt-chain-sender	TBD7	COSE_CWT	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	static key CWT chain

Figure 2: Static ECDH Algorithm Values.

6. Example

TBD

7. Security Considerations

Establishing trust in a CWT 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 CWT 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 to (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 CWT for validity. Validating CWTs may require that network resources be accessed in order to get statys information or retrieve CWTs during path building. The resulting network access can consume power and network bandwidth. On the other hand, if the CWT 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 validation and the CWT validation MUST be completed successfully before acting on any requests.

The end entity CWT MUST be integrity protected by COSE. Without proof of possession, an attacker can trick the CA into issuing an identity-misbinding CWT with someone else's "borrowed" public key but with a different subject. An on-path attacker can then perform an identity-misbinding attack by replacing the real end entity CWT in COSE with such an identity- misbinding CWT.

end entity CWTs contain identities that a passive on-path attacker eavesdropping on the conversation can use to identify and track the subject. The 'cwt-t' and 'cwt-u' header parameters are just alternative permanent identifiers and can also be used to track the subject. To provide identity protection, COSE can be sent inside another security protocol providing confidentiality. Additionally, the encryption capabilities of COSE itself can be used to protect the CWT content.

When processing the 'cwt-u' header parameter, the security considerations of [RFC3986], and specifically those defined in Section 7.1 of [RFC3986], also apply.

Protecting the integrity of the 'cwt-bag', 'cwt-chain', and 'cwt-t' contents by placing them in the protected header bucket can help mitigate some risks of a misbehaving CA (cf. Section 5.1 of [RFC2634]).

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

8. IANA Considerations

8.1. COSE Header Parameters Registry

IANA has registered the new COSE Header parameters in Figure 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.

8.2. COSE Header Algorithm Parameters Registry

IANA has registered the new COSE Header Algorithm parameters in Figure 2 in the "COSE Header Algorithm Parameters" registry. For each item, the "Reference" field points to this document.

8.3. Media Type application/cwt

When the application/cwt media type is used, the data is a CBOR sequence of single-entry COSE_CWT structures (encoding "bstr"). If the parameter "usage" is set to "chain", this sequence indicates a CWT chain.

The application/cwt media type is already registered by [RFC8392] and this document updates the IANA entry of this media type [RFC6838]:

* Type name: application

* Subtype name: cwt

* Required parameters: N/A

* Optional parameters: usage

  - Can be absent to provide no further information about the intended meaning of the order in the CBOR sequence of CWT.

  - Can be set to "chain" to indicate that the sequence of data items is to be interpreted as a CWT chain.

* Encoding considerations: binary

* Security considerations: See the Security Considerations section of RFC 8392 and [TBD: This RFC].

* Interoperability considerations: N/A

* Published specification: RFC 8392 and [TBD: This RFC]

* Applications that use this media type: Applications that employ COSE and use CWTs, including IoT applications and digital credentials in general.

* Fragment identifier considerations: N/A

* Additional information:

  - Deprecated alias names for this type: N/A

  - Magic number(s): N/A

  - File extension(s): N/A

  - Macintosh file type code(s): N/A

* Person & email address to contact for further information: iesg@ietf.org

* Intended usage: COMMON

* Restrictions on usage: N/A

* Author: COSE WG

* Change controller: IESG

Provisional registration? No

9. References

9.1. 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/rfc/rfc2119>.

[RFC2634] Hoffman, P., Ed., "Enhanced Security Services for S/MIME", RFC 2634, DOI 10.17487/RFC2634, June 1999, <https://www.rfc-editor.org/rfc/rfc2634>.

[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/rfc/rfc3986>.

[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/rfc/rfc5280>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[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/rfc/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/rfc/rfc8610>.

[RFC8747] Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", RFC 8747, DOI 10.17487/RFC8747, March 2020, <https://www.rfc-editor.org/rfc/rfc8747>.

[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/rfc/rfc8949>.

9.2. Informative References

[COSE-IANA]
IANA, "CBOR Object Signing and Encryption (COSE) IANA Registry", December 2023, <https://www.iana.org/assignments/cose/>.

[I-D.ietf-keytrans-architecture]
McMillion, B., "Key Transparency Architecture", Work in Progress, Internet-Draft, draft-ietf-keytrans-architecture-03, 25 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-keytrans-architecture-03>.

[I-D.ietf-oauth-status-list]
Looker, T., Bastian, P., and C. Bormann, "Token Status List", Work in Progress, Internet-Draft, draft-ietf-oauth-

status-list-08, 19 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-status-list-08>.

[RFC6024] Reddy, R. and C. Wallace, "Trust Anchor Management Requirements", RFC 6024, DOI 10.17487/RFC6024, October 2010, <https://www.rfc-editor.org/rfc/rfc6024>.

[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/rfc/rfc8446>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/rfc/rfc8613>.

[RFC9019] Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", RFC 9019, DOI 10.17487/RFC9019, April 2021, <https://www.rfc-editor.org/rfc/rfc9019>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", RFC 9360, DOI 10.17487/RFC9360, February 2023, <https://www.rfc-editor.org/rfc/rfc9360>.

[RFC9679] Isobe, K., Tschofenig, H., and O. Steele, "CBOR Object Signing and Encryption (COSE) Key Thumbprint", RFC 9679, DOI 10.17487/RFC9679, December 2024, <https://www.rfc-editor.org/rfc/rfc9679>.

Appendix A. Contributor

We would like to thank Ken Takayama for his work on the IETF SUIT trust domains draft, which created the idea for writing this specification. Ken provided valuable review feedback.

Appendix B. Acknowledgments

Add your name here.

Authors' Addresses

Hannes Tschofenig
University of Applied Sciences Bonn-Rhein-Sieg
Germany
Email: Hannes.Tschofenig@gmx.net

Brendan Moran
Arm Limited
Email: brendan.moran.ietf@gmail.com

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@sit.fraunhofer.de