Stream: Internet Engineering Task Force (IETF)
RFC: 9995
Category: Standards Track
Published: July 2026
ISSN: 2070-1721
Authors: O. Steele S. Lasker H. Birkholz
Fraunhofer SIT

RFC 9995
CBOR Object Signing and Encryption (COSE) Hash Envelope

Abstract

This document defines new CBOR Object Signing and Encryption (COSE) header parameters for signaling a payload as an output of a hash function. This mechanism enables faster validation, as access to the original payload is not required for signature validation. Additionally, hints of the hashed payload's content format and availability are defined, providing references to optional discovery mechanisms that can help to find the original payload content.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9995.

Copyright Notice

Table of Contents

1. Introduction

Section 2 of [RFC9052] defines detached payloads for COSE, using `nil` as the payload. In order to verify a COSE_Sign or a COSE_Mac, the recipient requires access to the payload content. Hashes are already used on a regular basis as identifiers for payload data, such as documents or software components. As hashes typically are smaller than the payload data they represent, they are simpler to transport. Additional hints in the protected header ensure cryptographic agility for the hashing and signing algorithms. Hashes and other identifiers are commonly used as hints to discover and distinguish resources. Using a hash as an identifier for a resource has the advantage of enabling integrity checking.

In some applications, such as remote signing procedures, conveyance of hashes instead of original payload content reduces transmission time and costs.

2. 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.

The terms "COSE" and "CDDL" are defined in [RFC9052] and [RFC8610], respectively. The term "payload" is defined in Section 4.1 of [RFC9052] for COSE_Sign and in Section 6.1 of [RFC9052] for COSE_Mac. The term "preimage" refers to the set of input values to a function that produce a given output, called the "image". A hash function applied to a message (preimage) produces a digest value (image).

3. Header Parameters

This document specifies the following new header parameters commonly used alongside hashes to identify resources:

258: The hash algorithm used to produce the payload.

259: The content type of the bytes that were hashed (preimage) to produce the payload, given as a content-format number (Section 12.3 of [RFC7252]) or as a media-type name optionally with parameters (Section 8.3 of [RFC9110]).

260: An identifier enabling retrieval of the original resource (preimage) identified by the payload.

4. Hash Envelope CDDL

```
<CODE BEGINS>
Hash_Envelope = #6.18(Hash_Envelope_as_COSE_Sign1)

Hash_Envelope_as_COSE_Sign1 = [
    protected: bstr .cbor Hash_Envelope_Protected_Header,
    unprotected: Hash_Envelope_Unprotected_Header,
    payload: bstr / nil,
    signature: bstr
]

Hash_Envelope_Protected_Header = {
    ? &(alg: 1) => int,
    &(payload_hash_alg: 258) => int,
    ? &(payload_preimage_content_type: 259) => uint / tstr,
    ? &(payload_location: 260) => tstr,
    * (int / tstr) => any
}

Hash_Envelope_Unprotected_Header = {
    * (int / tstr) => any
}

<CODE ENDS>
```

- Label `1` (alg) is the cryptographic algorithm to use.
- Label `258` (payload_hash_alg) **MUST** be present in the protected header and **MUST NOT** be present in the unprotected header.
- Label `259` (payload_preimage_content_type) **MAY** be present in the protected header and **MUST NOT** be present in the unprotected header.
- Label `260` (payload_location) **MAY** be present in the protected header and **MUST NOT** be present in the unprotected header.
- Label `3` (content_type) **MUST NOT** be present in the protected or unprotected headers.

Label `3` (content_type) is easily confused with Label `259` (payload_preimage_content_type). The difference between content_type (3) and payload_preimage_content_type (259) is that content_type is used to identify the content format associated with payload, whereas payload_preimage_content_type is used to identify the content format of the bytes that are hashed to produce the payload.

Output from hash algorithms is generally small; thus, the payload is typically expected to be inline. But it can also be detached, as in any other COSE message [RFC9052].

For example, when the actual content is a byte string (bstr), a verifier appraising the payload has to decide whether that bstr represents the digest bytes or the preimage bytes. Setting payload_preimage_content_type to bstr makes clear that the preimage bytes themselves were a bstr.

4.1. Envelope Extended Diagnostic Notation

The following informative example uses Extended Diagnostic Notation as defined in Appendix G of [RFC8610] and demonstrates how to construct a hash envelope for a resource already commonly referenced by its hash.

```
18([ # COSE_Sign1
  <<{
    / signature alg   / 1: -35, # ES384
    / key identifier  / 4: h'75726e3a...32636573',
    / COSE_Sign1 type / 16: "application/example+cose",
    / hash algorithm  / 258: -16, # sha256
    / media type      / 259: "application/spdx+json",
    / location        /
         260: "https://sbom.example/.../manifest.spdx.json"
  }>>
  / unprotected / {},
  / payload     / h'935b5a91...e18a588a',
         # SHA-256 digest of manifest.spdx.json"
  / signature   / h'15280897...93ef39e5'
         # ECDSA Signature with SHA-384 and P-384
])
```

In this example, a System Package Data Exchange [SPDX] Software Bill of Materials (SBOM) in JSON format is already commonly identified by its SHA-256 hash. The content type for "manifest.spdx.json" is already well known as "application/spdx+json" and is registered with IANA.

The full JSON SBOM is available at a URL, such as "https://sbom.example/.../manifest.spdx.json".

The payload of this COSE_Sign1 is the SHA-256 hash of "manifest.spdx.json".

The type of this COSE_Sign1 is "application/example+cose", but other types may be used to establish more-specific media types for signatures of hashes.

The signature is produced using ES384, as defined in Section 3.4 of [RFC7518], which means using the Elliptic Curve Digital Signature Algorithm (ECDSA) with the SHA-384 hash function and P-384 elliptic curve.

This example is chosen to highlight that an existing system may use a hash algorithm such as SHA-256. This hash becomes the payload of a COSE_Sign1. When signed with a signature algorithm that is parameterized via a hash function, such as ECDSA with SHA-384, the to-be-signed structure is as described in Section 4.4 of [RFC9052].

The resulting signature is computed over the protected header and payload, providing integrity and authenticity for the hash algorithm, content type, and location of the associated resource, in this case a software bill of materials.

5. Security Considerations

5.1. Choice of Hash Function

The hash/signature algorithm combination is **RECOMMENDED** to be at least as strong as the payload hash algorithm. For example, if the payload was produced with SHA-256, and is signed with ECDSA, use at least P-256 and SHA-256. Note that, when using a pre-hash algorithm, the algorithm **MUST** be registered in the IANA "COSE Algorithms" registry [COSE-Algorithms] and **MUST** be distinguishable from non-pre-hash variants that may also be present.

5.2. COSE_Encrypt

Only COSE_Sign/COSE_Sign1 and COSE_Mac/COSE_Mac0 are in scope for this document. COSE_Encrypt/COSE_Encrypt0 is out of scope for this document. At the time of publication, there is no known use case for COSE_Encrypt/COSE_Encrypt0. It may be covered by a future extension, which would address whether the hash function is applied before or after encryption and clarify privacy considerations.

5.3. Payload Verification

If a payload-location is specified, a verifier can choose to fetch the content and confirm that the digest of it, produced with the function defined by payload-hash-alg, matches the payload bytes. Verifiers that do not have access to the internet and obtain the preimage via other means will not be able to perform that check nor to derive utility from it.

6. IANA Considerations

6.1. COSE Header Parameters

IANA has registered the COSE header parameters defined in Section 3 (as listed in Table 1) in the "COSE Header Parameters" registry [COSE-HDR-PARAMS]. They have been registered in the 'Integer values from 256 to 65535' range per the 'Specification Required' registration policy [RFC8126].

Name	Label	Value Type	Value Registry	Description	Reference
`payload-hash-alg`	258	int	[COSE-Algorithms]	The hash algorithm used to produce the payload of a COSE_Sign1	RFC 9995, Section 3

Name	Label	Value Type	Value Registry	Description	Reference
`preimage-content-type`	259	uint / tstr	[CoAP-Content-Formats]	The content-format number or content-type (media-type name) of data that has been hashed to produce the payload of the COSE_Sign1	RFC 9995, Section 3
`payload-location`	260	tstr	(none)	The string or URI hint for the location of the data hashed to produce the payload of a COSE_Sign1	RFC 9995, Section 3

Table 1: Newly Registered COSE Header Parameters

7. References

7.1. Normative References

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

[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>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

7.2. Informative References

[CoAP-Content-Formats] IANA, "CoAP Content-Formats", <https://www.iana.org/assignments/cose>.

[COSE-Algorithms] IANA, "COSE Algorithms", <https://www.iana.org/assignments/cose>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[SPDX] "SPDX Specification", <https://spdx.dev/use/specifications/>.

Acknowledgments

The following individuals provided input into the final form of the document: Carsten Bormann, Antoine Delignat-Lavaud, and Cedric Fournet.

Authors' Addresses

Orie Steele
Email: orie@or13.io

Steve Lasker
Email: stevenlasker@hotmail.com

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@ietf.contact