Stream: Internet Engineering Task Force (IETF)
RFC: 9942
Category: Standards Track
Published: April 2026
ISSN: 2070-1721
Authors: O. Steele *Tradeverifyd*; H. Birkholz *Fraunhofer SIT*; A. Delignat-Lavaud *Microsoft*; C. Fournet *Microsoft*

RFC 9942
CBOR Object Signing and Encryption (COSE) Receipts

Abstract

CBOR Object Signing and Encryption (COSE) Receipts prove properties of a Verifiable Data Structure (VDS) to a verifier. Verifiable Data Structures and associated Proof Types enable security properties, such as minimal disclosure, transparency, and non-equivocation. Transparency helps maintain trust over time and has been applied to certificates, end-to-end encrypted messaging systems, and supply chain security. This specification enables concise transparency-oriented systems by building on Concise Binary Object Representation (CBOR) and COSE. The extensibility of the approach is demonstrated by providing CBOR encodings for Merkle inclusion and consistency proofs.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

COSE Receipts are signed proofs that include metadata about certain states of a Verifiable Data Structure (VDS) that are true when the COSE Receipt was issued. COSE Receipts can include proofs that a document is in a database (proof of inclusion), that a database is append-only (proof of consistency), that a smaller set of statements are contained in a large set of statements (proof of disclosure, a special case of proof of inclusion), or that certain data is not yet present in a database (proof of non-inclusion). Different VDSs can produce different Verifiable Data structure Proofs (VDP). The combination of representations of various VDSs and VDP can significantly increase the burden for implementers and create interoperability challenges for transparency services. This document describes how to convey VDS and associated VDP types in unified COSE envelopes.

1.1. Requirements Notation

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. New COSE Header Parameters

This document defines three new COSE header parameters, which are introduced up front in this section and elaborated on later in this document.

394: A COSE header parameter named "`receipts`" with a value type of array where the array contains one or more COSE Receipts as specified in this document.

395: A COSE header parameter named "vds" (for Verifiable Data Structure), which conveys the algorithm identifier for a Verifiable Data Structure. Correspondingly, see Section 8.2.2.1 for a registry defining the integers used to identify Verifiable Data Structures.

396: A COSE header parameter named "vdp" (for Verifiable Data Structure Proofs), which conveys a map containing Verifiable Data Structure Proofs organized by Proof Type. Correspondingly, see Section 8.2.2.2 for a registry defining the integers used to identify Verifiable Data Structure Proof Types.

3. Terminology

CDDL: Concise Data Definition Language (CDDL) is defined in [RFC8610].

EDN: CBOR Extended Diagnostic Notation (EDN) is defined in [RFC8949], where it is referred to as "diagnostic notation", and is revised in [CBOR-EDN].

Verifiable Data Structure (VDS): A data structure that supports one or more Verifiable Data Structure Proof Types. This property describes an algorithm used to maintain a Verifiable Data Structure, for example, a binary Merkle Tree algorithm.

Verifiable Data Structure Proofs (VDP): A data structure used to convey Proof Types for proving different properties, such as authentication, inclusion, consistency, and freshness. Parameters can include multiple proofs of a given type or multiple types of proof (inclusion and consistency).

Proof Type: A property that can be obtained by verifying a given proof over one or more entries in a Verifiable Data Structure. For example, a VDS, such as a binary Merkle Tree, can support inclusion proofs where each proof confirms that a given entry is included in a Merkle Tree root.

Proof Value: An encoding of a Proof Type in CBOR [RFC8949].

Entry: An entry in a Verifiable Data Structure for which proofs can be derived.

Receipt: A COSE Single Signer Data Object, as defined in [RFC9052], containing the header parameters necessary to convey one or more VDP for an associated VDS.

4. Verifiable Data Structures in CBOR

This section describes representations of Verifiable Data Structure Proofs in [RFC8949]. For example, construction of a Merkle Tree leaf or an inclusion proof from a leaf to a Merkle Tree root might have several different representations, depending on the Verifiable Data Structure used. Differences in representations are necessary to support efficient verification, unique security or privacy properties, and for compatibility with specific implementations. This document defines two extension points for enabling Verifiable Data Structures with COSE and provides concrete examples for the structures and proofs defined in Section 2.1.3 of [RFC9162] and Section 2.1.4 of [RFC9162]. The design of these structures is influenced by the conventions established for COSE Keys.

4.1. Structures

Similar to COSE Key Types [IANA.cose_header-parameters], different Verifiable Data Structures support different algorithms.

This document establishes a registry of Verifiable Data Structure algorithms; see Section 8.2.2.1 for details.

4.2. Proofs

Similar to COSE Key Type Parameters [IANA.cose_header-parameters], as EC2 keys (1: 2) require and give meaning to specific parameters, such as -1 (crv), -2 (x), -3 (y), -4 (d), RFC9162_SHA256 (395: 1) supports both (-1) inclusion and (-2) consistency proofs.

This document establishes a registry of Verifiable Data Structure Proofs; see Section 8.2.2.2 for details.

Proof Types are specific to their associated "Verifiable Data Structure"; for example, different Merkle Trees might support different representations of inclusion proof or consistency proof. Implementers should not expect interoperability across "Verifiable Data Structures". Security analysis **MUST** be conducted prior to migrating to new structures to ensure the new security and privacy assumptions are acceptable for the use case.

4.3. Usage

This document registers a new COSE header parameter "`receipts`" (394) to enable Receipts to be conveyed in the protected and unprotected headers of Enveloped COSE Structures.

When the "receipts" header parameter is present, the verifier **MUST** confirm that the associated Verifiable Data Structure and Verifiable Data Structure Proofs match entries present in the registries established in this specification, including values added in subsequent registrations.

Receipts **MUST** be tagged as COSE_Sign1.

The following definition from [RFC8610] is provided:

```
Signature_With_Receipt = /6.18(COSE_Sign1)

cose-label = int / text
cose-values = any

Protected_Header = {
  * cose-label => cose-values
}

Unprotected_Header = {
  &(receipts: 394)  => [+ bstr .cbor Receipt]
  * cose-label => cose-values
}

COSE_Sign1 = [
  protected   : bstr .cbor Protected_Header,
  unprotected : Unprotected_Header,
  payload     : bstr / nil,
  signature   : bstr
]

Receipt = Receipt_For_Inclusion / Receipt_For_Consistency

; Note the proof formats shown here are for RFC9162_SHA256.
; Other Verifiable Data Structures may have different proof formats.


Receipt_For_Inclusion = /6.18(Signed_Inclusion_Proof)

Signed_Inclusion_Proof = [
  protected   : bstr .cbor RFC9162_SHA256_Inclusion_Protected_Header,
  unprotected : RFC9162_SHA256_Inclusion_Unprotected_Header,
  payload     : bstr / nil,
  signature   : bstr
]

RFC9162_SHA256_Inclusion_Protected_Header = {
  &(alg: 1) => int
  &(vds: 395) => int
  * cose-label => cose-values
}

RFC9162_SHA256_Inclusion_Unprotected_Header = {
  &(vdp: 396) => RFC9162_SHA256_Verifiable_Inclusion_Proofs
  * cose-label => cose-values
}

RFC9162_SHA256_Verifiable_Inclusion_Proofs = {
  &(inclusion-proof: -1) => RFC9162_SHA256_Inclusion_Proofs
}

RFC9162_SHA256_Inclusion_Proofs = [ + RFC9162_SHA256_Inclusion_Proof ]

RFC9162_SHA256_Inclusion_Proof = bstr .cbor [
  tree_size: uint,
  leaf_index: uint,
  inclusion_path: [ + bstr ]
```

```
]


Receipt_For_Consistency = /6.18(Signed_Consistency_Proof)

Signed_Consistency_Proof = [
  protected   : bstr .cbor RFC9162_SHA256_Consistency_Protected_Header,
  unprotected : RFC9162_SHA256_Consistency_Unprotected_Header,
  payload     : bstr / nil, ; Newer Merkle Tree root
  signature   : bstr
]

RFC9162_SHA256_Consistency_Protected_Header = {
  &(alg: 1) => int,
  &(vds: 395) => int,
  * cose-label => cose-values
}

RFC9162_SHA256_Consistency_Unprotected_Header = {
  &(vdp: 396) => RFC9162_SHA256_Verifiable_Consistency_Proofs
  * cose-label => cose-values
}

RFC9162_SHA256_Verifiable_Consistency_Proofs = {
  &(consistency-proof: -2) => RFC9162_SHA256_Consistency_Proofs
}

RFC9162_SHA256_Consistency_Proofs = [ + RFC9162_SHA256_Consistency_Proof ]

RFC9162_SHA256_Consistency_Proof = bstr .cbor [
   tree_size_1: uint,
   tree_size_2: uint,
   consistency_path: [ + bstr ]
]
```

Figure 1: CDDL for a COSE_Sign1 with Attached Receipts

The following informative EDN is provided:

```
/ cose-sign1 / 18([
  / protected   / <<{
    / kid / 4 : h'bc297b51...e4edf0de',
    / algorithm / 1 : -7,  / ES256
  }>>,
  / unprotected / {
    / receipts / 394 : {         [ << ... >> ]
    }
    <</ cose-sign1 / 18([
        / protected   / <<{
          / kid / 4 : h'abcdef12...34567890',
          / algorithm / 1 : -7,  / ES256
          / vds       / 395 : 1, / RFC9162 SHA-256
        }>>,
        / unprotected / {
          / proofs / 396 : {
            / inclusion / -1 : [
              <<[
                / size / 9, / leaf / 8,
                / inclusion path /
                h'7558a95f...e02e35d6'
              ]>>
            ],
          },
        },
        / payload     / null,
        / signature   / h'02d227ed...ccd3774f'
      ])>>,
      <</ cose-sign1 / 18([
        / protected   / <<{
          / kid / 4 : h'abcdef12...34567890',
          / algorithm / 1 : -7,  / ES256
          / vds       / 395 : 1, / RFC9162 SHA-256
        }>>,
        / unprotected / {
          / proofs / 396 : {
            / inclusion / -1 : [
              <<[
                / size / 6, / leaf / 5,
                / inclusion path /
                [ h'9352f974...4ffa7ce0',
                  h'54806f32...f007ea06' ]
              ]>>
            ],
          },
        },
        / payload     / null,
        / signature   / h'36581f38...a5581960'
      ])>>
    },
  },
  / payload     / h'0167c57c...deeed6d4',
  / signature   / h'2544f2ed...5840893b'
])
```

Figure 2: An Example COSE Signature with Multiple Receipts

The specific structure of COSE Receipts is dependent on the structure of the COSE_Sign1 payload and the Verifiable Data Structure Proofs contained in the COSE_Sign1 unprotected header. The CDDL definition for Verifiable Data Structure Proofs is specific to each Verifiable Data Structure. This document describes proofs for RFC9162_SHA256 in the following sections.

4.4. Profiles

New Verifiable Data Structures can require the definition of a profile. The payload in such definitions **SHOULD** be detached. Detached payloads force verifiers to recompute the root from the proof and protect against implementation errors where the signature is verified but the payload is incompatible with the proof. Profiles of proof signatures that define additional protected header parameters are encouraged to make their presence mandatory to ensure that claims are processed with their intended semantics. One way to include this information in the COSE structure is use of the "typ" (type) header parameter; see [RFC9596] and the similar guidance provided in [RFC9597].

4.4.1. Registration Requirements

Each Verifiable Data Structure specification applying for inclusion in this registry **MUST** define how to encode the Verifiable Data Structure identifier and its Proof Types in CBOR. Each specification **MUST** define how to produce and consume the supported Proof Types. See Section 5 as an example.

Where a specification supports a choice of hash algorithm, a separate IANA registration must be made for each supported algorithm. For example, to provide support for SHA256 and SHA3_256 with Merkle inclusion proofs and Merkle consistency proofs defined, respectively, in Section 2.1.3 of [RFC9162] and Section 2.1.4 of [RFC9162], both "RFC9162_SHA256" and "RFC9162_SHA3_256" require entries in the relevant IANA registries. This document only defines "RFC9162_SHA256".

5. RFC9162_SHA256

This section defines how the data structure described in Section 2.1 of [RFC9162] is mapped to the terminology defined in this document, using [RFC8949] and [RFC9053].

5.1. Verifiable Data Structure

The integer identifier for this Verifiable Data Structure is 1. The string identifier for this Verifiable Data Structure is "RFC9162_SHA256", a Merkle Tree where SHA256 is used as the hash algorithm (see Table 2). See Section 2.1.1 of [RFC9162] for a complete description of this Verifiable Data Structure.

5.2. Inclusion Proof

See Section 2.1.3.1 of [RFC9162] for a complete description of this Verifiable Data Structure Proof Type.

The CBOR representation of an inclusion proof for RFC9162_SHA256 is:

```
inclusion-proof = bstr .cbor [

    ; tree size at current Merkle Tree root
    tree-size: uint

    ; index of leaf in tree
    leaf-index: uint

    ; path from leaf to current Merkle Tree root
    inclusion-path: [ + bstr ]
]
```

Figure 3: CBOR-Encoded Inclusion Proof for RFC9162_SHA256

The term `leaf-index` is used for alignment with the use established in Section 2.1.3.2 of [RFC9162].

Note that [RFC9162] defines inclusion proofs only for leaf nodes, and that:

> If leaf_index is greater than or equal to tree_size, then fail the proof verification.

The identifying index of a leaf node is relative to all nodes in the tree size for which the proof was obtained.

5.2.1. Receipt of Inclusion

In a signed proof, the payload is the Merkle Tree root that corresponds to the log at size `tree-size`. The protected header for an RFC9162_SHA256 inclusion proof signature is:

```
protected-header-map = {
  &(alg: 1) => int
  &(vds: 395) => int
  * cose-label => cose-value
}
```

Figure 4: Protected Header for a Receipt of Inclusion

alg (label: 1): **REQUIRED**. Signature algorithm identifier. Value type: int.

vds (label: 395): **REQUIRED**. Verifiable Data Structure algorithm identifier. Value type: int.

The unprotected header for an RFC9162_SHA256 inclusion proof signature is:

```
inclusion-proofs = [ + inclusion-proof ]

verifiable-proofs = {
  &(inclusion-proof: -1) => inclusion-proofs
}

unprotected-header-map = {
  &(vdp: 396) => verifiable-proofs
  * cose-label => cose-value
}
```

Figure 5: A Verifiable Data Structure Proofs in an Unprotected Header

vdp (label: 396): **REQUIRED**. Verifiable Data Structure Proofs. Value type: Map.

inclusion-proof (label: -1): **REQUIRED**. Inclusion proofs. Value type: Array of bstr.

The payload of an RFC9162_SHA256 inclusion proof signature is the Merkle Tree hash as defined in [RFC9162].

An EDN example for a Receipt containing an inclusion proof for RFC9162_SHA256 with a detached payload (see Section 4.4) is:

```
/ cose-sign1 / 18([
  / protected   / <<{
    / algorithm / 1 : -7,  / ES256
    / vds       / 395 : 1, / RFC9162 SHA-256
  }>>,
  / unprotected / {
    / proofs / 396 : {
      / inclusion / -1 : [
        <<[
          / size / 20, / leaf / 17,
          / inclusion path /
          [ h'fc9f050f...221c92cb',
            h'bd0136ad...6b28cf21',
            h'd68af9d6...93b1632b' ]
        ]>>
      ],
    },
  },
  / payload     / null,
  / signature   / h'de24f0cc...9a5ade89'
])
```

Figure 6: Receipt of Inclusion

The VDS in the protected header is necessary to understand the inclusion proof structure in the unprotected header.

The inclusion proof and signature are verified in order. First, the verifier applies the inclusion proof to a possible entry (set member) bytes. If this process fails, the inclusion proof may have been tampered with. If this process succeeds, the result is a Merkle Tree root, which in the attached as the COSE_Sign1 payload. Second, the verifier checks the signature of the COSE_Sign1. If the resulting signature can be verified, the Receipt has proved inclusion of the entry in the Verifiable Data Structure. If the resulting signature cannot be verified, the signature may have been tampered with.

5.3. Consistency Proof

See Section 2.1.4.1 of [RFC9162] for a complete description of this Verifiable Data Structure Proof Type.

The cbor representation of a consistency proof for RFC9162_SHA256 is:

```
consistency-proof =  bstr .cbor [

    ; older Merkle Tree size
    tree-size-1: uint

    ; newer Merkle Tree size
    tree-size-2: uint

    ; path from older Merkle Tree to newer Merkle Tree
    consistency-path: [ + bstr ]

]
```

Figure 7: CBOR-Encoded Consistency Proof for RFC9162_SHA256

5.3.1. Receipt of Consistency

In a signed consistency proof, the newer Merkle Tree root (proven to be consistent with an older Merkle Tree root) is a detached payload and corresponds to the log at size tree-size-2.

The protected header for an RFC9162_SHA256 consistency proof signature is:

```
protected-header-map = {
  &(alg: 1) => int
  &(vds: 395) => int
  * cose-label => cose-value
}
```

Figure 8: Protected Header for a Receipt of Consistency

alg (label: 1): **REQUIRED**. Signature algorithm identifier. Value type: int.

vds (label: 395): **REQUIRED**. Verifiable Data Structure algorithm identifier. Value type: int.

The unprotected header for an RFC9162_SHA256 consistency proof signature is:

```
consistency-proofs = [ + consistency-proof ]

verifiable-proofs = {
  &(consistency-proof: -2) => consistency-proofs
}

unprotected-header-map = {
  &(vdp: 396) => verifiable-proofs
  * cose-label => cose-value
}
```

vdp (label: 396): **REQUIRED**. Verifiable Data Structure Proofs. Value type: Map.

consistency-proof (label: -2): **REQUIRED**. Consistency proofs. Value type: Array of bstr.

The payload of an RFC9162_SHA256 consistency proof signature is: The newer Merkle Tree hash as defined in [RFC9162].

An EDN example for a Receipt containing a consistency proof for RFC9162_SHA256 with a detached payload (see Section 4.4) is:

```
/ cose-sign1 / 18([
  / protected   / <<{
    / algorithm / 1 : -7,  / ES256
    / vds       / 395 : 1, / RFC9162 SHA-256
  }>>,
  / unprotected / {
    / proofs / 396 : {
      / consistency / -2 : [
        <<[
          / old / 20, / new / 104,
          / consistency path /
          h'e5b3e764...c4a813bc',
          h'87e8a084...4f529f69',
          h'f712f76d...92a0ff36',
          h'd68af9d6...93b1632b',
          h'249efab6...b7614ccd',
          h'85dd6293...38914dc1'
        ]>>
      ],
    },
  },
  / payload     / null,
  / signature   / h'94469f73...52de67a1'
])
```

Figure 9: Example Consistency Receipt

The VDS in the protected header is necessary to understand the consistency proof structure in the unprotected header.

The signature and consistency proof are verified in order.

First, the verifier checks the signature on the COSE_Sign1. If the verification fails, the consistency proof is not checked. Second, the consistency proof is checked by applying a previous inclusion proof to the consistency proof. If the verification fails, the append-only property of the Verifiable Data Structure is not assured. This approach is specific to RFC9162_SHA256; different Verifiable Data Structures may not support consistency proofs. It is recommended that implementations return a single boolean result for Receipt-verification operations to reduce the chance of accepting a valid signature over an invalid consistency proof.

6. Privacy Considerations

The privacy considerations section of [RFC9162] and [RFC9053] apply to this document.

6.1. Log Length

Some structures and proofs leak the size of the log at the time of inclusion. In the case that a log only stores certain kinds of information, this can reveal details that could impact reputation. For example, if a transparency log only stored breach notices, a receipt for a breach notice would reveal the number of previous breaches at the time the notice was made transparent.

6.2. Header Parameters

Additional header parameters can reveal information about the transparency service or its log entries. The receipt producer **MUST** perform a privacy analysis for all mandatory fields in profiles based on this specification.

7. Security Considerations

See the Security Considerations sections of:

- [RFC9162]
- [RFC9053]

7.1. Choice of Signature Algorithms

A security analysis ought to be performed to ensure that the digital signature algorithm `alg` has the appropriate strength to secure receipts.

It is recommended to select signature algorithms that share cryptographic components with the Verifiable Data Structure used; for example, both RFC9162_SHA256 and ES256 depend on the sha-256 hash function.

7.2. Validity Period

In some cases, receipts **MAY** include strict validity periods, for example, activation not too far in the future or expiration not too far in the past. See the `iat`, `nbf`, and `exp` claims in [RFC8392] for one way to accomplish this. The details of expressing validity periods are out of scope for this document.

7.3. Status Updates

In some cases, receipts should be "revocable" or "suspendable" after being issued, regardless of their validity period. The details of expressing statuses are out of scope for this document.

8. IANA Considerations

8.1. COSE Header Parameter

IANA has added the COSE header parameters defined in Section 2, and as listed in Table 1, to the "COSE Header Parameters" subregistry [IANA.cose_header-parameters] in the "CBOR Object Signing and Encryption (COSE)" registry group. These COSE header parameters fall in the 'Integer values from 256 to 65535' range (with a Specification Required registration procedure (see [RFC8126])). The Value Registry listed for "vds" is the "COSE Verifiable Data Structure Algorithm" subregistry. The map labels in the "vdp" are assigned from the "COSE Verifiable Data Structure Proofs" subregistry.

Name	Label	Value Type	Value Registry	Description	Reference
`receipts`	394	array		Priority ordered sequence of CBOR encoded Receipts	RFC 9942, Section 2
`vds`	395	int	COSE Verifiable Data Structure	Algorithm identifier for Verifiable Data Structures that is used to produce Verifiable Data Structure Proofs	RFC 9942, Section 2
`vdp`	396	map	map key in COSE Verifiable Data Structure Proofs	Location for Verifiable Data Structure Proofs in COSE Header Parameters	RFC 9942, Section 2

Table 1: Newly Registered COSE Header Parameters

8.2. Verifiable Data Structure Registries

IANA established the "COSE Verifiable Data Structure Algorithms" and "COSE Verifiable Data Structure Proofs" subregistries under a Specification Required policy as described in Section 4.6 of [RFC8126].

8.2.1. Expert Review

Expert reviewers (see [RFC8126]) should take into consideration the following points:

- Experts are advised to assign the next available positive integer for Verifiable Data Structures.
- Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments.
- Specifications are required for all point assignments. early allocation is permissible, see Section 2 of [RFC7120].
- It is not permissible to assign points in COSE Verifiable Data Structure algorithms for which no corresponding COSE Verifiable Data Structure Proofs entry exists, and vice versa.
- The change controller for related registrations of structures and proofs should be the same.

8.2.2. Templates and Initial Contents

8.2.2.1. COSE Verifiable Data Structure Algorithms Registry

Registration Template:

Name:
: This is a descriptive name for the Verifiable Data Structure that enables easier reference to the item.

Value:
: This is the value used to identify the Verifiable Data Structure.

Description:
: This field contains a brief description of the Verifiable Data Structure.

Reference:
: This contains a pointer to the public specification for the Verifiable Data Structure.

Change Controller:
: For Standards Track RFCs, list the "IETF". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Name	Value	Description	Change Controller	Reference
Reserved	0	Reserved		RFC 9942
RFC9162_SHA256	1	SHA256 Binary Merkle Tree	IETF	Section 2.1 of [RFC9162]

Table 2: COSE Verifiable Data Structure Algorithms Initial Registry Contents

8.2.2.2. COSE Verifiable Data Structure Proofs Registry

Registration Template:

Verifiable Data Structure:
: This value used identifies the related Verifiable Data Structure.

Name:
: This is a descriptive name for the Proof Type that enables easier reference to the item.

Label:
: This is the value used to identify the Verifiable Data Structure Proof Type.

CBOR Type:
: This contains the CBOR type for the value portion of the label.

Description:
: This field contains a brief description of the Proof Type.

Reference:
: This contains a pointer to the public specification for the Proof Type.

Change Controller:
: For Standards Track RFCs, list the "IETF". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Verifiable Data Structure	Name	Label	CBOR Type	Description	Change Controller	Reference
1	inclusion proofs	-1	array (of bstr)	Proof of inclusion	IETF	RFC 9942, Section 5.2
1	consistency proofs	-2	array (of bstr)	Proof of append-only property	IETF	RFC 9942, Section 5.3

Table 3: COSE Verifiable Data Structure Proofs Initial Registry Contents

9. References

9.1. 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/info/rfc2119>.

[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>.

[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/info/rfc8949>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/info/rfc9053>.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://www.rfc-editor.org/info/rfc9162>.

[RFC9596] Jones, M.B. and O. Steele, "CBOR Object Signing and Encryption (COSE) "typ" (type) Header Parameter", RFC 9596, DOI 10.17487/RFC9596, June 2024, <https://www.rfc-editor.org/info/rfc9596>.

[RFC9597] Looker, T. and M.B. Jones, "CBOR Web Token (CWT) Claims in COSE Headers", RFC 9597, DOI 10.17487/RFC9597, June 2024, <https://www.rfc-editor.org/info/rfc9597>.

9.2. Informative References

[CBOR-EDN] Bormann, C., "CBOR Extended Diagnostic Notation (EDN)", Work in Progress, Internet-Draft, draft-ietf-cbor-edn-literals-21, 30 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-edn-literals-21>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[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>.

[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>.

[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>.

Acknowledgements

We would like to thank Maik Riechert, Jon Geater, Michael B. Jones, Mike Prorock, Ilari Liusvaara, and Amaury Chamayou for their contributions (some of which substantial) to this document and to the initial set of implementations.

Contributors

Amaury Chamayou
Microsoft
United Kingdom
Email: amaury.chamayou@microsoft.com

Steve Lasker
Email: stevenlasker@hotmail.com

Robert Martin
MITRE Corporation
United States of America
Email: ramartin@mitre.org

Monty Wiseman
United States of America
Email: mwiseman32@acm.org

Roy Williams
United States of America
Email: roywill@msn.com

Authors' Addresses

Orie Steele
Tradeverifyd
United States of America
Email: orie@or13.io

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

Antoine Delignat-Lavaud
Microsoft
United Kingdom
Email: antdl@microsoft.com

Cédric Fournet
Microsoft
United Kingdom
Email: fournet@microsoft.com