

Secure Patterns for Internet CrEentials
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

M. Prorock
mesur.io
O. Steele
Tradeverifyd
H. Birkholz
Fraunhofer SIT
R. Mahy
2 March 2026

Selective Disclosure CBOR Web Tokens (SD-CWT)
draft-ietf-spice-sd-cwt-07

Abstract

This specification describes a data minimization technique for use with CBOR Web Tokens (CWTs). The approach is inspired by the Selective Disclosure JSON Web Token (SD-JWT), with changes to align with CBOR Object Signing and Encryption (COSE) and CWTs.

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://ietf-wg-spice.github.io/draft-ietf-spice-sd-cwt/draft-ietf-spice-sd-cwt.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-spice-sd-cwt/>.

Discussion of this document takes place on the Secure Patterns for Internet CrEentials Working Group mailing list (<mailto:spice@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/spice/>. Subscribe at <https://www.ietf.org/mailman/listinfo/spice/>.

Source for this draft and an issue tracker can be found at <https://github.com/ietf-wg-spice/draft-ietf-spice-sd-cwt>.

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 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
1.1. High-Level Flow	5
2. Terminology	6
3. Overview of Selective Disclosure CWT	7
3.1. A CWT without Selective Disclosure	10
3.2. Holder gets an SD-CWT from the Issuer	11
4. Holder prepares an SD-CWT for a Verifier	15
5. SD-CWT Definition	16
5.1. Types of Blinded Claims	17
6. Differences from the CBOR Web Token Specification	19
6.1. Definite Length CBOR Required	19
6.2. Finite values for standard date claims	19
6.3. Allowed types of CBOR map keys	20
6.4. Duplicate map key detection	23
6.5. Level of Nesting of Claims	24
6.6. Use of Structured Suffixes	26
7. SD-CWT Issuance	26
7.1. Issuer Generation	28
7.2. Holder Validation	28
8. SD-CWT Presentation	30
8.1. Creating a Key Binding Token	31
9. SD-KBT and SD-CWT Verifier Validation	34
10. Decoy Digests	36
11. Tags Used Before SD-CWT Issuance	38
11.1. To Be Redacted Tag Definition	38

11.2. To Be Decoy	39
12. Encrypted Disclosures	39
12.1. AEAD Encrypted Disclosures Mechanism	40
13. Credential Types	42
14. Examples	42
14.1. Minimal Spanning Example	42
14.2. Nested Example	44
15. Privacy Considerations	48
15.1. Correlation	48
15.2. Determinism	49
15.3. Audience	49
15.4. Credential Types	49
16. Security Considerations	51
16.1. Issuer Key Compromise	51
16.2. Disclosure Coercion and Over-identification	51
16.3. Disclosure of Decoys	52
16.4. Random Numbers	52
16.5. Binding the KBT and the CWT	52
16.6. Covert Channels	53
16.7. Nested Disclosure Ordering	53
16.8. Choice of AEAD algorithms	54
17. IANA Considerations	54
17.1. COSE Header Parameters	54
17.1.1. sd_claims	54
17.1.2. sd_alg	55
17.1.3. sd_aead_encrypted_claims	55
17.1.4. sd_aead	55
17.2. CBOR Simple Values	56
17.3. CBOR Tags	56
17.3.1. To Be Redacted Tag	56
17.3.2. Redacted Claim Element Tag	56
17.3.3. To Be Decoy Tag	57
17.4. CBOR Web Token (CWT) Claims	57
17.4.1. vct	57
17.5. Media Types	57
17.5.1. application/sd-cwt	58
17.5.2. application/kb+cwt	59
17.6. Structured Syntax Suffix	60
17.7. Content-Formats	60
17.8. Verifiable Credential Type Identifiers	61
17.8.1. Registration Template	62
17.8.2. Initial Registry Contents	63
18. References	63
18.1. Normative References	63
18.2. Informative References	64
Appendix A. Complete CDDL Schema	66
Appendix B. Comparison to SD-JWT	70
B.1. Media Types	70

B.2. Redaction Claims	70
B.3. Issuance	71
B.4. Presentation	71
B.5. Validation	71
Appendix C. Keys Used in the Examples	71
C.1. Subject / Holder	71
C.2. Issuer	73
Appendix D. Implementation Status	75
D.1. Transmute Prototype	75
D.2. Rust Prototype	76
D.3. Python Prototype	76
Appendix E. Relationship between RATS Architecture and Verifiable Credentials	77
E.1. Three-Party Verifiable Credentials Model	77
E.2. RATS Architecture Roles	77
E.3. Role Mappings in the Three-Party Model	78
E.3.1. Verifiable Credential Issuer as RATS Endorser	78
E.3.2. Verifiable Credential Holder as RATS Verifier	78
E.3.3. Verifiable Credential Verifier as RATS Relying Party	79
E.3.4. All Parties Can Be Attesters	79
E.4. Comparison with RATS Interaction Models	80
E.5. Roles That Don't Map to the Three-Party Model	80
E.6. Application to SD-CWT	81
Appendix F. Sample Disclosure Matching Algorithm for Verifier	81
Appendix G. Document History	82
G.1. draft-ietf-spice-sd-cwt-06	82
G.2. draft-ietf-spice-sd-cwt-05	83
G.3. draft-ietf-spice-sd-cwt-04	84
G.4. draft-ietf-spice-sd-cwt-03	85
G.5. draft-ietf-spice-sd-cwt-02	85
G.6. draft-ietf-spice-sd-cwt-01	86
G.7. draft-ietf-spice-sd-cwt-00	86
Acknowledgments	86
Contributors	87
Authors' Addresses	87

1. Introduction

This specification creates a new format based on the CBOR Web Token (CWT) specification [RFC8392]. It enables the Holder of a CWT to disclose or withhold special claims marked as selectively disclosable by the Issuer of a CWT, when presenting those claims to a Verifier.

The approach is inspired by SD-JWT

[I-D.draft-ietf-oauth-selective-disclosure-jwt], with changes to align with conventions from CBOR Object Signing and Encryption (COSE) [RFC9052] and CWT. Holders of SD-CWT credentials can prove the integrity and authenticity of Holder-chosen attributes asserted by an

Issuer to a Verifier. The Holder also proves possession of the confirmation method (defined in [RFC8747]) to prevent copy and paste attacks.

SD-CWTs provide privacy improvements compared to regular CWTs, but remain traceable. Techniques such as one-time use and batch issuance can improve the confidentiality and security characteristics of CWT-based credential protocols.

This document defines a generic container format, not a specific credential type. For example, a license to operate a vehicle and a license to import a product will contain different attributes.

1.1. High-Level Flow

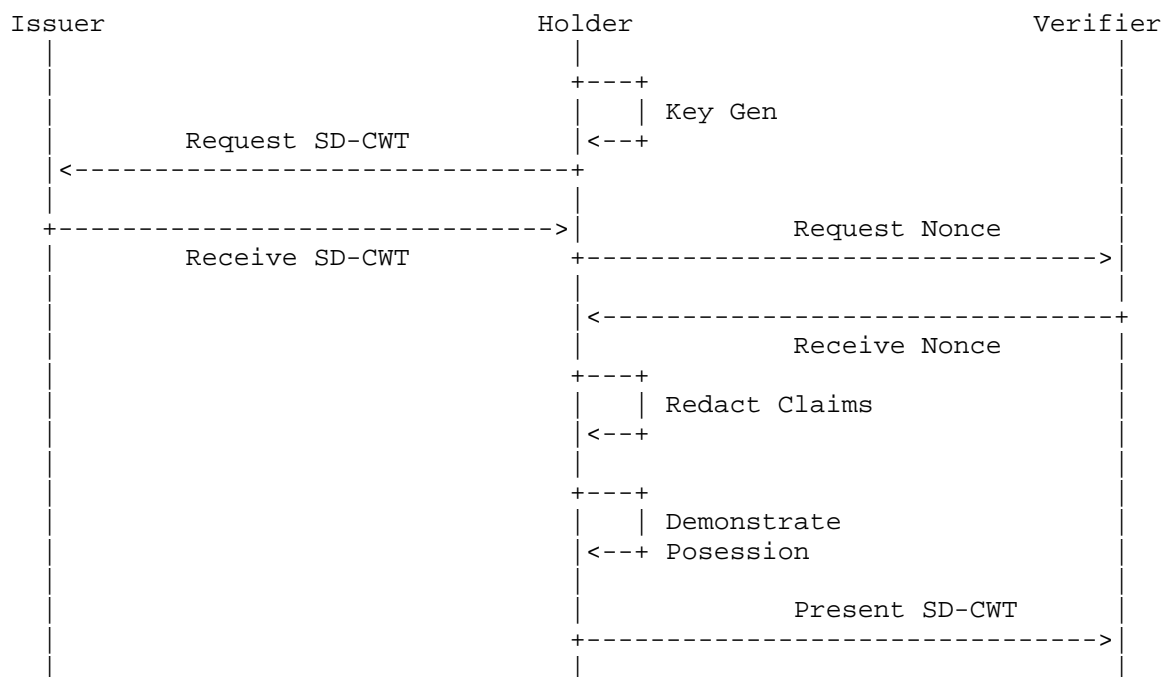


Figure 1: High-level SD-CWT Issuance and Presentation Flow

This diagram captures the flow to issue and present an SD-CWT.

The parameters necessary to support these processes can be obtained using transports or protocols that are out of scope for this specification.

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 CBOR examples in this document are shown in CBOR Extended Diagnostic Notation (EDN) [I-D.ietf-cbor-edn-literals]. EDN resembles JSON, but it can contain comments, and it can represent all native CBOR types, including byte strings, tagged values, and CBOR simple values.

Data model constraints are described using the Concise Data Definition Language (CDDL) [RFC8610].

This specification uses terms from CWT [RFC8392], COSE [RFC9052] [RFC9053] and JWT [RFC7519].

The terms Claim Name, Claim Key, and Claim Value are defined in [RFC8392].

This specification defines the following new terms:

Selective Disclosure CBOR Web Token (SD-CWT): A CWT with claims enabling selective disclosure with key binding.

Selective Disclosure Key Binding Token (SD-CWT-KBT): A CWT used to demonstrate possession of a confirmation method, associated with an SD-CWT.

Assertion Key: A key used by the Issuer to sign a Claim Values.

Confirmation Key: A key used by the Holder to sign a Selected Salted Disclosed Claims.

Issuer: An entity that produces a Selective Disclosure CBOR Web Token by signing a Claim Values with an Assertion Key.

Holder: An entity that presents a Selective Disclosure Key Binding Token, containing a Selective Disclosure CBOR Web Token and Selected Salted Disclosed Claims signed with a Confirmation Key.

Verifier: An entity that validates a Partial or Full Disclosure by a Holder.

Partial Disclosure: When a subset of the original claims, protected

by the Issuer, are disclosed by the Holder.

Full Disclosure: When the full set of claims protected by the Issuer is disclosed by the Holder. An SD-CWT with no blinded claims (when no claims are blinded by the Issuer) is considered a Full Disclosure.

Salted Disclosed Claim: A salted claim disclosed in the unprotected header of an SD-CWT.

Blinded Claim Hash: A hash digest of a Salted Disclosed Claim.

Blinded Claim: Any Redacted Claim Key or Redacted Claim Element that has been replaced in the CWT payload by a Blinded Claim Hash.

Redacted Claim Key: The hash of a claim redacted from a map data structure.

Redacted Claim Element: The hash of an element redacted from an array data structure.

Presented Disclosed Claims Set: The CBOR map containing zero or more Redacted Claim Keys or Redacted Claim Elements.

Validated Disclosed Claims Set: The CBOR map containing all mandatory to disclose claims signed by the Issuer, all selectively disclosed claims presented by the Holder, and omitting all undisclosed instances of Redacted Claim Keys and Redacted Claim Element claims that are present in the original SD-CWT.

3. Overview of Selective Disclosure CWT

SD-CWT operates on CWT Claims Sets as described in [RFC8392]. CWT Claims Sets contain Claim Keys and Claim Values. Issuers choose which Claim Keys and Claim Values to blind or not blind. Holders choose to disclose none, some, or all of the blinded Claim Keys and Claim Values, and whether to present an issued SD-CWT at all. Holders present an SD-CWT and any disclosures to Verifiers in a Key Binding Token (KBT) that proves the Holder's control of the private key corresponding to the SD-CWT confirmation (public) key.

Selective Disclosure CBOR Web Tokens (SD-CWTs) can be deployed in protocols that are already using CWTs with minor changes, even if they contain no optional-to-disclose claims. A Verifier that does not understand selective disclosure at all can only act on unblinded claims sent by the Holder; it will ignore Blinded Claims representing array items, and will fail to process any SD-CWT containing Blinded Claims that represent map keys. Optional Claim Keys, whether they

are disclosed or not, can only be processed by a Verifier that understands this specification. However, Claim Keys and Claim Values that are not understood remain ignored, as described in Section 3 of [RFC8392].

The following diagram explains the relationships between the terminology used in this specification.

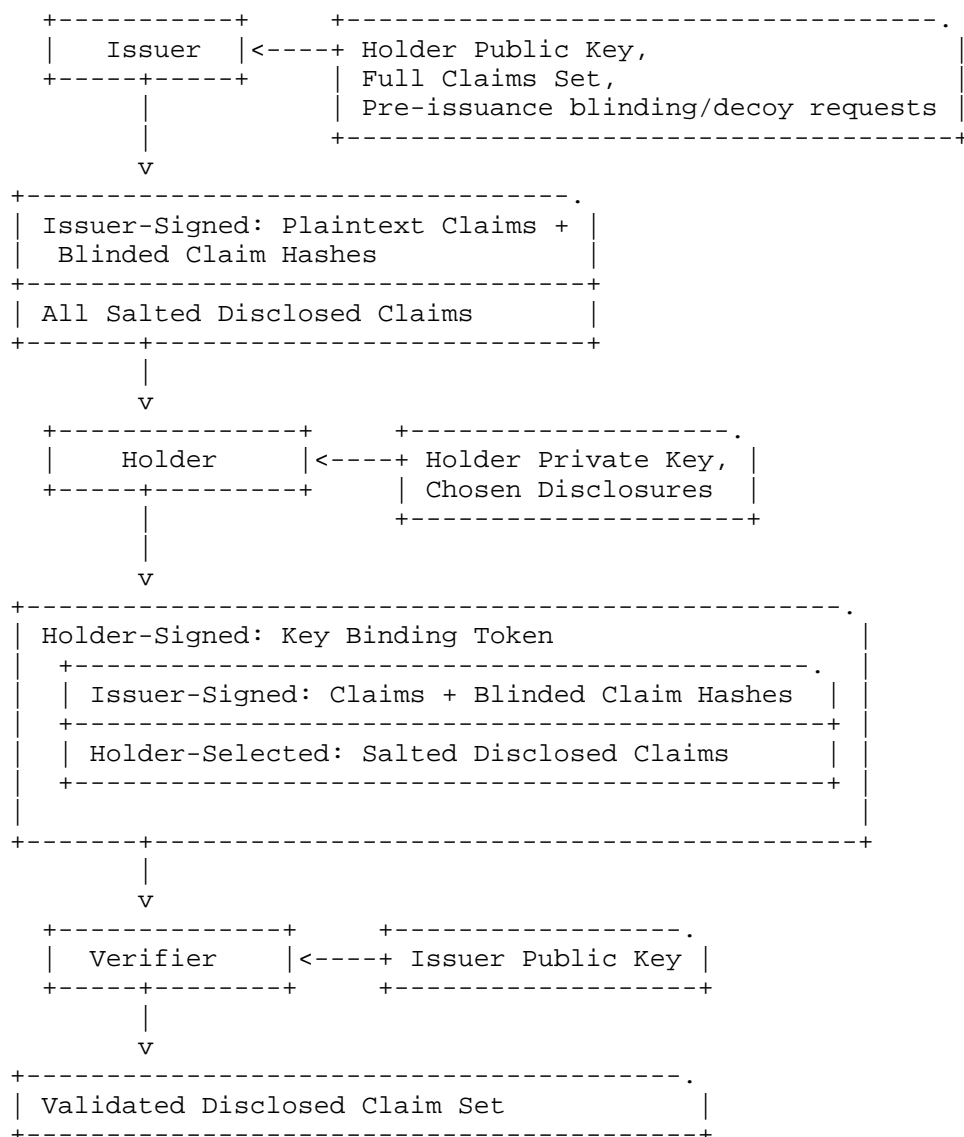
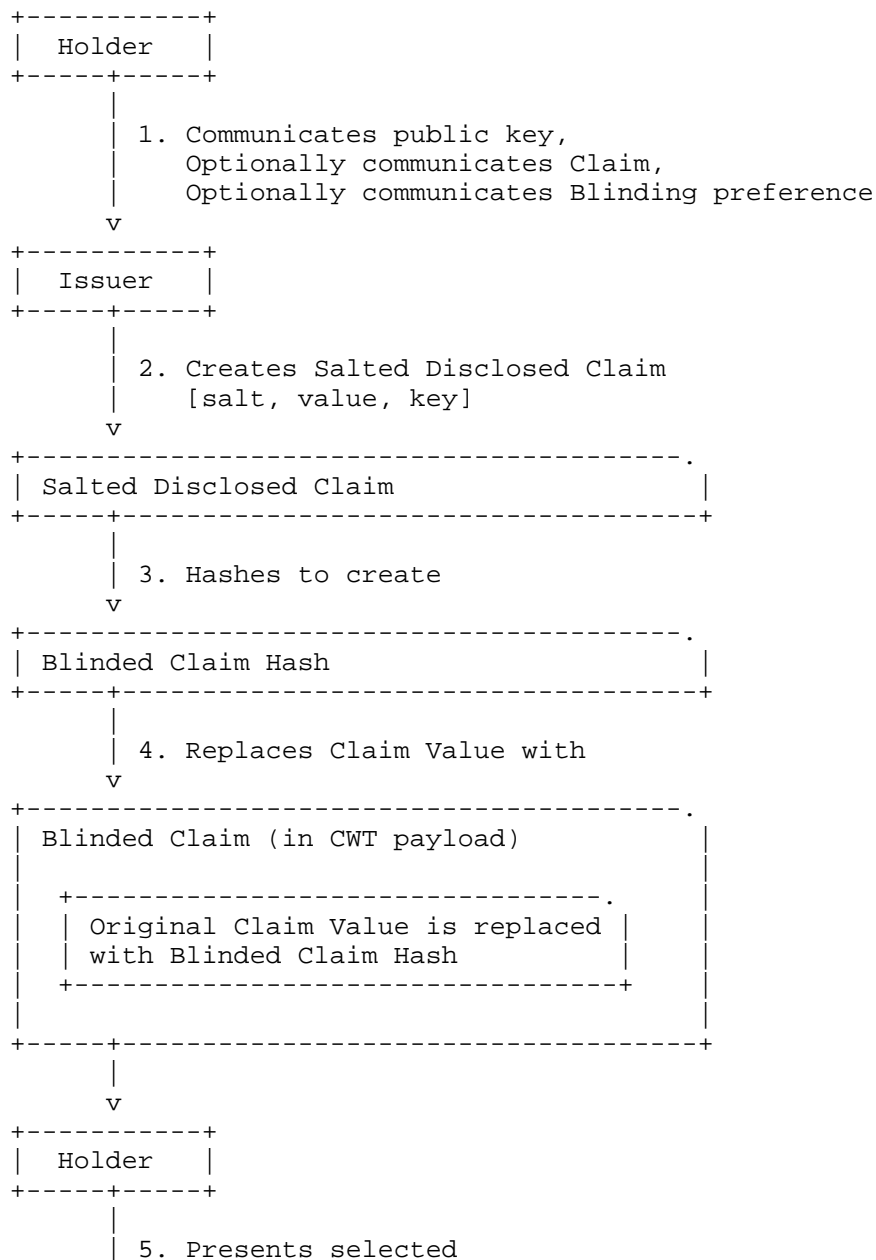


Figure 2: Inputs provided to each Role

This diagram relates the terminology specific to selective disclosure and redaction.



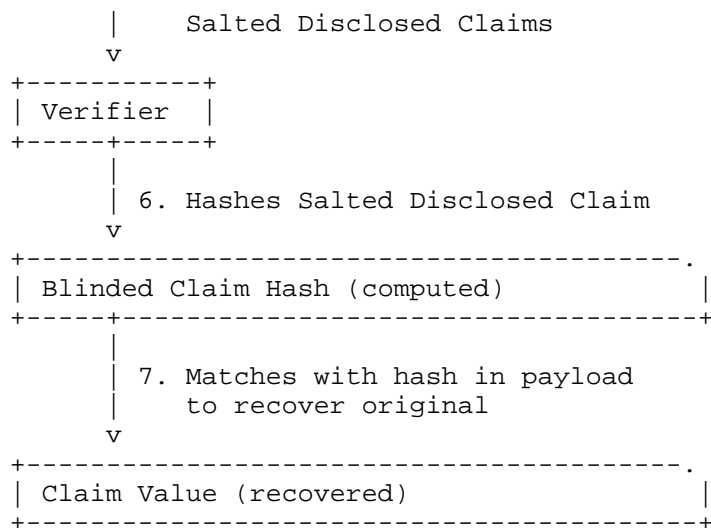


Figure 3: Round trip journey of a blinded claim

3.1. A CWT without Selective Disclosure

Below is the payload, or claims set, of a standard CWT not using selective disclosure. It consists of standard CWT claims, the Holder confirmation key, and five fictitious example claims. The payload is shown below in EDN [I-D.ietf-cbor-edn-literals]. Note that the fictitious map keys shown in the examples do not have IANA registered integer keys.

```

{
  / iss / 1 : "https://issuer.example",
  / sub / 2 : "https://device.example",
  / exp / 4 : 1725330600, /2024-09-02T19:30:00Z/
  / nbf / 5 : 1725243840, /2024-09-01T19:25:00Z/
  / iat / 6 : 1725244200, /2024-09-01T19:30:00Z/
  / cnf / 8 : {
    / cose key / 1 : {
      / kty / 1: 2, / EC2 /
      / crv / -1: 1, / P-256 /
      / x / -2: h'8554eb275dcd6fbd1c7ac641aa2c90d9
        2022fd0d3024b5af18c7cc61ad527a2d',
      / y / -3: h'4dc7ae2c677e96d0cc82597655ce92d5
        503f54293d87875d1e79ce4770194343'
    }
  },
  /most_recent_inspection_passed/ 500: true,
  /inspector_license_number/ 501: "ABCD-123456",
  /inspection_dates/ 502 : [
    1549560720, / 2019-02-07T17:32:00 /
    1612498440, / 2021-02-04T20:14:00 /
    1674004740, / 2023-01-17T17:19:00 /
  ],
  /inspection_location/ 503: {
    "country": "us", / United States /
    "region": "ca", / California /
    "postal_code": "94188"
  }
}

```

Figure 4: CWT Claims Set Without Selective Disclosure

The fictitious claims deal with attributes of an inspection of the subject: the pass/fail result, the inspection location, the license number of the inspector, and a list of dates when the subject was inspected.

3.2. Holder gets an SD-CWT from the Issuer

Alice would like to selectively disclose some of these claims to different Verifiers. Note that some of the claims may not be selectively disclosable. In our next example, the pass/fail status of the inspection, the most recent inspection date, and the country of the inspection will be claims that are always present in the SD-CWT. After the Holder requests an SD-CWT from the Issuer, the Issuer generates the following SD-CWT:

```

/ cose-sign1 / 18([ / issuer SD-CWT /
/ CWT protected / << {
  / alg / 1 : -35, / ES384 /
  / kid / 4 : 'https://issuer.example/cose-key3',
  / typ / 16 : 293, # application/sd-cwt
  / sd_alg / 170 : -16 / SHA256 /
} >>,
/ CWT unprotected / {
  / sd_claims / 17 : [ / these are all the disclosures /
    <<[
      /salt/ h'bae611067bb823486797dalebbb52f83',
      /value/ "ABCD-123456",
      /claim/ 501 / inspector_license_number /
    ]>>,
    <<[
      /salt/ h'8de86a012b3043ae6e4457b9e1aaab80',
      /value/ 1549560720 / inspected 7-Feb-2019 /
    ]>>,
    <<[
      /salt/ h'7af7084b50badeb57d49ea34627c7a52',
      /value/ 1612560720 / inspected 4-Feb-2021 /
    ]>>,
    <<[
      /salt/ h'ec615c3035d5a4ff2f5ae29ded683c8e',
      /value/ "ca",
      /claim/ "region" / region=California /
    ]>>,
    <<[
      /salt/ h'37c23d4ec4db0806601e6b6dc6670df9',
      /value/ "94188",
      /claim/ "postal_code"
    ]>>,
  ]
},
/ CWT payload / << {
  / iss / 1 : "https://issuer.example",
  / sub / 2 : "https://device.example",
  / exp / 4 : 1725330600, /2024-09-03T02:30:00+00:00Z/
  / nbf / 5 : 1725243900, /2024-09-02T02:25:00+00:00Z/
  / iat / 6 : 1725244200, /2024-09-02T02:30:00+00:00Z/
  / cnf / 8 : {
    / cose key / 1 : {
      / kty / 1: 2, / EC2 /
      / crv / -1: 1, / P-256 /
      / x / -2: h'8554eb275dcd6fbd1c7ac641aa2c90d9
        2022fd0d3024b5af18c7cc61ad527a2d',
      / y / -3: h'4dc7ae2c677e96d0cc82597655ce92d5
        503f54293d87875d1e79ce4770194343'
    }
  }
}

```

```

    }
  },
  /most_recent_inspection_passed/ 500: true,
  /inspection_dates/ 502 : [
    / redacted inspection date 7-Feb-2019 /
    60(h'1b7fc8ecf4b1290712497d226c04b503
      b4aa126c603c83b75d2679c3c613f3fd'),
    / redacted inspection date 4-Feb-2021 /
    60(h'64afccd3ad52da405329ad935de1fb36
      814ec48fd79e3a108ef858e291e146'),
    1674004740, / 2023-01-17T17:19:00 /
  ],
  / inspection_location / 503 : {
    "country" : "us", / United States /
    / redacted_claim_keys / simple(59) : [
      / redacted region /
      h'0d4b8c6123f287a1698ff2db15764564
        a976fb742606e8fd00e2140656ba0df3'
      / redacted postal_code /
      h'c0b7747f960fc2e201c4d47c64fee141
        b78e3ab768ce941863dc8914e8f5815f'
    ]
  },
  / redacted_claim_keys / simple(59) : [
    / redacted inspector_license_number /
    h'af375dc3fbald082448642c00be7b2f7
      bb05c9d8fb61cfc230ddfd7b4616a693'
  ]
} >>,
/ CWT signature / h'536b3797c8f396d6dc4fedfa54fa605f
  be897df02267ede5b40b257cba5eb2b3
  cbf7d4f5733e0ca2e77783d860b8d74a
  2b98878930be6c8e3d1de63df909d484
  2a800f9d377fa41e9bd5f72743c06cfb
  1e5d59892fac51a806cd4caf6cb30cce'
])

```

Figure 5: Issued SD-CWT with all disclosures

Some of the claims are `_redacted_` in the payload. The corresponding `_disclosure_` is communicated in the unprotected header in the `sd_claims` header parameter. For example, the `inspector_license_number` claim is a Salted Disclosed Claim, consisting of a per-disclosure random salt, the Claim Key, and Claim Value.

```
<<[
  /salt/    h'bae611067bb823486797dalebbb52f83',
  /value/   "ABCD-123456",
  /claim/   501   / inspector_license_number /
]>>,
```

Figure 6: CBOR extended diagnostic notation representation of inspector_license_number disclosure

This is represented in CBOR pretty-printed format as follows (with end-of-line comments and spaces inserted for clarity):

```
83                                     # array(3)
  50                                 # bytes(16)
    bae611067bb823486797dalebbb52f83 # 16-byte salt
  6b                                 # text(11)
    414243442d313233343536          # "ABCD-123456"
  19 01f5                           # unsigned(501)
```

Figure 7: CBOR encoding of inspector_license_number disclosure

The cryptographic hash, using the hash algorithm identified by the `sd_alg` header parameter in the protected headers, of that byte string is the Blinded Claim Hash (shown in hex). The digest value is included in the payload in a `redacted_claim_keys` field for a Redacted Claim Key (in this example), or in a named array for a Redacted Claim Element (for example, for the redacted claim element of `inspection_dates`).

```
d9df03da474fcb3c65771748e2e0608cf437504ecc24f450aaeacd40dd552b3f
```

Figure 8: SHA-256 hash of inspector_license_number disclosure

Finally, since this redacted claim is a map key and value, the Blinded Claim Hash is placed in a `redacted_claim_keys` array in the SD-CWT payload at the same level of hierarchy as the original claim.

```
/ redacted_claim_keys / simple(59) : [
  / redacted inspector_license_number /
  h'd9df03da474fcb3c65771748e2e0608c
    f437504ecc24f450aaeacd40dd552b3f',
  / ... next redacted claim at the same level would go here / ],
```

Figure 9: redacted inspector_license_number claim in the issued CWT payload

Redacted claims that are array elements are handled slightly differently, as described in Section 5.1.

The Issuer SHOULD confirm the Holder controls all confirmation material before issuing credentials using the cnf claim.

4. Holder prepares an SD-CWT for a Verifier

When the Holder wants to send an SD-CWT and disclose none, some, or all of the redacted values, it makes a list of the values to disclose and puts them in `sd_claims` header parameter in the unprotected header. If the Holder does not disclose any claims, it MUST omit the `sd_claims` header parameter.

For example, Alice decides to disclose to a Verifier the `inspector_license_number` claim (ABCD-123456), the `region` claim (California), and the earliest date element in the `inspection_dates` array (7-Feb-2019).

```

/ sd_claims / 17 : [ / these are the disclosures /
  <<[
    /salt/    h'bae611067bb823486797dalebbb52f83',
    /value/   "ABCD-123456",
    /claim/   501    / inspector_license_number /
  ]>>,
  <<[
    /salt/    h'8de86a012b3043ae6e4457b9e1aaab80',
    /value/   1549560720    / inspected 7-Feb-2019 /
  ]>>,
  <<[
    /salt/    h'ec615c3035d5a4ff2f5ae29ded683c8e',
    /value/   "ca",
    /claim/   "region"    / region=California /
  ]>>,
]
```

Figure 10: The Holder's choice of the disclosures to present

The Holder MAY fetch a nonce from the Verifier to prevent replay, or obtain a nonce acceptable to the Verifier through a process similar to the method described in [I-D.ietf-httpbis-unprompted-auth].

Finally, the Holder generates a Selective Disclosure Key Binding Token (SD-KBT) that ties together the SD-CWT generated by the Issuer (with the disclosures the Holder chose for the Verifier in its unprotected header), the Verifier target audience and optional nonces, and proof of possession of the Holder's private key.

The issued SD-CWT is placed in the `kcwt` (Confirmation Key CWT) protected header field (defined in [RFC9528]).

```

/ cose-sign1 / 18( / sd_kbt / [
  / KBT protected / << {
    / alg / 1: -7, / ES256 /
    / kcwt / 13: ...
    / *** SD-CWT from Issuer goes here /
    / with Holder's choice of disclosures /
    / in the SD-CWT unprotected header *** /,
  / end of issuer SD-CWT /
  / typ / 16: 294 # application/kb+cwt,
} >>, / end of KBT protected header /
/ KBT unprotected / {},
/ KBT payload / << {
  / aud / 3 : "https://verifier.example/app",
  / iat / 6 : 1725244237, / 2024-09-02T02:30:37+00:00Z /
  / cnonce / 39 : h'8c0f5f523b95bea44a9a48c649240803'
} >>, / end of KBT payload /
/ KBT signature / h'67116f888eab35fe82c171db146262c9
                      922fb3d4fd769641c80569e3c6010f90
                      251fa2b1dd335bc6bd8314603f57fd03
                      af7ddb5eb4ccele59ac07d11dfdce742'
]) / end of kbt /

```

Figure 11: Elided example of a Key Binding Token

The digests in protected parts of the issued SD-CWT and the disclosures hashed in unprotected header of the issuer_sd_cwt are used together by the Verifier to confirm the disclosed claims. Since the unprotected header of the included SD-CWT is covered by the signature in the SW-KBT, the Verifier has assurance that the Holder included the sent list of disclosures.

5. SD-CWT Definition

SD-CWT is modeled after SD-JWT, with adjustments to align with conventions in CBOR, COSE, and CWT. An SD-CWT MUST declare its content type, by including the protected header parameter typ [RFC9596] with one of the following values:

- * the string content type value application/sd-cwt,
- * the unsigned integer Constrained Application Protocol (CoAP) [RFC7252] content-format value 293,
- * or a value declaring that the object is a more specific kind of SD-CWT, such as a content type value using the +sd-cwt structured suffix.

The Issuer SHOULD use the value 293 instead of application/sd-cwt, as the CBOR representation is considerably smaller (3 bytes versus of 19).

An SD-CWT is a format based on CWT, but it allows some additional types in maps to indicate values that were or should be redacted, and includes some additional constraints to improve robustness. Unlike CWT, SD-CWT requires key binding.

An SD-CWT can contain blinded claims (each expressed as a Blinded Claim Hash), at the root level or in any arrays or maps inside that claim set. It is not required to contain any blinded claims.

Optionally the salted Claim Values (and Claim Keys) for the corresponding Blinded Claim Hash are disclosed in the sd_claims header parameter in the unprotected header of the CWT (the disclosures). If there are no disclosures (and when no Blinded Claims Hash is present in the payload) the sd_claims header parameter is not present in the unprotected header.

Any party with a Salted Disclosed Claim can generate its hash, find that hash in the CWT payload, and unblind the content. However, a Verifier with the hash cannot reconstruct the corresponding blinded claim without disclosure of the Salted Disclosed Claim.

5.1. Types of Blinded Claims

Salted Disclosed Claims for Claim Keys are structured as a 128-bit salt, the disclosed value, and the map key (the claim "name") of the redacted element. For Salted Disclosed Claims of Claim Values (items in an array), the "name" of the claim is omitted.

```

; an array of bstr-encoded Salted Disclosed Claims
salted-array = [ +bstr-encoded-salted ]

bstr-encoded-salted = bstr .cbor salted-entry
salted-entry = salted-claim / salted-element / decoy
salted-claim = [
  bstr .size 16,      ; 128-bit salt
  any,                ; Claim Value
  (int / text)        ; Claim Key
]
salted-element = [
  bstr .size 16,      ; 128-bit salt
  any                 ; Claim Value
]
decoy = [
  bstr .size 16       ; 128-bit salt
]

```

Figure 12: CDDL of Salted Disclosed Claims

When a blinded claim is a key in a map, its blinded claim hash is added to a `redacted_claim_keys` array claim in the CWT payload that is at the same level of hierarchy as the key being blinded. The `redacted_claim_keys` key is the CBOR simple value 59 registered for that purpose. CBOR "simple values" Section 3.3 of [RFC8949] are values (like false or undefined) that do need any additional content. In this specification a simple value of 59 is used as the content of a map key to indicate that one or more map key/value pairs was blinded in this CBOR map. The simple value 59 is represented in examples using the syntax `simple(59)`. The simple value 59 in CDDL is represented using the syntax `#7.59`.

When blinding an individual item in an array, the value of the item is replaced with the digested salted hash as a CBOR byte string, wrapped with the CBOR tag 60. CBOR tags Section 3.4 of [RFC8949] annotate other values. The tag 60 is represented in examples as `60(_tagged value_)`. The tag 60 is represented in CDDL as `#6.60(_tagged value_)`.

```
; redacted_claim_keys is used as a map key. The corresponding value is
; an array of Blinded Claim Hashes whose corresponding unblinded map
; keys and values are in the same map.
redacted_claim_keys = #7.59 ; CBOR simple value 59

; CBOR tag for wrapping a redacted element in an array
REDACTED_ELEMENT_TAGNUM = 60

; redacted_claim_element is used in CDDL payloads that contain
; array elements that are meant to be redacted.
redacted_claim_element = #6.<REDACTED_ELEMENT_TAGNUM>( bstr )
```

Figure 13: CDDL of Blinded Claim Keys and Blinded Claim Elements

Blinded claims can be nested. For example, both individual keys in the `inspection_location` claim, and the entire `inspection_location` element can be separately blinded. An example nested claim is shown in Section 14.2.

Finally, an Issuer MAY create decoy digests, which look like blinded claim hashes but have only a salt. Decoy digests are discussed in Section 10.

6. Differences from the CBOR Web Token Specification

The following subsections discuss differences between CWT and SD-CWT or clarify ambiguities in CWT. Some of these changes are necessary to enable the new functionality of SD-CWT, while some constraints were made in the interest of more robustness.

Variability in serialization can also be exploited to impact privacy. See Section 16 for more details on the privacy impact of serialization and profiling.

6.1. Definite Length CBOR Required

Encoders of SD-CWT and SD-KBT MUST NOT send indefinite length CBOR. Decoders of SD-CWT and SD-KBT MUST reject any SD-CWT or SD-KBT received containing indefinite length CBOR.

6.2. Finite values for standard date claims

The standard CWT claims `exp`, `nbf`, and `iat` MUST be finite numbers. For the avoidance of doubt, not a number (NaN) values and positive and negative infinity are not acceptable in those claims.

In [RFC8392], these three claims are of type `NumericDate`. Section 2 of the same spec refers to `NumericDate` as a JWT `NumericDate`, "except that it is represented as [an untagged] CBOR numeric date (from Section 2.4.1 of [RFC7049]) instead of a JSON number". In CBOR, a `NumericDate` can be represented as an unsigned integer, a negative integer, or a floating point value. CBOR (both [RFC7049] and [RFC8949]) refer to floating-point values to include NaNs, and floating-point numbers that include finite and infinite numbers. Neither JSON [RFC8259] nor JWT [RFC7519] can represent infinite values.

As IEEE double-precision floating point numbers smaller than $-(2^{53})$ and larger than 2^{53} are no longer as precise as CBOR integers, use of floating point numbers smaller than $-(2^{53})$ and larger than 2^{53} is FORBIDDEN.

6.3. Allowed types of CBOR map keys

According to Section 1.1 of the CBOR Web Token Specification [RFC8392], "CBOR uses text strings, negative integers, and unsigned integers as map keys." Section 1.5 of CBOR Object Signing and Encryption (COSE): Structures and Process [RFC9052] states: "In COSE, we use text strings, negative integers, and unsigned integers as map keys." While a CBOR map key can contain any CBOR type, this statement implies that CWT map keys only contain those types.

An SD-CWT payload is typically its COSE payload. [RFC9597] also defines the CWT Claims COSE Header Parameter (value 15) that can appear in the protected headers; if it exists, it contains a CBOR map with additional claims that are treated as if they were in the SD-CWT payload. Both of these maps are described as SD-CWT Claims Maps.

Note that CWT Claims is a separate CBOR map from the COSE payload and can contain the same Claim Keys as the COSE payload CBOR map.

The same valid CWT claim keys could be present in both SD-CWT Claims Maps, but if so, they MUST have the same unblinded value. Neither, one, or both could be redacted. If both are redacted they would have different disclosures, salts, and Blinded Claim Hashes.

In addition to map keys that are valid in CWT, SD-CWT Claims Maps MAY contain the CBOR simple value registered in this specification in Section 17.2. In SD-CWTs exchanged between the Holder and the Issuer prior to issuance, map keys MAY also consist of the To Be Redacted tag (defined in Section 11.1), containing an integer or text string; or a To Be Decoy tag (defined in Section 11.2), containing a positive integer. These two tags provide a way for the Holder to indicate specific claims to be redacted or decoys to be inserted.

The following list summarizes the map key constraints on SD-CWTs and SD-KBTs:

- * The SD-KBT protected headers kcwt Header Parameter exclusively contains:
 - a single valid SD-CWT
- * The SD-KBT protected headers MUST NOT contain a CWT Claims Header Parameter.
- * The SD-KBT payload map; unprotected headers map; and protected headers map (excluding the kcwt Header Parameter) exclusively contain map keys (at any level of depth) with the following map key types:
 - unsigned integers
 - negative integers
 - text strings with a length no greater than 255 octets
- * The SD-CWT unprotected headers map; and the protected headers map (excluding the CWT Claims Header Parameter) exclusively contain map keys (at any level of depth) with the following map key types:
 - unsigned integers
 - negative integers
 - text strings with a length no greater than 255 octets
- * The SD-CWT Claims Maps at any level of depth, exclusively contain map keys with the following map key types:
 - unsigned integers;
 - negative integers;
 - text strings with a length no greater than 255 octets;
 - the simple value 59; or
 - when disclosable claims are communicated to the Issuer, prior to issuance:
 - o the To Be Decoy tag 62 Section 11.2 containing a positive integer, or

- o the To Be Redacted tag 58 Section 11.1 containing:
 - + an unsigned integer,
 - + a negative integer, or
 - + a text strings with a length no greater than 255 octets.

In other words, there are exactly three places that can contain map keys (including values that might contain nested maps) with the SD-CWT values that are not allowed in a CWT:

- * in the payload Claims Map of an SD-CWT
- * in the CWT Claims Header Parameter Claims Map in the protected headers of an SD-CWT
- * in the kcwt Header Parameter of an SD-KBT (in one of the embedded SD-CWT Claims Maps).

All the other Header Parameters, and the KBT payload need to contain values valid in a CWT. These values are represented by the safe-value CDDL type.

```
; CWT claim legal values only
safe_map = { * label => safe_value }

safe_value =
  int / tstr / bstr /
  [ * safe_value ] /
  safe_map /
  #6.<safe_tag>(safe_value) / #7.<safe_simple> / float

; legal values in issued SD-CWT
issued_sd_cwt_map = {
  ? redacted_claim_keys ^ => [ * bstr ],
  * label => issued_sd_cwt_value
}

issued_array_element = redacted_claim_element / issued_sd_cwt_value

issued_sd_cwt_value =
  int / tstr / bstr /
  [ * issued_array_element ] /
  issued_sd_cwt_map /
  #6.<safe_tag>(issued_sd_cwt_value) / #7.<safe_simple> / float
```

```

; legal values in claim set sent to Issuer
preissuance_label = label /
    #6.<TO_BE_REDACTED_TAGNUM>(label) /
    #6.<TO_BE_DECOY_TAGNUM>(uint .gt 0)

preissuance_map = { * preissuance_label => preissuance_value }

preissuance_value =
    int / tstr / bstr /
    [ * preissuance_value ] /
    preissuance_map /
    #6.<safe_tag>(preissuance_value) / #7.<safe_simple> / float

; CBOR tag number for wrapping to-be-redacted keys or elements
TO_BE_REDACTED_TAGNUM = 58
; CBOR tag number for indicating a decoy value is to be inserted here
TO_BE_DECOY_TAGNUM = 62

label = int / tstr .size (1..255)
safe_tag = uint .ne (TO_BE_REDACTED_TAGNUM /
    TO_BE_DECOY_TAGNUM /
    REDACTED_ELEMENT_TAGNUM)

; exclude reserved simple values (24 through 31) from Table 4 of
; RFC 8949, and redacted keys array
safe_simple = 0..23 / 32..58 / 60..255
secs = int / float53
float53 = -9007199254740992.0..9007199254740992.0 ; from 2^53 to 2^53

```

Figure 14: CDDL of Safe Values in SD-CWTs

Note that Holders presenting to a Verifier that does not support this specification would need to present a CWT without tagged map keys or simple value map keys.

Tagged keys are not registered in the CBOR Web Token Claims IANA registry. Instead, the tag provides additional information about the tagged Claim Key and the corresponding (untagged) value. Multiple levels of tags in a map key are not permitted.

6.4. Duplicate map key detection

Implementations MUST NOT send multiple map keys inside the same CBOR map having the same CBOR Preferred Encoding (see Section 4.1 of [RFC8949]). This applies to any map anywhere in an SD-CWT or an SD-KBT.

Note that it is not necessary to actually encode the map keys using Preferred Encoding to satisfy this requirement.

Likewise, a single SD-CWT claim set MUST NOT contain a map (at any level of depth) with both a map key *k*, and *k* tagged with the To Be Redacted tag (see Section 11.1). Map keys and their To Be Redacted tagged version are considered duplicate map keys for the purposes of this specification.

For example, if the map below is contained inside a payload, it is invalid because the map key 500 and the map key 58(500) cannot both be present.

```
{
  500: "ABCD-123456",      # map key 500
  58(500): "DEFG-456789"  # to be redacted tag containing 500
}
```

Figure 15: EDN Example considered a duplicate map key

6.5. Level of Nesting of Claims

Selective disclosure of deeply nested structures (exceeding a depth of 16 levels), is NOT RECOMMENDED as it could lead to resource exhaustion vulnerabilities.

The individual map key / value pairs in a Claim Set are defined as the "top level", or level 1. For each value that is an array, a map, or a tagged item, each of the elements of the array, each value corresponding to each map key in the map, and the tagged item are at the next level of depth.

For example, considering the following abbreviated document, the following table shows the level of depth of the corresponding values:

Level	Value
1	https://issuer.example
2	1549560720
3	DCBA-101777
4	us
5	27315

Table 1: Levels of Depth in
Below Example

```

{
  # contents are level 1
  1: "https://issuer.example",
  ...
  502: 1(1549560720),          # tagged value is level 2
  504: [                      # contents are level 2
    {                          # contents are level 3
      ...
      501: "DCBA-101777",
      503: {                  # contents are level 4
        1: "us",
        ...
      },
      505: 4(                # decimal fraction tag
        [                    # 273.15
          -2,
          27315              # level 5
        ]
      )
    },
    ...
  ]
}

```

Figure 16: EDN Example to demonstrate levels of depth

The contents of the top-level claims map are level 1. The contents of the array for map key 504 are level 2. The contents of the map inside that array are level 3 (ex: the value of map key 505). The value of tag 4 is at level 4. The values in the array inside tag 4 are at level 5.

6.6. Use of Structured Suffixes

Any type which contains the +sd-cwt structured suffix MUST be a legal SD-CWT. A type that is a legal CWT and does not contain any blinded claims SHOULD use the +cwt structure suffix instead, unless the CBOR map being secured contains claim keys with different semantics than those registered in the CBOR Web Token Claims IANA registry.

7. SD-CWT Issuance

How the Holder communicates to the Issuer to request a CWT or an SD-CWT is out of scope for this specification. Likewise, how the Holder determines which claims to blind or to always disclose is a policy matter, which is not discussed in this specification. This specification defines the format of an SD-CWT communicated between an Issuer and a Holder in this section, and describes the format of a Key Binding Token containing that SD-CWT communicated between a Holder and a Verifier in Section 8.

The protected header MAY contain the sd_alg header parameter identifying the algorithm (from the COSE Algorithms registry) used to hash the Salted Disclosed Claims. If no sd_alg header parameter is present, the default hash function SHA-256 is used.

If any Salted Disclosed Claims or Decoys are present, the unprotected header MUST contain the sd_claims header parameter with a Salted Disclosed Claim for every blinded claim hash present anywhere in the payload, and any decoys (see Section 10). If there are no disclosures, the sd_claims header parameter value is omitted. The payload also MUST include a key confirmation element (cnf) [RFC8747] for the Holder's public key.

The following table describes the claim requirements for an SD-CWT:

Claim	Requirement	Mandatory to Disclose
sub / 2	MUST be present (disclosed or redacted)	No
iss / 1	SHOULD (see note)	Yes
aud / 3	OPTIONAL	Yes
exp / 4	OPTIONAL	Yes
nbf / 5	OPTIONAL	Yes
iat / 6	OPTIONAL	Yes
cti / 7	OPTIONAL	Yes
cnf / 8	MUST	Yes
cnonce / 39	OPTIONAL	Yes

Table 2: SD-CWT Claim Requirements

The iss claim SHOULD be present unless the protected header contains a certificate or certificate-like entity that fully identifies the Issuer.

The following table describes the claim requirements for an SD-KBT:

Claim	Requirement	Mandatory to Disclose
sub / 2	MUST NOT be present	N/A
iss / 1	MUST NOT be present	N/A
aud / 3	MUST	Yes
iat / 6	MUST (unless cti present)	Yes
cti / 7	MUST (unless iat present)	Yes
cnonce / 39	OPTIONAL	Yes

Table 3: SD-KBT Claim Requirements

Any claims not addressed in the tables above are OPTIONAL and MAY be redacted in an SD-CWT, unless specified differently by a profile or more specific media type.

To further reduce the size of the SD-CWT, a COSE Key Thumbprint (ckt) [RFC9679] MAY be used in the cnf claim.

7.1. Issuer Generation

The Issuer follows all the requirements of generating a valid SD-CWT, largely a CWT extended by Section 6. The Issuer MUST implement COSE_Sign1 using an appropriate fully-specified asymmetric signature algorithm (for example, ESP256 or Ed25519).

The Issuer MUST generate a unique cryptographically random salt with at least 128-bits of entropy for each Salted Disclosed Claim. If the client communicates a client-generated nonce (cnonce) when requesting the SD-CWT, the Issuer MUST include it in the payload.

7.2. Holder Validation

Upon receiving an SD-CWT from the Issuer with the Holder as the subject, the Holder verifies the following:

- * the issuer (iss) and subject (sub) are correct;
- * if an audience (aud) is present, it is acceptable;
- * the CWT is valid according to the nbf and exp claims, if present;

- * a public key under the control of the Holder is present in the cnf claim;
- * the hash algorithm identified by the sd_alg header parameter in the protected headers is supported by the Holder;
- * if a cnonce is present, it was provided by the Holder to this Issuer and is still fresh;
- * there are no unblinded claims about the subject that violate its privacy policies;
- * any place where the cardinality of claims needs to be protected have sufficient decoys (Section 10);
- * every blinded claim hash (some of which may be nested as in Section 14.2) has a corresponding Salted Disclosed Claim, and vice versa;
- * the values of the Salted Disclosed Claims when placed in their unblinded context in the payload are acceptable to the Holder.

A Holder MAY choose to validate the appropriateness or correctness of some or all of the information in a token, should it have the ability to do so, and it MAY choose to not present information to a Verifier that it deems to be incorrect.

The following informative CDDL is provided to describe the syntax for SD-CWT issuance. A complete CDDL schema is in Appendix A.

```

sd-cwt-issued = #6.18([
    protected: bstr .cbor sd-protected,
    sd-unprotected,
    payload: bstr .cbor sd-payload,
    signature: bstr
])

sd-protected = {
    &(typ: 16) ^ => 293 / "application/sd-cwt",
    &(alg: 1) ^ => int,
    ? &(kid: 4) ^ => bstr,
    ? &(CWT_Claims: 15) ^ => issued_sd_cwt_map,
    ? &(sd_alg: 170) ^ => int,          ; -16 for sha-256
    ? &(sd_aead: 172) ^ => uint .size 2,
    * label => safe_value
}

sd-unprotected = {
    ? &(sd_claims: 17) ^ => salted-array,
    ? &(sd_aead_encrypted_claims: 171) ^ => aead-encrypted-array,
    * label => safe_value
}

sd-payload = {
    ; standard claims
    &(iss: 1) ^ => tstr, ; "https://issuer.example"
    ? &(sub: 2) ^ => tstr, ; "https://device.example"
    ? &(aud: 3) ^ => tstr, ; "https://verifier.example/app"
    ? &(exp: 4) ^ => secs, ; 1883000000
    ? &(nbf: 5) ^ => secs, ; 1683000000
    ? &(iat: 6) ^ => secs, ; 1683000000
    ? &(cti: 7) ^ => bstr,
    &(cnf: 8) ^ => safe_map, ; key confirmation
    ? &(vct: 11) ^ => bstr,
    ? &(cnonce: 39) ^ => bstr,
    ;
    ? redacted_claim_keys ^ => [ * bstr ],
    * label => issued_sd_cwt_value
}

```

Figure 17: CDDL describing issued SD-CWT syntax

8. SD-CWT Presentation

When issuing an SD-CWT to a Holder, the Issuer includes all the Salted Disclosed Claims in the unprotected header.

By contrast, when a Holder presents an SD-CWT to a Verifier, it can disclose none, some, or all of its blinded claims. If the Holder wishes to disclose any blinded claims, it includes that subset of its Salted Disclosed Claims in the `sd_claims` header parameter of the unprotected header. If there is nothing to be disclosed, the `sd_claims` header parameter is omitted.

The Holder has flexibility in determining the order of disclosures when making presentations. The order can be sorted, randomized, or optimized for performance based on the Holder's needs.

An SD-CWT presentation to a Verifier has the same syntax as an SD-CWT issued to a Holder, except the Holder chooses the subset of disclosures included in the `sd_claims` header parameter.

Since the unprotected header is not included in the Issuer's signature, the list of disclosed claims can differ without invalidating the corresponding signature.

Finally, the SD-CWT used for presentation to a Verifier is included in a key binding token, as discussed in the next section.

8.1. Creating a Key Binding Token

Regardless if it discloses any claims, the Holder sends the Verifier a unique Holder key binding (SD-KBT) Section 8.1 for every presentation of an SD-CWT to a different Verifier.

An SD-KBT is itself a type of CWT, signed using the private key corresponding to the key in the `cnf` claim in the presented SD-CWT. The SD-KBT contains the SD-CWT, including the Holder's choice of presented disclosures, in the `kcwt` protected header field in the SD-KBT.

The Holder is conceptually both the subject and the Issuer of the Key Binding Token. Therefore, the `sub` and `iss` of an SD-KBT are implied from the `cnf` claim in the included SD-CWT, and MUST NOT be present in the SD-KBT. (Profiles of this specification MAY define additional semantics.)

The `aud` claim MUST be included and MUST correspond to the Verifier. The SD-KBT payload MUST contain either the `iat` (issued at) claim, or the `cti` (CWT ID) claim. If the Holder has access to an accurate clock, use of the `iat` is preferred.

The protected header of the SD-KBT MUST include the typ header parameter with the value application/kb+cwt or the unsigned integer value of 294. The Holder SHOULD use the value 294 instead of application/kb+cwt, as the CBOR representation is considerably smaller (3 bytes versus of 19).

The SD-KBT MUST NOT be valid for any time period when its contained SD-CWT is invalid.

The SD-KBT provides the following assurances to the Verifier:

- * the Holder of the SD-CWT controls the confirmation method chosen by the Issuer;
- * the Holder's disclosures have not been tampered with since confirmation occurred;
- * the Holder intended to address the SD-CWT to the Verifier specified in the audience (aud) claim;
- * if there are any time claims, the Holder's disclosure is linked to the creation time (iat) of the key binding; otherwise the Holder's disclosure is linked to a CWT ID (cti), which is expected to be unique per KBT.

The SD-KBT prevents an attacker from copying and pasting disclosures, or from adding or removing disclosures without detection. Confirmation is established according to [RFC8747], using the cnf claim in the payload of the SD-CWT.

The Holder signs the SD-KBT using the key specified in the cnf claim in the SD-CWT. This proves possession of the Holder's private key.


```

kbt-cwt = #6.18([
  protected: bstr .cbor kbt-protected,
  kbt-unprotected,
  payload: bstr .cbor kbt-payload,
  signature: bstr
])

kbt-protected = {
  &(typ: 16) ^ => 294 / "application/kb+cwt",
  &(alg: 1) ^ => int,
  &(kcwt: 13) ^ => sd-cwt-issued,
  * label => safe_value
}

kbt-unprotected = {
  * label => safe_value
}

kbt-payload = {
  &(aud: 3) ^ => tstr, ; "https://verifier.example/app"
  time-or-cti-claims,
  ? &(cnonce: 39) ^ => bstr,
  * label => safe_value
}

time-or-cti-claims = (
  ; Requires either time claims that include an iat OR a cti claim.
  ; Note that in CDDL, the // operator has very low precedence (see
  ; Section 2.2.2 of RFC 8610)
  ;
  ; It is possible to have both a cti and time claims, but
  ; time-or-cti-claims insures at least cti or iat is present
  ? &(exp: 4) ^ => secs,
  ? &(nbf: 5) ^ => secs,
  &(iat: 6) ^ => secs //
  &(cti: 7) ^ => bstr
)

```

Figure 18: CDDL describing KBT syntax

The SD-KBT payload MAY include a cnonce claim. If included, the cnonce is a bstr and MUST be treated as opaque to the Holder. All other claims are OPTIONAL in an SD-KBT.

9. SD-KBT and SD-CWT Verifier Validation

To protect against replay attacks, the Verifier SHOULD provide a nonce, and reject requests that do not include an acceptable nonce (cnonce). This guidance can be ignored in cases where replay attacks are mitigated at another layer.

The exact order of the following steps MAY be changed, as long as all checks are performed before deciding if an SD-CWT is valid.

1. First the Verifier must open the protected headers of the SD-KBT and find the Issuer SD-CWT present in the kcwt field.
2. Next, the Verifier must validate the SD-CWT as described in Section 7.2 of [RFC8392]. Validators MUST treat an `sd_claims` or `sd_aead_encrypted_claims` unprotected Header Parameter with an empty array as invalid.
3. The Verifier checks the time claims in the SD-CWT as follows:
 - * `nbf` \leq `iat` (if both exist)
 - * `nbf` $<$ `exp` (if both exist)
 - * `iat` $<$ `exp` (if both exist)
4. The Verifier extracts the confirmation key from the `cnf` claim in the SD-CWT payload.
5. Using the confirmation key, the Verifier validates the SD-KBT as described in Section 7.2 of [RFC8392].
6. The Verifier checks the time claims in the SD-KBT to enforce the following logical constraints:
 - * if no `cti` claim is present in the SD-KBT, there MUST be an `iat` in the SD-KBT;
 - * if there is no `iat` claim in the SD-KBT, there MUST NOT be an `exp` claim or an `nbf` claim in the SD-KBT;
 - * `nbf` in the SD-KBT \leq `iat` in the SD-KBT (if both exist)
 - * `iat` in the SD-KBT $<$ `exp` in the SD-KBT (if both exist)
 - * `nbf` in the SD-CWT \leq `iat` in the SD-CWT (if both exist)
 - * `iat` in the SD-CWT $<$ `exp` in the SD-CWT (if both exist)

- * nbf in the SD-CWT < exp in the SD-CWT (if both exist)
 - * exp in the SD-KBT <= exp in the SD-CWT (if both exist)
 - * nbf in the SD-KBT >= nbf in the SD-CWT (if both exist)
 - * iat in the SD-KBT >= iat in the SD-CWT (if both exist)
 - * nbf in the SD-KBT < exp in the SD-CWT (if both exist)
 - * iat in the SD-KBT < exp in the SD-CWT (if both exist)
 - * iat in the SD-KBT >= nbf in the SD-CWT (if both exist)
7. The Verifier MUST extract and decode the disclosed claims from the sd_claims header parameter in the unprotected header of the SD-CWT. Each decoded disclosure is treated as if it is a claim key or claim element at the location corresponding to its Blinded Claim Hash in the payload. If there are any disclosures that do not have a corresponding Blinded Claim Hash, the entire SD-CWT is invalid. If any decoded Redacted Claim Key duplicates another claim key in the same position, the entire SD-CWT is invalid.
- Note: A Verifier MUST be prepared to process disclosures in any order. When disclosures are nested, a disclosed value could appear before the disclosure of its parent.
8. A Verifier MUST reject the SD-CWT if the audience claim in either the SD-CWT or the SD-KBT contains a value that does not correspond to the intended recipient.
9. Otherwise, the SD-CWT is considered valid. Once any remaining redacted elements (either redacted claims or decoys) are deleted, the Validated Disclosed Claims Set is now a CWT Claims Set with no claims marked for redaction.
- Note: Undisclosed Redacted Claim Elements will be removed from the Validated Disclosed Claims Set, changing the length of the array. If the semantics of the position of items in the array is important, the issuer should instead disclose or redact the entire array.
10. Further validation logic can be applied to the Validated Disclosed Claims Set, just as it might be applied to a validated CWT Claims Set.

By performing these steps, the recipient can cryptographically verify the integrity of the protected claims and verify they have not been tampered with.

10. Decoy Digests

An Issuer MAY add additional digests to the SD-CWT payload (including nested maps and arrays within the payload) that are not associated with any unblinded claim. The purpose of such "decoy" digests is to make it more difficult for an adversarial Verifier to infer private information based on the number of Redacted Claim Keys or Redacted Claim Elements.

The list of disclosures sent by the Issuer to the Holder contains one disclosure for each decoy digest. Each disclosure contains a single element array with a per-decoy salt. Each salt MUST be unique.

```
<<[ h'C1069BC056E234D64F58BAFF8A7B776B' ]>>
```

Figure 19: EDN showing a sample decoy salt

The Blinded Claim Hash for each disclosure is calculated using the same algorithm for decoys as for Redacted Claim Keys and Redacted Claim Elements. An example issued SD-CWT containing decoy digests is shown below.

```
/ cose-sign1 / 18([ / issuer SD-CWT /
/ CWT protected / << {
/ alg / 1 : -35, / ES384 /
/ kid / 4 : 'https://issuer.example/cose-key3',
/ typ / 16 : 293, # application/sd-cwt
/ sd_alg / 170 : -16 / SHA256 /
} >>,
/ CWT unprotected / {
/ sd_claims / 17 : [ / these are all the disclosures /
<<[
/salt/ h'2bfeab9315829fe0c82a15b2c57a47b2',
/value/ "fr" / France /
]>>,
<<[
/salt/ h'c1069bc056e234d64f58baff8a7b776b',
]>>,
<<[
/salt/ h'b0392772caefd08178218f86f3e2b3a9',
/value/ true,
/claim/ 500 / inspection result /
]>>,
<<[
```

```

    /salt/    h'89c41c270dd06cd3517d371c9ddf2bab',
  ]>>,
]
},
/ CWT payload / << {
  / iss / 1   : "https://issuer.example",
  / sub / 2   : "https://device.example",
  / exp / 4   : 1725330600, /2024-09-03T02:30:00+00:00Z/,
  / nbf / 5   : 1725243900, /2024-09-02T02:25:00+00:00Z/,
  / iat / 6   : 1725244200, /2024-09-02T02:30:00+00:00Z/,
  / cnf / 8   : {
    / cose key / 1 : {
      / kty / 1: 2, / EC2 /
      / crv / -1: 1, / P-256 /
      / x /   -2: h'8554eb275dcd6fbd1c7ac641aa2c90d9
                    2022fd0d3024b5af18c7cc61ad527a2d',
      / y /   -3: h'4dc7ae2c677e96d0cc82597655ce92d5
                    503f54293d87875d1e79ce4770194343'
    }
  },
/countries array/ 98: [
  /redacted country == "fr" /
  60(h'dc5f753b66acd89d78481039934a86cc
    14f9959c64c4037dea3f872b9a8453f1')
  /decoy country #1 /
  60(h'3f80963a1246b412d6567f2a5ca446fd
    19a01dd8cfc291bed69e8c575c5abfb8')
],
/ redacted_claim_keys / simple(59) : [
  / redacted claim 500 (== true) /
  h'bd0fd88127b3071ff5433eef59a5e3c5
    f18341f25c5bd119c41fd34802a9797b'
  / decoy claim #2 /
  h'eeec970897a5b9108f24f44751baedab
    b53alf3d241ab6b60c9f309f114ecf88'
]
} >>,
/ CWT signature / h'fd0b7edae0fa5bd429f07809762a32c5
                  c160c089091a5c3abbee91710fa388d8
                  3c1ad6a5a3d219c4265b9f89aef1f511
                  ff362cbbea6f46bae08978822b672aa5
                  75cd6fc519394af2a883c394faf16ac1
                  781afb01f626bdd6fee1205a29dcc203'
])

```

Figure 20: EDN showing a SD-CWT containing decoys

As with regular Blinded Claims, the Holder needs to receive the disclosure for every decoy so it can be sure the Issuer is not communicating unwanted information to Verifiers (see Section 16.3).

11. Tags Used Before SD-CWT Issuance

This section describes the semantics of two CBOR tags that are (optionally) only used to convey information to the Issuer about disclosures to create.

11.1. To Be Redacted Tag Definition

In order to indicate specific claims that the Holder would like to be redacted in a Claim Set, this specification defines a new CBOR tag "To Be Redacted". The tag can be used by a library to automatically convert a Claim Set with "To Be Redacted" tags into a) a new Claim Set containing Redacted Claim Keys and Redacted Claim Elements replacing the tagged claim keys or claim elements, and b) a set of corresponding Salted Disclosed Claims.

When used on an element in an array, the value to be redacted is present inside the tag. When used on a map key and value, the tag is placed around the map key, while the map value remains.

Examples in this draft use the To Be Redacted tag in order to distinguish their pre-issued, post-issued, and post-presented representations in EDN and CDDL. The snippet of EDN shown below shows one mechanism to communicate to the Issuer to redact the inspector license number claim, and two of the inspection dates in our primary example.

```
{
  ...
  58(501): "ABCD-123456",    # redact inspector license number claim
  /inspection dates/ 502: [
    58(1549560720),        # redact 07-Feb-2019
    58(1612560720),        # redact 04-Feb-2021
    1674004740             # don't redact 17-Jan-2023
  ]
  ...
}
```

Figure 21: EDN example requesting blinding of license number and two inspection dates

11.2. To Be Decoy

In order to indicate a location that should contain a decoy digest Section 10 in the issued SD-CWT, this specification defines a new CBOR tag "To Be Decoy". This tag can be used by a library to automatically a) add a decoy digest at a particular location in an array, or at a particular level in a map; and b) create the corresponding Salted Disclosed Claims. The value inside is a positive integer that MUST be unique for each decoy location within the SD-CWT. The integer could be used to look up the salt for the decoy deterministically, but does not impose any ordering. When a decoy digest is requested in a map, the map `_value_` is always null.

Note: Requiring an integer that is unique per decoy within the entire CWT simplifies the implementation of SD-CWT libraries. Issuers that receive To Be Decoy tags with duplicate values inside the same CWT MUST NOT issue an SD-CWT based on the preissued Claims Set.

In the example fragment below, the transit countries claim contains an array of countries. The Claim Elements array contains Germany (de) and the Philippines (ph). The Holder wants to redact each country, but add decoys to obfuscate the number of component origin countries. The example fragment also shows two decoy digests in the same map.

```
{
  ...
  /component origin countries/ 607: [
    58("de"),
    58("ph"),
    62(1),      # add two decoys in this array
    62(2)
  ],
  62(3): null,  # add a decoy in this map
  62(4): null,  # add a second decoy in the same map
  ...
}
```

Figure 22: EDN example requesting two decoys

12. Encrypted Disclosures

The RATS architecture [RFC9334] defines a model where the Verifier is split into separate entities, with an initial verifier called an Attester, and a target entity called a Relying Party. Other protocols have a similar type of internal structure for the Verifier.

In some of these use cases, there is existing usage of AES-128 GCM and other Authenticated Encryption with Additional Data (AEAD) [RFC5116] algorithms.

This section describes how to use AEADs to encrypt disclosures to a target entity, while enabling a initial verifier to confirm the authenticity of the presentation from the Holder.

In these systems, an appropriate symmetric key and its context are provided completely out-of-band.

The entire SD-CWT is included in the protected header of the SD-KBT, which secures the entire Issuer-signed SD-CWT including its unprotected headers that include its disclosures.

When encrypted disclosures are present, they MUST be in the unprotected headers of the Issuer-signed SD-CWT, before the SD-KBT can be generated by the Holder.

The initial Verifier of the key binding token might not be able to decrypt encrypted disclosures and MAY decide to forward them to an inner Verifier that can decrypt them.

12.1. AEAD Encrypted Disclosures Mechanism

This section defines two new COSE Header Parameters. If present in the protected headers, the first header parameter (`sd_aead`) specifies an Authenticated Encryption with Additional Data (AEAD) algorithm [RFC5116] registered in the IANA AEAD Algorithms registry (<https://www.iana.org/assignments/aead-parameters/aead-parameters.xhtml>). (Guidance on specific algorithms is discussed in Section 16.8.) The second header parameter (`sd_aead_encrypted_claims`), if present, contains a non-empty list of AEAD encrypted disclosures. Taking the first example disclosure from above:

```
<<[
  /salt/    h'bae611067bb823486797dalebbb52f83',
  /value/    "ABCD-123456",
  /claim/    501    / inspector_license_number /
]>>,
```

Figure 23: EDN example of an encrypted disclosure

The corresponding bstr is encrypted with an AEAD algorithm [RFC5116]. If present, the algorithm of the `sd_aead` protected header field is used, or `AEAD_AES_128_GCM` if no algorithm was specified. The bstr is encrypted with a unique, random 16-octet nonce. The AEAD ciphertext

consists of its encryption algorithm's ciphertext and its authentication tag. (For example, in AEAD_AES_128_GCM the authentication tag is 16 octets.) The nonce (nonce), the encryption algorithm's ciphertext (ciphertext) and authentication tag (tag) are put in an array. The resulting array is placed in the `sd_aead_encrypted_claims` header parameter in the unprotected headers of the SD-CWT.

Given the AEAD encryption key (in hexadecimal):

```
a061c27a3273721e210d031863ad81b6
```

A valid AEAD encrypted disclosure for that first disclosure is:

```
/ sd_aead_encrypted_claims / 171 : [ / AEAD encrypted disclosures /
[
  / nonce /      h'95d0040fe650e5baf51c907c31be15dc',
  / ciphertext / h'2e9a9570254916fd81db30a4126381c4
                  dc8221a2d95e85a7191747e12f68b2fd
                  9cfae5',
  / tag /        h'799147ab37efe19d2b14ff330bfff08b'
],
  ...
]
```

Figure 24: EDN Example of an AEAD Encrypted Claim

The blinded claim hash is still over the unencrypted disclosure. The receiver of an AEAD encrypted disclosure locates the appropriate key by looking up the authentication tag. If the Verifier is able to decrypt and verify an encrypted disclosure, the decrypted disclosure is then processed as if it were in the `sd_claims` header parameter in the unprotected headers of the SD-CWT.

Details of key management are left to profiles of the specific protocols that make use of AEAD encrypted disclosures.

The CDDL for AEAD encrypted disclosures is below.

```
aead-encrypted-array = [ +aead-encrypted ]
aead-encrypted = [
  bstr .size 16,      ; 128-bit nonce
  bstr,               ; the encryption ciphertext output of a
                      ; bstr-encoded-salted
  bstr                ; the corresponding authentication tag
]
```

Figure 25: CDDL describing syntax of AEAD Encrypted disclosures

Note: Because the encryption algorithm is in a registry that contains only AEAD algorithms, an attacker cannot replace the algorithm or the message, without a decryption verification failure.

13. Credential Types

This specification defines the CWT claim `vct` (for Verifiable Credential Type). The `vct` value is an identifier for the type of the SD-CWT Claims Set. Like the `typ` header parameter [RFC9596], its value can be either a string or an integer. For size reasons, it is RECOMMENDED that the numeric representation be used.

If its value is a string, it is a case-sensitive `StringOrURI`, as defined in [RFC7519]. In this case, the `vct` string MUST either be registered in the IANA "Verifiable Credential Type Identifiers" registry established in Section 17.8, or be a Collision-Resistant Name, as defined in Section 2 of [RFC7515].

If its value is an integer, it is either a value in the range 0-64999 registered in the IANA "Verifiable Credential Type Identifiers" registry established in Section 17.8 or an Experimental Use value in the range 65000-65535, which is not to be used in operational deployments.

This claim is defined for COSE-based verifiable credentials, similar to the JOSE-based verifiable credentials claim (`vct`) described in Section 3.2.2.1.1 of [I-D.draft-ietf-oauth-sd-jwt-vc].

14. Examples

14.1. Minimal Spanning Example

The following example contains claims needed to demonstrate redaction of key-value pairs and array elements.

```
/ cose-sign1 / 18( / sd_kbt / [  
  / KBT protected / << {  
    / alg /      1:  -7, / ES256 /  
    / kcwt /    13:  18([ / issuer SD-CWT /  
      / CWT protected / << {  
        / alg /      1  : -35, / ES384 /  
        / kid /      4  : 'https://issuer.example/cose-key3',  
        / typ /     16 : 293, # application/sd-cwt  
        / sd_alg / 170 : -16 / SHA256 /  
      } >>,  
    / CWT unprotected / {  
      / sd_claims / 17 : [ / these are the disclosures /
```

```

    <<[
      /salt/    h'bae611067bb823486797dalebbb52f83',
      /value/   "ABCD-123456",
      /claim/   501    / inspector_license_number /
    ]>>,
    <<[
      /salt/    h'8de86a012b3043ae6e4457b9e1aaab80',
      /value/   1549560720    / inspected 7-Feb-2019 /
    ]>>,
    <<[
      /salt/    h'ec615c3035d5a4ff2f5ae29ded683c8e',
      /value/   "ca",
      /claim/   "region"    / region=California /
    ]>>,
  ]
},
/ CWT payload / << {
  / iss / 1    : "https://issuer.example",
  / sub / 2    : "https://device.example",
  / exp / 4    : 1725330600,    /2024-09-03T02:30:00+00:00Z/
  / nbf / 5    : 1725243900,    /2024-09-02T02:25:00+00:00Z/
  / iat / 6    : 1725244200,    /2024-09-02T02:30:00+00:00Z/
  / cnf / 8    : {
    / cose key / 1 : {
      / kty / 1: 2,    / EC2    /
      / crv / -1: 1,    / P-256 /
      / x /    -2: h'8554eb275dcd6fbd1c7ac641aa2c90d9
                    2022fd0d3024b5af18c7cc61ad527a2d',
      / y /    -3: h'4dc7ae2c677e96d0cc82597655ce92d5
                    503f54293d87875d1e79ce4770194343'
    }
  },
  /most_recent_inspection_passed/ 500: true,
  /inspection_dates/ 502 : [
    / redacted inspection date 7-Feb-2019 /
    60(h'1b7fc8ecf4b1290712497d226c04b503
      b4aa126c603c83b75d2679c3c613f3fd'),
    / redacted inspection date 4-Feb-2021 /
    60(h'64afccd3ad52da405329ad935de1fb36
      814ec48fdfd79e3a108ef858e291e146'),
    1674004740,    / 2023-01-17T17:19:00 /
  ],
  / inspection_location / 503 : {
    "country" : "us",    / United States /
    / redacted_claim_keys / simple(59) : [
      / redacted region /
      h'0d4b8c6123f287a1698ff2db15764564
      a976fb742606e8fd00e2140656ba0df3'
    ]
  }
}

```

```

        / redacted postal_code /
        h'c0b7747f960fc2e201c4d47c64fee141
          b78e3ab768ce941863dc8914e8f5815f'
    ]
  },
  / redacted_claim_keys / simple(59) : [
    / redacted inspector_license_number /
    h'af375dc3fbald082448642c00be7b2f7
      bb05c9d8fb61cfc230dddfdb4616a693'
  ]
} >>,
/ CWT signature / h'536b3797c8f396d6dc4fedfa54fa605f
                  be897df02267ede5b40b257cba5eb2b3
                  cbf7d4f5733e0ca2e77783d860b8d74a
                  2b98878930be6c8e3d1de63df909d484
                  2a800f9d377fa41e9bd5f72743c06cfb
                  1e5d59892fac51a806cd4caf6cb30cce'
]),
/ end of issuer SD-CWT /
/ typ / 16: 294 # application/kb+cwt,
} >>, / end of KBT protected header /
/ KBT unprotected / {},
/ KBT payload / << {
  / aud / 3 : "https://verifier.example/app",
  / iat / 6 : 1725244237, / 2024-09-02T02:30:37+00:00Z /
  / cnonce / 39 : h'8c0f5f523b95bea44a9a48c649240803'
} >>, / end of KBT payload /
/ KBT signature / h'67116f888eab35fe82c171db146262c9
                  922fb3d4fd769641c80569e3c6010f90
                  251fa2b1dd335bc6bd8314603f57fd03
                  af7ddb5eb4ccele59ac07d11dfdce742'
]) / end of kbt /

```

Figure 26: A minimal, full EDN Example

14.2. Nested Example

Instead of the structure from the previous example, imagine that the payload contains an inspection history log with the following structure. It could be blinded at multiple levels of the claims set hierarchy.

```

{
  / iss / 1 : "https://issuer.example",
  / sub / 2 : "https://device.example",
  / exp / 4 : 1725330600, /2024-09-02T19:30:00Z/
  / nbf / 5 : 1725243840, /2024-09-01T19:25:00Z/
  / iat / 6 : 1725244200, /2024-09-01T19:30:00Z/
  / cnf / 8 : { ... },
  504: [
    {
      500: True,           / inspection passed /
      502: 1549560720,     / 2019-02-07T17:32:00 /
      501: "DCBA-101777", / inspector license /
      503: {
        1: "us",          / United States /
        2: "co",          / region=Colorado /
        3: "80302"        / postcode /
      }
    },
    {
      500: True,           / inspection passed /
      502: 1612560720,     / 2021-02-04T20:14:00 /
      501: "EFGH-789012", / inspector license /
      503: {
        1: "us",          / United States /
        2: "nv",          / region=Nevada /
        3: "89155"        / postcode /
      }
    },
    {
      500: True,           / inspection passed /
      502: 17183928,       / 2023-01-17T17:19:00 /
      501: "ABCD-123456", / inspector license /
      503: {
        1: "us",          / United States /
        2: "ca",          / region=California /
        3: "94188"        / postcode /
      }
    }
  ],
}

```

Figure 27: EDN example of Nested Claims Set before blinding

For example, looking at the nested disclosures below, the first disclosure unblinds the entire January 2023 inspection record. However, when the record is disclosed, the inspector license number and inspection location are redacted inside the record. The next disclosure unblinds the inspector_license_number, and the next

disclosure unblinds the inspection location record, but the region and postcode claims inside the location record are also individually blinded. The fourth disclosure unblinds the inspection region.

The fifth disclosure unblinds the earliest inspection record, and the last disclosure unblinds the inspector_license_number for that record.

Verifiers start unblinding claims for which they have blinded claim hashes. They continue descending until there are no blinded claim hashes at any level of the hierarchy for which they have a corresponding disclosure.

```
/ sd_claims / 17 : [ / these are the disclosures /
  <<[
    /salt/    h'cd99b3858f1d659f9d16039abf8c5fba',
    /value/   {
      500: true,
      502: 1674004740,
      simple(59): [
        h'af375dc3fbald082448642c00be7b2f7
          bb05c9d8fb61cfc230ddfdfb4616a693',
        h'9d151abeb800adcc11ff10ff61fbd3d7
          5944c134b40a24abef1787d3ae6583aa'
      ]
    } / inspection 17-Jan-2023 /
  ]>>,
  <<[
    /salt/    h'bae611067bb823486797dalebbb52f83',
    /value/   "ABCD-123456",
    /claim/   501 / inspector_license_number /
  ]>>,
  <<[
    /salt/    h'483e4b3c194df6073a9c41ca9f274067',
    /value/   {
      1: "us",
      simple(59): [
        h'2470fb9175b062c347ab3c3a19776d02
          476112a17cd7cfc9416664bc058c220b',
        h'cf397a08917528624ca3b332c9edcc54
          a72c9411dd5983f68017ce160f709f52'
      ]
    },
    /claim/   503 / San Francisco location /
  ]>>,
  <<[
    /salt/    h'52da9de5dc61b33775f9348b991d3d78',
    /value/   "ca",
```

```

    /claim/ 2 / region=California /
  ]>>,
  <<[
    /salt/ h'2df7d2c105b5bf3acf9c698f3658552f',
    /value/ {
      500: true,
      502: 1549560720,
      simple(59): [
        h'7257a8697dfa40221079b00fb65fe587
          c310e6ca3da1aa33b090335de66ec810',
        h'c24c646b52fec773c6ea01c6caa5a73
          422b85d3afa5900fa998336d83a88025'
      ]
    } / inspection 7-Feb-2019 /
  ]>>,
  <<[
    /salt/ h'591eb2081b05be2dcbb6f8459cc0fe51',
    /value/ "DCBA-101777",
    /claim/ 501 / inspector_license_number /
  ]>>,
  <<[
    /salt/ h'c23a4d192be75dbd583be570482de8dd',
    /value/ {
      1: "us",
      simple(59): [
        h'1b89717167f39d51eec08b13baeda570
          eff5d0aedaald7d0821185c33634a5a0',
        h'49412884fa1e3787c17d1320bdd48f6e
          0e5365da010cde0571d4a7effd13cc2a'
      ]
    },
    /claim/ 503 / Denver location /
  ]>>,
]
```

Figure 28: EDN Example of SD-CWT with Nested Blinded Claims

After applying the disclosures of the nested structure above, the disclosed Claims Set visible to the Verifier would look like the following:

```

{
  / iss / 1 : "https://issuer.example",
  / sub / 2 : "https://device.example",
  / exp / 4 : 1725330600, /2024-09-02T19:30:00Z/
  / nbf / 5 : 1725243840, /2024-09-01T19:25:00Z/
  / iat / 6 : 1725244200, /2024-09-01T19:30:00Z/
  / cnf / 8 : { ... },
  504: [
    {
      500: True,           / inspection passed /
      501: "DCBA-101777", / inspector license /
      502: 1549560720,    / 2019-02-07T17:32:00 /
      503: {
        1: "us"           / United States /
      }
    },
    {
      500: True,           / inspection passed /
      501: "ABCD-123456", / inspector license /
      502: 17183928,      / 2023-01-17T17:19:00 /
      503: {
        1: "us",          / United States /
        2: "ca"           / region=California /
      }
    }
  ]
}

```

Figure 29: EDN of validated nested claims

15. Privacy Considerations

This section describes the privacy considerations in accordance with the recommendations from [RFC6973]. Many of the topics discussed in [RFC6973] apply to SD-CWT, but are not repeated here.

15.1. Correlation

Presentations of the same SD-CWT to multiple Verifiers can be correlated by matching on the signature component of the COSE_Sign1. That is, SD-CWT does not naturally provide Verifier-Verifier unlinkability. Signature based linkability can be mitigated by issuing multiple single-use tokens, at a credential management complexity cost.

Any presentation of an SD-CWT can be correlated with its issuance if an Issuer and Verifier cooperate. That is, SD-CWT cannot provide Verifier-Issuer unlinkability.

Any Claim Value that pertains to a sufficiently small set of subjects can be used to facilitate tracking the subject. For example, a high precision issuance time might match the issuance of only a few credentials for a given Issuer, and as such, any presentation of a credential issued at that time can be determined to be associated with the set of credentials issued at that time, for those subjects.

15.2. Determinism

It is possible to encode additional information through the choices made during the serialization stage of producing an SD-CWT, for example, by adjusting the order of CBOR map keys, or by choosing different numeric encodings for certain data elements. [I-D.draft-ietf-cbor-cde] provides guidance for constructing application profiles that constrain serialization optionality beyond CBOR Common Deterministic Encoding rulesets (CDE). The construction of such profiles has a significant impact on the privacy properties of a credential type.

15.3. Audience

If the audience claim is present in both the SD-CWT and the SD-KBT, they are not required to be the same. SD-CWTs with audience claims that do not correspond to the intended recipients **MUST** be rejected, to protect against accidental disclosure of sensitive data.

15.4. Credential Types

The privacy implications of selective disclosure vary significantly across different credential types due to their inherent characteristics and intended use cases. The mandatory and optional-to-disclose data elements in an SD-CWT must be carefully chosen based on the specific privacy risks associated with each credential type.

For example, a passport credential contains highly sensitive personal information where even partial disclosure can have significant privacy implications:

- * Revealing citizenship status may expose an individual to discrimination
- * Date of birth combined with any other attribute enables age-based profiling
- * Biometric data, even if selectively disclosed, presents irreversible privacy risks

- * The mere possession of a passport from certain countries can be sensitive information

In contrast, a legal entity certificate has fundamentally different privacy considerations:

- * The entity's legal name and registration number are often public information
- * Business addresses and contact details may already be in public registries
- * Authorized signatories' names might be required for legal validity
- * The primary concern is often business confidentiality rather than personal privacy

These differences mean that:

- * Passport credentials should minimize mandatory disclosures and maximize holder control over optional elements
- * Legal entity certificates might reasonably require disclosure of more fields to establish business legitimacy
- * The granularity of selective disclosure should match the credential type's privacy sensitivity
- * Default disclosure sets must be carefully calibrated to each credential's risk profile

Several distinct credential types might be applicable to a given use case, each with unique privacy trade-offs. Issuers MUST perform a comprehensive privacy and confidentiality assessment for each credential type they intend to issue, considering:

- * The sensitivity spectrum of contained attributes
- * Likely disclosure scenarios and their privacy impacts
- * Correlation risks when attributes are combined
- * Long-term privacy implications of disclosed information
- * Cultural and jurisdictional privacy expectations

16. Security Considerations

Security considerations from COSE [RFC9052] and CWT [RFC8392] apply to this specification.

16.1. Issuer Key Compromise

Verification of an SD-CWT requires that the Verifier have access to a verification key (public key) associated with the Issuer. Compromise of the Issuer's signing key would enable an attacker to forge credentials for any subject, potentially undermining the entire trust model of the credential system. Beyond key compromise, attacks targeting the provisioning and binding between issuer names and their cryptographic key material pose significant risks. An attacker who can manipulate these bindings could substitute their own keys for legitimate issuer keys, enabling credential forgery while appearing to be a trusted issuer.

16.2. Disclosure Coercion and Over-identification

The Security Considerations from Section 10.2 of [I-D.draft-ietf-oauth-selective-disclosure-jwt] apply, with additional attention to disclosure coercion risks. Holders face risks of being coerced into disclosing more claims than necessary. This threat warrants special attention because:

1. Verifier Trust: Holders MUST be able to verify that a Verifier will handle disclosed claims appropriately and only for stated purposes.
2. Elevated Risk: Claims from trusted authorities (e.g., government-issued credentials) carry higher misuse potential due to their inherent legitimacy.
3. Irreversibility: Disclosed claims cannot be withdrawn. This permanent exposure risk MUST be considered in any disclosure decision.

Mitigation Measures:

1. Verifiers SHOULD demonstrate eligibility to receive claims
2. Holders MUST conduct risk assessments when Verifier eligibility cannot be established
3. Trust lists maintained by trusted parties can help identify authorized Verifiers

Without proper safeguards (such as Verifier trust lists), Holders remain vulnerable to over-identification and long-term misuse of their disclosed information.

Implementers deploying SD-CWT in credential systems should review [W3CTAG-Credential-Abuse], which discusses broader risks of credential abuse including surveillance, exclusion, and overuse.

16.3. Disclosure of Decoys

Issuers control the claimset and may include decoy digests to obscure the number of claims in a credential. Unless all decoy disclosures are made available to Holders, an Issuer could use this mechanism to hide claims from Holders without their knowledge. A Holder receiving a credential with undisclosed decoys cannot distinguish between a digest that conceals a real claim and one that is genuinely a decoy. This creates a severe privacy risk: an Issuer could embed hidden claims — such as behavioral flags, risk scores, or sensitive attributes — that Verifiers can later be given selective access to, without the Holder's awareness or consent.

16.4. Random Numbers

Each salt used to protect disclosed claims MUST be generated independently. Identical salt values, or values that can be correlated, might allow a Verifier to recover blinded values that were not disclosed.

The salts MUST be generated using fresh randomness or from a source that has outputs cannot be predicted by any potential Verifier. Poor choice of salts can lead to brute force attacks that can reveal redacted claims.

16.5. Binding the KBT and the CWT

The "iss" claim in the SD-CWT is self-asserted by the Issuer.

Because confirmation is mandatory, the subject claim of an SD-CWT, when present, is always related directly to the confirmation claim. There might be many subject claims and many confirmation keys that identify the same entity or that are controlled by the same entity, while the identifiers and keys are distinct values. Reusing an identifier or key enables correlation, but MUST be evaluated in terms of the confidential and privacy constraints associated with the credential type. Conceptually, the Holder is both the Issuer and the subject of the SD-KBT, even if the "iss" or "sub" claims are not present. If they are present, they are self-asserted by the Holder. All three are represented by the confirmation (public) key in the SD-CWT.

As with any self-assigned identifiers, Verifiers need to take care to verify that the SD-KBT "iss" and "sub" claims match the subject in the SD-KBT, and are a valid representation of the Holder and correspond to the Holder's confirmation key. Extra care should be taken in case the SD-CWT subject claim is redacted. Likewise, Holders and Verifiers MUST verify that the "iss" claim of the SD-CWT corresponds to the Issuer and the key described in the protected header of the SD-CWT.

Finally, the Verifier MUST verify that the time claims in both the SD-CWT and SD-KBT are self-consistent and that the SD-KBT is not valid for any period of time when the SD-CWT is not. Specific tests for time claims are described in steps 3 and 6 of Section 9. Likewise, if there is a notion of SD-CWT revocation, an SD-KBT containing a revoked SD-CWT is not valid.

16.6. Covert Channels

Any data element that is supplied by the Issuer, and that appears random to the Holder might be used to produce a covert channel between the Issuer and the Verifier. The ordering of claims, and precision of timestamps can also be used to produce a covert channel. This is more of a concern for SD-CWT than typical CWTs, because the Holder is usually considered to be aware of the Issuer claims they are disclosing to a Verifier.

16.7. Nested Disclosure Ordering

The Holder has flexibility in determining the order of nested disclosures when making presentations. The order can be sorted, randomized, or optimized for performance based on the Holder's needs. This ordering choice has no security impact on encrypted disclosures. However, the order can affect the runtime of the verification process.

16.8. Choice of AEAD algorithms

The AEAD encrypted disclosures mechanism discussed in Section 12.1 can refer to any AEAD algorithm in the IANA AEAD Algorithms registry (<https://www.iana.org/assignments/aead-parameters/aead-parameters.xhtml>) .

When choosing an AEAD algorithm, the tag length is critical for the integrity of encrypted disclosures in SD-CWT. As such, implementations MUST NOT use any AEAD algorithm with a tag length less than 16 octets.

Algorithms using AES-CCM are NOT RECOMMENDED.

As of this writing, implementations MUST NOT use algorithms 3 through 14, 18, 19, 21, 22, 24, 25, 27, or 28. Implementations using the AEGIS algorithms containing an X MUST only use the 256-bit tag variant.

17. IANA Considerations

17.1. COSE Header Parameters

IANA is requested to add the following entries to the IANA "COSE Header Parameters" registry (<https://www.iana.org/assignments/cose/cose.xhtml#header-parameters>):

17.1.1. sd_claims

The following completed registration template per RFC8152 is provided:

- * Name: sd_claims
- * Label: 17
- * Value Type: [+bstr]
- * Value Registry: (empty)
- * Description: A list of selectively disclosed claims, which were originally redacted, then later disclosed at the discretion of the sender.
- * Reference: Section 4 of this specification

17.1.2. sd_alg

The following completed registration template per RFC8152 is provided:

- * Name: sd_alg
- * Label: 170
- * Value Type: int
- * Value Registry: IANA COSE Algorithms
- * Description: The hash algorithm used for redacting disclosures.
- * Reference: Section 7 of this specification

17.1.3. sd_aead_encrypted_claims

The following completed registration template per RFC8152 is provided:

- * Name: sd_aead_encrypted_claims
- * Label: 171
- * Value Type: [+[bstr,bstr,bstr]]
- * Value Registry: (empty)
- * Description: A list of AEAD encrypted selectively disclosed claims, which were originally redacted, then later disclosed at the discretion of the sender.
- * Reference: Section 12.1 of this specification

17.1.4. sd_aead

The following completed registration template per RFC8152 is provided:

- * Name: sd_aead
- * Label: 172
- * Value Type: uint .size 2
- * Value Registry: IANA AEAD Algorithm number

- * Description: The AEAD algorithm used for encrypting disclosures.
- * Reference: Section 12.1 of this specification

17.2. CBOR Simple Values

IANA is requested to add the following entry to the IANA "CBOR Simple Values" registry (<https://www.iana.org/assignments/cbor-simple-values#simple>):

- * Value: 59
- * Semantics: This value as a map key indicates that the Claim Value is an array of redacted Claim Keys at the same level as the map key.
- * Specification Document(s): Section 5.1 of this specification

17.3. CBOR Tags

IANA is requested to add the following entries to the IANA "CBOR Tags" registry (<https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml#tags>):

17.3.1. To Be Redacted Tag

The array claim element, or map key and value inside the "To be redacted" tag is intended to be redacted using selective disclosure.

- * Tag: 58
- * Data Item: (any)
- * Semantics: An array claim element intended to be redacted, or a map key whose key and value are intended to be redacted.
- * Specification Document(s): Section 11.1 of this specification

17.3.2. Redacted Claim Element Tag

The byte string inside the tag is a selective disclosure redacted claim element of an array.

- * Tag: 60
- * Data Item: byte string
- * Semantics: A selective disclosure redacted (array) claim element.

- * Specification Document(s): Section 5.1 of this specification

17.3.3. To Be Decoy Tag

The positive integer inside the tag is a unique number to indicate a specific decoy instance among all the instances in the document.

- * Tag: 62 (requested)
- * Data Item: uint .gt 0
- * Semantics: A marker of a location in a map or an array where a decoy is intended to be inserted.
- * Specification Document(s): Section 11.2 of this specification

17.4. CBOR Web Token (CWT) Claims

IANA is requested to add the following entry to the IANA "CWT Claims" registry (<https://www.iana.org/assignments/cwt/cwt.xhtml#claims-registry>):

17.4.1. vct

The following completed registration template per RFC8392 is provided:

- * Claim Name: vct
- * Claim Description: Verifiable credential type
- * JWT Claim Name: vct
- * Claim Key: 11
- * Claim Value Type(s): bstr
- * Change Controller: IETF
- * Specification Document(s): Section 13 of this specification

17.5. Media Types

IANA is requested to add the following entries to the IANA "Media Types" registry (<https://www.iana.org/assignments/media-types/media-types.xhtml#application>):

17.5.1. application/sd-cwt

The following completed registration template is provided:

- * Type name: application
- * Subtype name: sd-cwt
- * Required parameters: n/a
- * Optional parameters: n/a
- * Encoding considerations: binary
- * Security considerations: Section 16 of this specification and [RFC8392]
- * Interoperability considerations: n/a
- * Published specification: Section 5 of this specification
- * Applications that use this media type: TBD
- * Fragment identifier considerations: n/a
- * Additional information:
 - 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: SPICE WG mailing list (spice@ietf.org) or IETF Security Area (saag@ietf.org)
- * Intended usage: COMMON
- * Restrictions on usage: none
- * Author: See Author's Addresses section
- * Change controller: IETF
- * Provisional registration? No

17.5.2. application/kb+cwt

The following completed registration template is provided:

- * Type name: application
- * Subtype name: kb+cwt
- * Required parameters: n/a
- * Optional parameters: n/a
- * Encoding considerations: binary
- * Security considerations: Section 16 of this specification and [RFC8392]
- * Interoperability considerations: n/a
- * Published specification: Section 8.1 of this specification
- * Applications that use this media type: TBD
- * Fragment identifier considerations: n/a
- * Additional information:
 - 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: SPICE WG mailing list (spice@ietf.org) or IETF Security Area (saag@ietf.org)
- * Intended usage: COMMON
- * Restrictions on usage: none
- * Author: See Author's Addresses section
- * Change controller: IETF
- * Provisional registration? No

17.6. Structured Syntax Suffix

IANA is requested to add the following entry to the IANA "Structured Syntax Suffix" registry (<https://www.iana.org/assignments/media-type-structured-suffix/media-type-structured-suffix.xhtml#structured-syntax-suffix>):

- * Name: SD-CWT
- * +suffix: +sd-cwt
- * References: Section 5 of this specification
- * Encoding considerations: binary
- * Interoperability considerations: n/a
- * Fragment identifier considerations: n/a
- * Security considerations: Section 16 of this specification
- * Contact: See Author's Addresses section
- * Author/Change controller: IETF

17.7. Content-Formats

IANA is requested to register the following entries in the IANA "CoAP Content-Formats" registry (<https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats>):

Content-Type	Content Coding	ID	Reference
application/sd-cwt	-	293	Section 5 of this specification
application/kb+cwt	-	294	Section 8.1 of this specification

Table 4: New CoAP Content Formats

17.8. Verifiable Credential Type Identifiers

This specification establishes the Verifiable Credential Type Identifiers registry, under the IANA "CBOR Web Token (CWT) Claims" group registry heading (<https://www.iana.org/assignments/cwt/cwt.xhtml>). It registers identifiers for the type of the SD-CWT Claims Set.

It enables short integers in the range 0-65535 to be used as vct Claim Values, similarly to how CoAP Content-Formats (Section 12.3 of [RFC7252]) enable short integers to be used as typ header parameter [RFC9596] values.

The registration procedures for numbers in specific ranges are as described below:

Range	Registration Procedure from RFC8126
0-9999	Specification Required
10000-64999	First Come First Served
65000-65535	Experimental Use (no operational use)

Table 5

Values in the Specification Required [RFC8126] range are registered after a two-week review period on the spice-ext-review@ietf.org mailing list, on the advice of one or more Designated Experts. To allow for the allocation of values prior to publication of the final version of a specification, the Designated Experts may approve registration once they are satisfied that the specification will be completed and published. However, if the specification is not completed and published in a timely manner, as determined by the Designated Experts, the Designated Experts may request that IANA withdraw the registration.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register VCT value").

Within the review period, the Designated Experts will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful. The IANA escalation process can be initiated by the party requesting registration when the Designated Experts are not responsive within 14 days.

Criteria that should be applied by the Designated Experts includes determining whether the proposed registration duplicates existing functionality, determining whether it is likely to be of general applicability or whether it is useful only for a single application, and whether the registration makes sense.

IANA must only accept registry updates from the Designated Experts and should direct all requests for registration in the Specification Required range to the review mailing list.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly-informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

17.8.1. Registration Template

Verifiable Credential Type Identifier String: String identifier for use as a JWT vct or CWT vct Claim Value. It is a StringOrURI value.

Verifiable Credential Type Identifier Number: Integer in the range 0-64999 for use as a CWT vct Claim Value. (Integers in the range 65000-65535 are not to be registered.)

Description: Brief description of the verifiable credential type

Change Controller: For IETF stream RFCs, use "IETF". For others, give the name of the responsible party. Other details (e.g., postal address, e-mail address, home page URI) may also be included.

Specification Document(s): Reference to the document or documents that specify the values to be registered, preferably including URLs that can be used to retrieve the documents. An indication of the relevant sections may also be included, but is not required.

17.8.2. Initial Registry Contents

No initial values are provided for the registry.

18. References

18.1. Normative 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>>.
- [I-D.ietf-cbor-edn-literals]
Bormann, C., "CBOR Extended Diagnostic Notation (EDN)",
Work in Progress, Internet-Draft, draft-ietf-cbor-edn-
literals-19, 16 October 2025,
<[https://datatracker.ietf.org/doc/html/draft-ietf-cbor-
edn-literals-19](https://datatracker.ietf.org/doc/html/draft-ietf-cbor-edn-literals-19)>.
- [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>>.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated
Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008,
<<https://www.rfc-editor.org/rfc/rfc5116>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web
Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May
2015, <<https://www.rfc-editor.org/rfc/rfc7515>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token
(JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015,
<<https://www.rfc-editor.org/rfc/rfc7519>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <<https://www.rfc-editor.org/rfc/rfc9053>>.
- [RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <<https://www.rfc-editor.org/rfc/rfc9528>>.
- [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/rfc/rfc9596>>.
- [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>>.

18.2. Informative References

- [I-D.draft-ietf-cbor-cde] Bormann, C., "CBOR Common Deterministic Encoding (CDE)", Work in Progress, Internet-Draft, draft-ietf-cbor-cde-13, 13 October 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-cbor-cde-13>>.

- [I-D.draft-ietf-keytrans-protocol]
McMillion, B. and F. Linker, "Key Transparency Protocol",
Work in Progress, Internet-Draft, draft-ietf-keytrans-
protocol-03, 19 October 2025,
<[https://datatracker.ietf.org/doc/html/draft-ietf-
keytrans-protocol-03](https://datatracker.ietf.org/doc/html/draft-ietf-keytrans-protocol-03)>.
- [I-D.draft-ietf-oauth-sd-jwt-vc]
Terbu, O., Fett, D., and B. Campbell, "SD-JWT-based
Verifiable Digital Credentials (SD-JWT VC)", Work in
Progress, Internet-Draft, draft-ietf-oauth-sd-jwt-vc-15,
26 February 2026, <[https://datatracker.ietf.org/doc/html/
draft-ietf-oauth-sd-jwt-vc-15](https://datatracker.ietf.org/doc/html/draft-ietf-oauth-sd-jwt-vc-15)>.
- [I-D.draft-ietf-oauth-selective-disclosure-jwt]
Fett, D., Yasuda, K., and B. Campbell, "Selective
Disclosure for JWTs (SD-JWT)", Work in Progress, Internet-
Draft, draft-ietf-oauth-selective-disclosure-jwt-22, 29
May 2025, <[https://datatracker.ietf.org/doc/html/draft-
ietf-oauth-selective-disclosure-jwt-22](https://datatracker.ietf.org/doc/html/draft-ietf-oauth-selective-disclosure-jwt-22)>.
- [I-D.ietf-httpbis-unprompted-auth]
Schinazi, D., Oliver, D., and J. Hoyland, "The Concealed
HTTP Authentication Scheme", Work in Progress, Internet-
Draft, draft-ietf-httpbis-unprompted-auth-12, 19 September
2024, <[https://datatracker.ietf.org/doc/html/draft-ietf-
httpbis-unprompted-auth-12](https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-unprompted-auth-12)>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J.,
Morris, J., Hansen, M., and R. Smith, "Privacy
Considerations for Internet Protocols", RFC 6973,
DOI 10.17487/RFC6973, July 2013,
<<https://www.rfc-editor.org/rfc/rfc6973>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object
Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,
October 2013, <<https://www.rfc-editor.org/rfc/rfc7049>>.
- [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>>.
- [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/rfc/rfc8126>>.

- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/rfc/rfc8259>>.
- [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/rfc/rfc9162>>.
- [RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <<https://www.rfc-editor.org/rfc/rfc9334>>.
- [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/rfc/rfc9597>>.
- [t-Closeness]
"t-Closeness: Privacy Beyond k-Anonymity and l-Diversity",
4 June 2007,
<<https://ieeexplore.ieee.org/document/4221659>>.
- [W3CTAG-Credential-Abuse]
W3C Technical Architecture Group, "Preventing the Abuse of Digital Credentials", n.d.,
<<https://w3ctag.github.io/prevent-credential-abuse/>>.

Appendix A. Complete CDDL Schema

```
sd-cwt-types = sd-cwt-issued / kbt-cwt / sd-cwt-preissued

sd-cwt-issued = #6.18([
  protected: bstr .cbor sd-protected,
  sd-unprotected,
  payload: bstr .cbor sd-payload,
  signature: bstr
])

sd-protected = {
  &(typ: 16) ^ => 293 / "application/sd-cwt",
  &(alg: 1) ^ => int,
  ? &(kid: 4) ^ => bstr,
  ? &(CWT_Claims: 15) ^ => issued_sd_cwt_map,
  ? &(sd_alg: 170) ^ => int,           ; -16 for sha-256
  ? &(sd_aead: 172) ^ => uint .size 2,
  * label => safe_value
}
```

```

sd-unprotected = {
  ? &(sd_claims: 17) ^ => salted-array,
  ? &(sd_aead_encrypted_claims: 171) ^ => aead-encrypted-array,
  * label => safe_value
}

sd-payload = {
  ; standard claims
  &(iss: 1) ^ => tstr, ; "https://issuer.example"
  ? &(sub: 2) ^ => tstr, ; "https://device.example"
  ? &(aud: 3) ^ => tstr, ; "https://verifier.example/app"
  ? &(exp: 4) ^ => secs, ; 1883000000
  ? &(nbf: 5) ^ => secs, ; 1683000000
  ? &(iat: 6) ^ => secs, ; 1683000000
  ? &(cti: 7) ^ => bstr,
  &(cnf: 8) ^ => safe_map, ; key confirmation
  ? &(vct: 11) ^ => bstr,
  ? &(cnonce: 39) ^ => bstr,
  ;
  ? redacted_claim_keys ^ => [ * bstr ],
  * label => issued_sd_cwt_value
}

kbt-cwt = #6.18([
  protected: bstr .cbor kbt-protected,
  kbt-unprotected,
  payload: bstr .cbor kbt-payload,
  signature: bstr
])

kbt-protected = {
  &(typ: 16) ^ => 294 / "application/kb+cwt",
  &(alg: 1) ^ => int,
  &(kcwt: 13) ^ => sd-cwt-issued,
  * label => safe_value
}

kbt-unprotected = {
  * label => safe_value
}

kbt-payload = {
  &(aud: 3) ^ => tstr, ; "https://verifier.example/app"
  time-or-cti-claims,
  ? &(cnonce: 39) ^ => bstr,
  * label => safe_value
}

```

```

time-or-cti-claims = (
  ; Requires either time claims that include an iat OR a cti claim.
  ; Note that in CDDL, the // operator has very low precedence (see
  ; Section 2.2.2 of RFC 8610)
  ;
  ; It is possible to have both a cti and time claims, but
  ; time-or-cti-claims insures at least cti or iat is present
  ? &(exp: 4) ^ => secs,
  ? &(nbf: 5) ^ => secs,
    &(iat: 6) ^ => secs //
    &(cti: 7) ^ => bstr
)
sd-cwt-preissued = #6.18([
  protected: bstr .cbor sd-protected,
  sd-unprotected,
  payload: bstr .cbor preissuance_map,
  signature: bstr
])

; CWT claim legal values only
safe_map = { * label => safe_value }

safe_value =
  int / tstr / bstr /
  [ * safe_value ] /
  safe_map /
  #6.<safe_tag>(safe_value) / #7.<safe_simple> / float

; legal values in issued SD-CWT
issued_sd_cwt_map = {
  ? redacted_claim_keys ^ => [ * bstr ],
  * label => issued_sd_cwt_value
}

issued_array_element = redacted_claim_element / issued_sd_cwt_value

issued_sd_cwt_value =
  int / tstr / bstr /
  [ * issued_array_element ] /
  issued_sd_cwt_map /
  #6.<safe_tag>(issued_sd_cwt_value) / #7.<safe_simple> / float

; legal values in claim set sent to Issuer
preissuance_label = label /
  #6.<TO_BE_REDACTED_TAGNUM>(label) /
  #6.<TO_BE_DECOY_TAGNUM>(uint .gt 0)

```

```

preissuance_map = { * preissuance_label => preissuance_value }

preissuance_value =
  int / tstr / bstr /
  [ * preissuance_value ] /
  preissuance_map /
  #6.<safe_tag>(preissuance_value) / #7.<safe_simple> / float

; CBOR tag number for wrapping to-be-redacted keys or elements
TO_BE_REDACTED_TAGNUM = 58
; CBOR tag number for indicating a decoy value is to be inserted here
TO_BE_DECOY_TAGNUM = 62

label = int / tstr .size (1..255)
safe_tag = uint .ne (TO_BE_REDACTED_TAGNUM /
                    TO_BE_DECOY_TAGNUM /
                    REDACTED_ELEMENT_TAGNUM)

; exclude reserved simple values (24 through 31) from Table 4 of
; RFC 8949, and redacted keys array
safe_simple = 0..23 / 32..58 / 60..255
secs = int / float53
float53 = -9007199254740992.0..9007199254740992.0 ; from 2^53 to 2^53

salted-array = [ +bstr-encoded-salted ]

bstr-encoded-salted = bstr .cbor salted-entry
salted-entry = salted-claim / salted-element / decoy
salted-claim = [
  bstr .size 16,      ; 128-bit salt
  any,                ; Claim Value
  (int / text)        ; Claim Key
]
salted-element = [
  bstr .size 16,      ; 128-bit salt
  any                 ; Claim Value
]
decoy = [
  bstr .size 16       ; 128-bit salt
]

aead-encrypted-array = [ +aead-encrypted ]
aead-encrypted = [
  bstr .size 16,      ; 128-bit nonce
  bstr,               ; the encryption ciphertext output of a
                      ;   bstr-encoded-salted
  bstr                ; the corresponding authentication tag
]

```

```
; redacted_claim_keys is used as a map key. The corresponding value is
; an array of Blinded Claim Hashes whose corresponding unblinded map
; keys and values are in the same map.
redacted_claim_keys = #7.59 ; CBOR simple value 59

; CBOR tag for wrapping a redacted element in an array
REDACTED_ELEMENT_TAGNUM = 60

; redacted_claim_element is used in CDDL payloads that contain
; array elements that are meant to be redacted.
redacted_claim_element = #6.<REDACTED_ELEMENT_TAGNUM>( bstr )
```

Figure 30: A complete CDDL description of SD-CWT

Appendix B. Comparison to SD-JWT

SD-CWT is modeled after SD-JWT, with adjustments to align with conventions in CBOR, COSE, and CWT.

B.1. Media Types

The COSE equivalent of application/sd-jwt is application/sd-cwt.

The COSE equivalent of application/kb+jwt is application/kb+cwt.

The COSE equivalent of the +sd-jwt structured suffix is +sd-cwt.

B.2. Redaction Claims

The COSE equivalent of `_sd` is a CBOR Simple Value (requested assignment 59). The following value is an array of the redacted Claim Keys.

The COSE equivalent of `...` is a CBOR tag (requested assignment 60) of the digested salted claim.

In SD-CWT, the order of the fields in a disclosure is salt, value, key. In SD-JWT the order of fields in a disclosure is salt, key, value. This choice ensures that the second element in the CBOR array is always the value, which makes parsing faster and more efficient in strongly-typed programming languages.

In SD-CWT, all decoy digests are disclosed between the Issuer and the Holder. In SD-JWT, no disclosure is sent for a decoy digest.

B.3. Issuance

The issuance process for SD-CWT is similar to SD-JWT, with the exception that a confirmation claim is REQUIRED.

B.4. Presentation

The presentation process for SD-CWT is similar to SD-JWT, except that a Key Binding Token is REQUIRED. The Key Binding Token then includes the issued SD-CWT, including the Holder-selected disclosures. Because the entire SD-CWT is included as a claim in the SD-KBT, the disclosures are covered by the Holder's signature in the SD-KBT, but not by the Issuer's signature in the SD-CWT.

B.5. Validation

The validation process for SD-CWT is similar to SD-JWT, however, JSON Objects are replaced with CBOR Maps, which can contain integer keys and CBOR Tags.

Appendix C. Keys Used in the Examples

C.1. Subject / Holder

Holder COSE key pair in EDN format

```
{
  /kty/ 1 : 2, /EC/
  /alg/ 3 : -7, /ES256/
  /crv/ -1 : 1, /P-256/
  /x/ -2 : h'8554eb275dcd6fbd1c7ac641aa2c90d9
          2022fd0d3024b5af18c7cc61ad527a2d',
  /y/ -3 : h'4dc7ae2c677e96d0cc82597655ce92d5
          503f54293d87875d1e79ce4770194343',
  /d/ -4 : h'5759a86e59bb3b002dde467da4b52f3d
          06e6c2cd439456cf0485b9b864294ce5'
}
```

The fields necessary for the COSE Key Thumbprint [RFC9679] in EDN format:

```
{
  /kty/ 1 : 2, /EC/
  /crv/ -1 : 1, /P-256/
  /x/    -2 : h'8554eb275dcd6fbd1c7ac641aa2c90d9
            2022fd0d3024b5af18c7cc61ad527a2d',
  /y/    -3 : h'4dc7ae2c677e96d0cc82597655ce92d5
            503f54293d87875d1e79ce4770194343'
}
```

The same map in CBOR pretty printing

```
A4                                # map(4)
  01                                # unsigned(1)
  02                                # unsigned(2)
  20                                # negative(0)
  01                                # unsigned(1)
  21                                # negative(1)
  58 20                            # bytes(32)
  8554EB275DCD6FBD1C7AC641AA2C90D92022FD0D3024B5AF18C7CC61AD527A2D
  22                                # negative(2)
  58 20                            # bytes(32)
  4DC7AE2C677E96D0CC82597655CE92D5503F54293D87875D1E79CE4770194343
```

The COSE thumbprint (in hexadecimal)--SHA256 hash of the thumbprint fields:

```
8343d73cdfcb81f2c7cd11a5f317be8eb34e4807ec8c9ceb282495cffdf037e0
```

Holder key pair in JWK format

```
{
  "kty": "EC",
  "alg": "ES256",
  "kid": "WRQ2RbY5RYJCIXfDQL9agl9fFSCYVu4Xocqb6zerc1M",
  "crv": "P-256",
  "x": "hVTrJl3Nb70cesZBqiyQ2SAi_Q0wJLWvGMfMYa1Sei0",
  "y": "TceuLGd-ltDMgll2Vc6S1VA_VCk9h4ddHnnOR3AZQ0M",
  "d": "Vlmoblm7OwAt3kZ9pLUvPQbmws1DlFbPBIW5uGQpTOU"
}
```

Input to Holder public JWK thumbprint (ignore line breaks)

```
{"crv": "P-256", "kty": "EC", "x": "hVTrJl3Nb70cesZBqiyQ2SAi_Q0wJLWvGMfMYa1Sei0", "y": "TceuLGd-ltDMgll2Vc6S1VA_VCk9h4ddHnnOR3AZQ0M"}
```

SHA-256 of the Holder public JWK input string (in hex)

```
59143645b6394582422317c340bf5a825f5f15209856ee17a1ca9beb37ab7353
```


Holder public JWK thumbprint

```
WRQ2RbY5RYJCIXfDQL9ag19fFSCYVu4Xocqb6zerc1M
```

Holder public key in PEM format

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEhVTrJ13Nb70cesZBqiyQ2SAi/Q0w
JLWvGMfMYa1SeilNx64sZ36W0MyCWxZVzpLVUD9UKT2Hh10eec5HcB1DQw==
-----END PUBLIC KEY-----
```

Holder private key in PEM format

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGBYqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgV1moblm7OwAt3kZ9
pLUvPQbmws1DlFbPBIW5uGQpTOWhRANCAASFVOsnXc1vvRx6xkGqLJDZICL9DTAk
ta8Yx8xhRVJ6LU3HrixnfpbQzIJZdlXOktVQPlQpPYeHXR55zkdwGUND
-----END PRIVATE KEY-----
```

C.2. Issuer

Issuer COSE key pair in Extended Diagnostic Notation (EDN)

```
{
  /kty/ 1 : 2, /EC/
  /kid/ 2 : "https://issuer.example/cwk3.cbor",
  /alg/ 3 : -51, /ESP384/
  /crv/ -1 : 2, /P-384/
  /x/ -2 : h'c31798b0c7885fa3528fbf877e5b4c3a6dc67a5a5dc6b307
          b728c3725926f2abe5fb4964cd91e3948a5493f6ebb6cbbf',
  /y/ -3 : h'8f6c7ec761691cad374c4daa9387453f18058ece58eb0a8e
          84a055a31fb7f9214b27509522c159e764f8711e11609554',
  /d/ -4 : h'71c54d2221937ea612db1221f0d3ddf771c9381c4e3be41d
          5aa0a89d685f09cfef74c4bbf104783fd57e87ab227d074c'
}
```

The fields necessary for the COSE Key Thumbprint [RFC9679] in EDN format:

```
{
  /kty/ 1 : 2, /EC/
  /crv/ -1 : 2, /P-384/
  /x/ -2 : h'c31798b0c7885fa3528fbf877e5b4c3a6dc67a5a5dc6b307
          b728c3725926f2abe5fb4964cd91e3948a5493f6ebb6cbbf',
  /y/ -3 : h'8f6c7ec761691cad374c4daa9387453f18058ece58eb0a8e
          84a055a31fb7f9214b27509522c159e764f8711e11609554'
}
```

The same map in CBOR pretty printing

```
A4                                # map(5)
  01                              # unsigned(1)
  02                              # unsigned(2)
  20                              # negative(0)
  02                              # unsigned(2)
  21                              # negative(1)
  58 30                          # bytes(48)
    C31798B0C7885FA3528FBF877E5B4C3A6DC67A5A5DC6B307
    B728C3725926F2ABE5FB4964CD91E3948A5493F6EBB6CBBF
  22                              # negative(2)
  58 30                          # bytes(48)
    8F6C7EC761691CAD374C4DAA9387453F18058ECE58EB0A8E
    84A055A31FB7F9214B27509522C159E764F8711E11609554
```

The COSE thumbprint (in hexadecimal)--SHA256 hash of the thumbprint fields:

```
554550a611c9807b3462cfec4a690a1119bc43b571da1219782133f5fd6dbcb0
```

Issuer key pair in JWK format

```
{
  "kty": "EC",
  "alg": "ES384",
  "kid": "https://issuer.example/cwk3.cbor",
  "crv": "P-384",
  "x": "wxeYsMeIX6NSj7-HfltMOm3GelpdxrMHtyjDclkm8qvl-0lkzZHjlIpUk_brtsu_",
  "y": "j2x-x2FpHK03TE2qk4dFPxgFjs5Y6wqOhKBVox-3-SFLJ1CVIsFZ52T4cR4RYJVU",
  "d": "ccVNIiGTfqYS2xIh8NPd93HJOBxOO-QdWqConWhfCc_vdMS78QR4P9V-h6sifQdM"
}
```

Input to Issuer JWK thumbprint (ignore line breaks)

```
{"crv": "P-384", "kty": "EC", "x": "wxeYsMeIX6NSj7-HfltMOm3GelpdxrMHtyjDclkm8qvl-0lkzZHjlIpUk_brtsu_", "y": "j2x-x2FpHK03TE2qk4dFPxgFjs5Y6wqOhKBVox-3-SFLJ1CVIsFZ52T4cR4RYJVU"}
```

SHA-256 of the Issuer JWK input string (in hex)

```
18d4ddb7065d945357e3972dee76af4eddc7c285fb42efcfa900c6a4f8437850
```

Issuer JWK thumbprint

```
GNTdtwZdlFNX45ct7navTt3HwoX7Qu_PqQDGpPhDeFA
```

Issuer public key in PEM format

```
-----BEGIN PUBLIC KEY-----
MHYwEAYHKOZIZj0CAQYFK4EEACIDYgAEwxeYsMeIX6NSj7+HflTMom3GelpdxrMH
tyjDclkm8qvl+0lkzZHjlIpUk/brtsu/j2x+x2FpHK03TE2qk4dFPxgFjs5Y6wqO
hKBVox+3+SFLJlCVIsFZ52T4cR4RYJVU
-----END PUBLIC KEY-----
```

Issuer private key in PEM format

```
-----BEGIN PRIVATE KEY-----
MIG2AgEAMBAGByqGSM49AgEGBSuBBAAiBIGeMIGbAgEBDBxxU0iIZN+phLbEiHw
0933cck4HE475BlaoKidaF8Jz+90xLvxBHg/lX6HqyJ9B0yhZANiAATDF5iwx4hf
o1KPv4d+W0w6bcZ6Wl3Gswe3KMNyWSbyq+X7SWTNkeOUilST9uu2y7+PbH7HYWkc
rTdMTaqTh0U/GAWOzljrCo6EoFWjH7f5IUsnUJUiwVnnZPhxHhFglVQ=
-----END PRIVATE KEY-----
```

Appendix D. Implementation Status

Note to the 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 made 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".

D.1. Transmute Prototype

Organization: Transmute Industries Inc

Name: github.com/transmute-industries/sd-cwt (<https://github.com/transmute-industries/sd-cwt>)

Description: An open-source implementation of this specification.

Maturity: Prototype

Coverage: The current version ('main') implements functionality similar to that described in this specification, and will be revised, with breaking changes to support the generation of example data to support this specification.

License: Apache-2.0

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

Contact: Orie Steele (orie.steele@tradeverifyd.com)

D.2. Rust Prototype

Organization: SimpleLogin

Name: github.com/beltram/esdicawt (<https://github.com/beltram/esdicawt>)

Description: An open-source Rust implementation of this specification in Rust.

Maturity: Prototype

Coverage: The current version is close to the spec with the exception of `redacted_claim_keys` using a CBOR SimpleValue as label instead of a tagged key. Not all of the verifications have been implemented yet.

License: Apache-2.0

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

Contact: Beltram Maldant (beltram.ietf.spice@pm.me)

D.3. Python Prototype

Organization: Tradeverifyd

Name: github.com/tradeverifyd/sd-cwt (<https://github.com/tradeverifyd/sd-cwt>)

Description: An open-source Python implementation of this specification.

Maturity: Prototype

Coverage: The current version does not implement decoys, but does verify the test vectors in draft-ietf-spice-sd-cwt-05.

License: Apache-2.0

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

Contact: Orie Steele (orie.steele@tradeverifyd.com)

Appendix E. Relationship between RATS Architecture and Verifiable Credentials

This appendix describes the relationship between the Remote ATtestation procedureS (RATS) architecture defined in [RFC9334] and the three-party model used in verifiable credentials.

E.1. Three-Party Verifiable Credentials Model

The verifiable credentials model involves three distinct parties:

- * ***Issuer***: Creates and signs credentials containing claims about a subject
- * ***Holder***: Controls the credential and presents it to verifiers (the holder is typically the subject of the credential)
- * ***Verifier***: Receives and validates presented credentials to make authorization or access decisions. In this appendix we refer to this role as a ***Credential Verifier***

In SD-CWT, these roles are explicitly represented: the Issuer signs claims using an Assertion Key (Section 2), the Holder controls the credential and creates presentations using a Confirmation Key, and the Verifier validates both the Issuer's signature over the credential and the Holder's signature over the presentation (key binding token).

E.2. RATS Architecture Roles

The RATS architecture defines the following key roles:

- * ***Attester***: Produces Evidence about its own trustworthiness and operational state
- * ***Endorser***: Provides Endorsements about an Attester (typically a manufacturer)

- * ***Reference Value Provider***: Supplies Reference Values used by Verifiers to evaluate Evidence
- * ***Verifier***: Appraises Evidence and produces Attestation Results. In this appendix we refer to this role as a ***RATS Verifier***
- * ***Relying Party***: Consumes Attestation Results to make authorization decisions

E.3. Role Mappings in the Three-Party Model

The mapping between RATS roles and verifiable credential roles can be understood as follows:

E.3.1. Verifiable Credential Issuer as RATS Endorser

A verifiable credential Issuer functions as a RATS Endorser. The Endorser role in RATS produces Endorsements - secure statements about an Attester's capabilities, identity, or trustworthiness. Similarly, a credential Issuer produces signed credentials containing claims about a subject (the Holder). Both roles:

- * Make authoritative statements about another party's attributes or capabilities
- * Use cryptographic signatures to ensure integrity and authenticity
- * Are typically trusted third parties in their respective ecosystems
- * Provide information that enables downstream authorization decisions

The credential issued by the Issuer serves the same function as an Endorsement in RATS: it is a signed assertion about the Holder's attributes that can be used by Credential Verifiers to make trust decisions.

E.3.2. Verifiable Credential Holder as RATS Verifier

A verifiable credential Holder functions as a RATS Verifier. The RATS Verifier appraises Evidence and Endorsements and produces Attestation Results. In the credentials model, the Holder:

- * Receives credentials (analogous to Endorsements) from Issuers
- * Evaluates which credentials to present and which claims to disclose

- * Produces presentations (analogous to Attestation Results) that are sent to Credential Verifiers
- * Uses their Confirmation Key to create key binding tokens that prove control

The Holder's presentation, which includes the Issuer's credential plus the Holder's signature over selected disclosures, functions as an Attestation Result - a processed, signed assertion derived from the original credential (Endorsement).

E.3.3. Verifiable Credential Verifier as RATS Relying Party

A verifiable credential Credential Verifier functions as a RATS Relying Party. The Relying Party:

- * Consumes Attestation Results (credential presentations) to make authorization decisions
- * Validates the cryptographic integrity of received assertions
- * Makes access control or authorization decisions based on the claims received
- * Does not directly interact with the original Endorsement source (the Issuer)

The Credential Verifier appraises the Holder's presentation in the same way a Relying Party appraises Attestation Results from a RATS Verifier.

E.3.4. All Parties Can Be Attesters

Importantly, any of these parties - Issuer, Holder, or Credential Verifier - can simultaneously function as a RATS Attester. The Attester role in RATS is about producing Evidence about one's own trustworthiness:

- * An *Issuer* may be an Attester when it needs to prove its own integrity, platform state, or authorization to issue certain credential types. For example, an Issuer might provide Evidence about its secure enclave or certified infrastructure when establishing trust with Holders or during credential issuance.

- * A **Holder** may be an Attester when presenting credentials, particularly when the presentation itself requires proof of the Holder's platform integrity. For example, a Holder might provide Evidence about their device's secure boot state, firmware version, or trusted execution environment alongside their credential presentation.
- * A **Credential Verifier** may be an Attester when it needs to prove its own trustworthiness to Holders or to upstream systems. For example, a Credential Verifier might provide Evidence about its data protection capabilities, compliance certifications, or secure processing environment before Holders agree to disclose sensitive claims.

The Attester role is orthogonal to the three primary roles - it represents the ability to produce evidence about one's own state, while the Issuer/Holder/Credential Verifier roles represent the flow of credentials and claims about subjects.

E.4. Comparison with RATS Interaction Models

RATS defines two interaction models:

Passport Model: The Attester sends Evidence to a RATS Verifier, receives Attestation Results, and presents these results to Relying Parties. This maps to the three-party credentials model where the Holder obtains credentials from Issuers and presents them to Credential Verifiers.

Background-Check Model: The Attester sends Evidence to a Relying Party, which forwards it to a RATS Verifier. The RATS Verifier returns results directly to the Relying Party. This is a two-party model from the Attester's perspective and does not map well to the three-party credentials model, as it lacks Holder mediation and control over presentations.

E.5. Roles That Don't Map to the Three-Party Model

The **Reference Value Provider** role from RATS does not have a direct equivalent in the three-party verifiable credentials model. This role supplies reference values (known-good measurements or configurations) that RATS Verifiers use to appraise Endorsements and Evidence. In credentials systems, equivalent functionality might be provided through:

- * Trust registries that list authorized Issuers

- * Schema registries, or lists of valid claims that define credential formats
- * Governance frameworks that specify validation rules
- * Revocation registries

However, these are typically considered part of the trust infrastructure rather than a distinct party in the presentation protocol. The Reference Value Provider role is primarily relevant in scenarios where raw Evidence must be evaluated against known-good values - a pattern more common in the two-party background-check model than in the three-party credentials model where Issuers have already performed evaluation and produced credentials.

E.6. Application to SD-CWT

When applying RATS concepts to SD-CWT:

- * SD-CWT credentials function as Endorsements about the Holder (subject)
- * The Holder's key binding token and selective disclosure act as the RATS Verifier's appraisal and production of Attestation Results
- * The Credential Verifier consumes these presentations as a Relying Party consumes Attestation Results
- * Any party can additionally provide Evidence about their own platform or operational state (act as an Attester)
- * The three-party model with selective disclosure maps naturally to the RATS passport model
- * Reference Value Provider functionality is addressed through trust infrastructure and out-of-band mechanisms rather than protocol-level roles

Appendix F. Sample Disclosure Matching Algorithm for Verifier

The Verifier of an SD-CWT needs to decode disclosed claims match them with their redacted versions. The following example algorithm describes a way to accomplish this.

1. The decoded `sd_claims` are converted to an intermediate data structure called a Digest To Disclosed Claim Map that is used to transform the Presented Disclosed Claims Set into a Validated Disclosed Claims Set.

2. The Verifier MUST compute the hash of each Salted Disclosed Claim (salted), in order to match each disclosed value to each entry of the Presented Disclosed Claims Set.

One possible concrete representation of the intermediate data structure for the Digest To Disclosed Claim Map is a CBOR map with the hash of the bstr-encoded-salted data structure (from the CDDL) as the map key and its value as the contents of the corresponding salted-entry data structure.

3. The Verifier constructs an empty CBOR map called the Validated Disclosed Claims Set, and initializes it with all mandatory to disclose claims from the verified Presented Disclosed Claims Set.
4. Next, the Verifier performs a depth-first traversal of the Presented Disclosed Claims Set and Validated Disclosed Claims Set, using the Digest To Disclosed Claim Map to insert claims into the Validated Disclosed Claims Set when they appear in the Presented Disclosed Claims Set.
5. The Verifier repeats the fourth step if the previous iteration resulted in any new Presented Disclosed Claims.
6. If there remain unused claims in the Digest To Disclosed Claim Map at the end of this procedure the SD-CWT MUST be considered invalid. Likewise, if this algorithm results in any duplicate CBOR map keys, the entire SD-CWT MUST be considered invalid.

Note: If there are remaining digests without corresponding disclosures, this means that either the holder intentionally did not disclose a claim, or that the digest is a decoy digest
Section 10.

Appendix G. Document History

Note: RFC Editor, please remove this entire section on publication.

G.1. draft-ietf-spice-sd-cwt-06

- * Refactored deterministic draft generation code (PR#152).
- * Added pointer to a Python implementation (PR#155).
- * Added decoy digests (PR#157).
- * Addressed early IANA feedback about the escalation process (PR#160).

- * Used the CoAP Content Type in the examples instead of the text strings "application/sd-cwt" and "application/kb+cwt" (PR#162).
- * Added a section on specific CBOR encoding and data model considerations (PR#163).
- * Swapped the order of Sections 5 and 6 (PR#167).
- * Split the CDDL definitions for payload maps in Issued CWT and pre-issued (PR#168).
- * Used the IANA early assigned values in the draft (PR#169).
- * Defined the To Be Decoy tag (PR#171).
- * Made use of the CoAP Content Type a SHOULD (PR#172).
- * Made the draft generation code agnostic to hash algorithm (PR#173)
- * Added time claim verification rules and security considerations (PR#175)
- * Instead of an empty array, sd_claims is now omitted if empty (PR#176)
- * Update the COSE header values to use their newly assigned values (also PR#176)
- * Fix some kramdown-xml bracket errors (PR#177)
- * Add IANA Considerations for To Be Decoy tag (PR#180)
- * Clarify that remaining redacted claims are removed in validated claim set (PR#181)
- * Restrict floating point dates to -2^{53} to 2^{53} (PR#182)
- * Rename CDDL production using a standard prelude name (PR#183)

G.2. draft-ietf-spice-sd-cwt-05

- * Added this change log (PR#150)
- * Moved non-normative validation algorithm to an appendix (PR#149)
- * Added appendix describing mapping to RATS concepts (#147)
- * Provided guidance on choice of AEAD algorithm (#148)

- * Fixed algorithm in COSE key examples (#145)
- * Updated contact information (PR#142, PR #150)
- * Removed SPICE from the title of the document (PR#139)
- * Made clear extent to which verifiers cannot process unknown claims (PR#138)
- * Sorted CBOR map keys in examples to facilitate use as test vectors (PR#135)
- * Consistently use term "tag" in context of AEAD algorithms (PR#134)
- * Improved AASVG diagram in Terminology section (PR#129)

G.3. draft-ietf-spice-sd-cwt-04

- * Place value before claim name in disclosures
- * Use CBOR simple value 59 for the redacted_key_claims
- * Greatly improved text around AEAD encrypted disclosures
- * Applied clarifications and corrections suggested by Mike Jones.
- * Do not update CWT [RFC8392].
- * Use application/sd-cwt media type and define +sd-cwt structured suffix.
- * Made SHA-256 be the default sd_alg value.
- * Created Verifiable Credential Type Identifiers registry.
- * Corrected places where Claim Name was used when what was meant was Claim Key.
- * Defined the To Be Redacted CBOR tag
- * In the SD-KBT, iss and sub are now forbidden
- * Clarified text about aud
- * Described Trust Lists
- * EDN Examples are now in deterministic order

- * Expressed some validation steps as a list
- * Clarified handling of nested claims
- * Fixed the handling of the to be registered items in the CDDL; made CDDL self consistent
- * Fixed some references

G.4. draft-ietf-spice-sd-cwt-03

- * remove bstr encoding from sd_claims array (but not the individual disclosures)
- * clarify which claims are optional/mandatory
- * correct that an SD-CWT may have zero redacted claims
- * improve the walkthrough of computing a disclosure
- * clarify that duplicate map keys are not allowed, and how tagged keys are represented.
- * added security considerations section (#42) and text about privacy and linkability risks (#43)
- * register SD-CWT and SD-KBT as content formats in CoAP registry (#39)
- * updated media types registrations to have more useful contacts (#44)
- * build most of the values (signatures/salts/hashes/dates) in the examples automatically using a script that implements SD-CWT
- * regenerate all examples with correct signatures
- * add nested example
- * add optional encrypted disclosures

G.5. draft-ietf-spice-sd-cwt-02

- * KBT now includes the entire SD-CWT in the Confirmation Key CWT (kcwt) existing COSE protected header. Has algorithm now specified in new sd_alg COSE protected header. No more sd_hash claim. (PR #34, 32)

- * Introduced tags for redacted and to-be-redacted claim keys and elements. (PR#31, 28)
- * Updated example to be a generic inspection certificate. (PR#33)
- * Add section saying SD-CWT updates the CWT spec (RFC8392). (PR#29)

G.6. draft-ietf-spice-sd-cwt-01

- * Added Overview section
- * Rewritten the main normative section
- * Made redacted_claim_keys use an unlikely to collide claim key integer
- * Make cnonce optional (it now says SHOULD)
- * Made most standard claims optional.
- * Consistently avoid use of bare term "key" - to make crypto keys and map keys clear
- * Make clear issued SD-CWT can contain zero or more redactions; presented SD-CWT can disclose zero, some, or all redacted claims.
- * Clarified use of sd_hash for issuer to holder case.
- * Lots of editorial cleanup
- * Added Rohan as an author and Brian Campbell to Acknowledgements
- * Updated implementation status section to be BCP205-compatible
- * Updated draft metadata

G.7. draft-ietf-spice-sd-cwt-00

- * Initial working group version based on draft-prorock-spice-cose-sd-cwt-01.

Acknowledgments

The authors would like to thank those that have worked on similar items for providing selective disclosure mechanisms in JSON, especially: Brent Zundel, Roy Williams, Tobias Looker, Kristina Yasuda, Daniel Fett, Brian Campbell, Oliver Terbu, and Michael B. Jones.

The authors would like to thank the following individuals for their contributions to this specification: Michael B. Jones.

Contributors

Michael B. Jones
Self-Issued Consulting
United States
Email: michael_b_jones@hotmail.com
URI: <https://self-issued.info/>

Authors' Addresses

Michael Prorock
mesur.io
Email: mprorock@mesur.io

Orie Steele
Tradeverifyd
Email: orie@orl3.io

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

Rohan Mahy
Email: rohan.ietf@gmail.com