Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 24 April 2026

O. Steele

S. Lasker

H. Birkholz
Fraunhofer SIT

21 October 2025

COSE Hash Envelope
draft-ietf-cose-hash-envelope-09

Abstract

This document defines new 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 detached payload's content format and availability are defined, providing references to optional discovery mechanisms that can help to find original payload content.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://cose-wg.github.io/draft-ietf-cose-hash-envelope/draft-ietf-cose-hash-envelope.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-cose-hash-envelope/.

Discussion of this document takes place on the CBOR Object Signing and Encryption Working Group mailing list (mailto:cose@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/cose/. Subscribe at https://www.ietf.org/mailman/listinfo/cose/.

Source for this draft and an issue tracker can be found at https://github.com/cose-wg/draft-ietf-cose-hash-envelope.

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/.

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. Header Parameters 3
4. Hash Envelope CDDL 4
4.1. Envelope Extended Diagnostic Notation (RFC8610). 5
5. Security Considerations 6
5.1. Choice of Hash Function 6
5.2. COSE_Encrypt 6
5.3. Payload Verification 6
6. IANA Considerations 6
6.1. COSE Header Parameters 7
7. References 7
7.1. Normative References 7
7.2. Informative References 8
Appendix A. Implementation Status 9
A.1. Transmute Prototype 9
A.2. DataTrails Preview 10
A.3. DigiCert Preview 10
A.4. Microsoft CoseSignTool 11
Acknowledgments 11
Authors' Addresses 11

1. Introduction

COSE defined detached payloads in Section 2 of [RFC9052], 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) 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 (content type of the preimage of the payload) 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 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 which are hashed to produce the payload.

For example, when the actual content is a bstr, a Verifier appraising a content-type bstr has to decide if that bstr describes the digest bytes or the preimage bytes. Setting preimage-content-type to bstr, makes it clear that the preimage bytes themselves were a bstr.

4.1. Envelope Extended Diagnostic Notation (Appendix G of [RFC8610]).

The following informative example 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',
          # As seen in manifest.spdx.json.sha256
  / signature   / h'15280897...93ef39e5'
          # ECDSA Signature with SHA 384 and P-384
])
```

In this example, an [SPDX] software bill of materials (SBOM) in JSON format is already commonly identified by its SHA256 hash. For example, some tooling generates a file, such as manifest.spdx.json.sha256, which contains the SHA256 hash of the corresponding manifest.spdx.json file.

The content type for manifest.spdx.json is already well known as application/spdx+json, and is registered with IANA (https://www.iana.org/assignments/media-types/application/spdx+json).

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 SHA256 hash of the manifest.spdx.json, which is typically found in an adjacent file (manifest.spdx.json.sha256).

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 ECDSA with the SHA384 hash function and P-384 elliptic curve.

This example is chosen to highlight that an existing system may use a hash algorithm such as SHA256. 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 SHA384, 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 equal or stronger to that of 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 SHOULD be registered in the IANA COSE Algorithms registry, and should 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 the scope of this document.

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 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 is requested to add the COSE header parameters defined in Section 3, as listed in Table 1, to the "COSE Header Parameters" registry [IANA.cose_header-parameters], in the 'Integer values from 256 to 65535' range ('Specification Required' Registration Procedure).

Name	Label	Value Type	(1)	Description	Reference
payload-hash-alg	258	int	(2)	The hash algorithm used to produce the payload of a COSE_Sign1	RFCthis, Section 3
preimage-content-type	259	uint / tstr	(3)	The content-format number or content-type (media-type name) of data that has been hashed to produce the payload of the COSE_Sign1	RFCthis, 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	RFCthis, Section 3

Table 1: Newly registered COSE Header Parameters
(1): Value Registry
(2): https://www.iana.org/assignments/cose/cose.xhtml#algorithms
(3): https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats

7. References

7.1. Normative References

[IANA.cose_header-parameters]
IANA, "COSE Header Parameters",
<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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/rfc9110>.

7.2. Informative References

[BCP205] Best Current Practice 205,
<https://www.rfc-editor.org/info/bcp205>.
At the time of writing, this BCP comprises the following:

Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

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

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

Appendix A. Implementation Status

Note to RFC Editor: Please remove this section as well as references to [BCP205] before AUTH48.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [BCP205]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [BCP205], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

A.1. Transmute Prototype

Organization: Transmute Industries Inc

Name: https://github.com/transmute-industries/transmute

Description: A command line tool and GitHub action for securing software artifacts in GitHub workflows.

Maturity: Prototype

Coverage: The current version (’main’) implements this specification and demonstrates hash envelope signing with Azure Key Vault and Google Cloud KMS in addition to supporting local keys.

License: Apache-2.0

Implementation Experience: No interop testing has been done yet. The code works as proof of concept, but is not yet production ready.

Contact: Orie Steele (orie@or13.io)

A.2. DataTrails Preview

Organization: DataTrails

Name: https://github.com/datatrails/scitt-action

Description: A GitHub Action for registering statements about artifacts on a transparency service.

Maturity: Preview

Coverage: The current version ('main') implements this specification and demonstrates hash envelope signing with DataTrails implementation of SCITT.

License: MIT

Implementation Experience: Interop testing has been performed between DigiCert and DataTrails. The code works as proof of concept, but is not yet production ready.

Contact: Steve Lasker (stevenlasker@hotmail.com)

A.3. DigiCert Preview

Organization: DigiCert

Name: https://github.com/digicert/scitt-action

Description: A GitHub Action for remote signing and registering statements about artifacts on a transparency service.

Maturity: Preview

Coverage: The current version ('main') implements this specification and demonstrates hash envelope signing with DigiCert Software Trust Manager.

License: MIT

Implementation Experience: Interop testing has been performed between DigiCert and DataTrails. The code works as proof of concept, but is not yet production ready.

Contact: Corey Bonnell (Corey.Bonnell@digicert.com)

A.4. Microsoft CoseSignTool

Organization: Microsoft

Name: https://github.com/microsoft/CoseSignTool

Description: A platform-agnostic command line application to create and validate COSE signatures.

Maturity: This is an alpha release.

Coverage: The current version (1.6.5) implements this specification through the 'indirect-sign' and 'indirect-verify' plugins.

License: MIT

Implementation Experience: Tests are run with CDDL schema-validated inputs and outputs. No direct interoperability testing with other implementations has been performed so far.

Contact: The COSE Sign Tool team, via GitHub Issues (https://github.com/microsoft/CoseSignTool/issues)

Acknowledgments

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