Secure Patterns for Internet CrEdentials
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

M. Prorock
mesur.io
O. Steele
Transmute
H. Birkholz
Fraunhofer SIT
R. Mahy
Rohan Mahy Consulting Services

7 July 2025

SPICE SD-CWT
draft-ietf-spice-sd-cwt-04

Abstract

This specification describes a data minimization technique for use with CBOR Web Tokens (CWTs). The approach is based on 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 CrEdentials 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/.

Copyright Notice

Table of Contents

1. Introduction

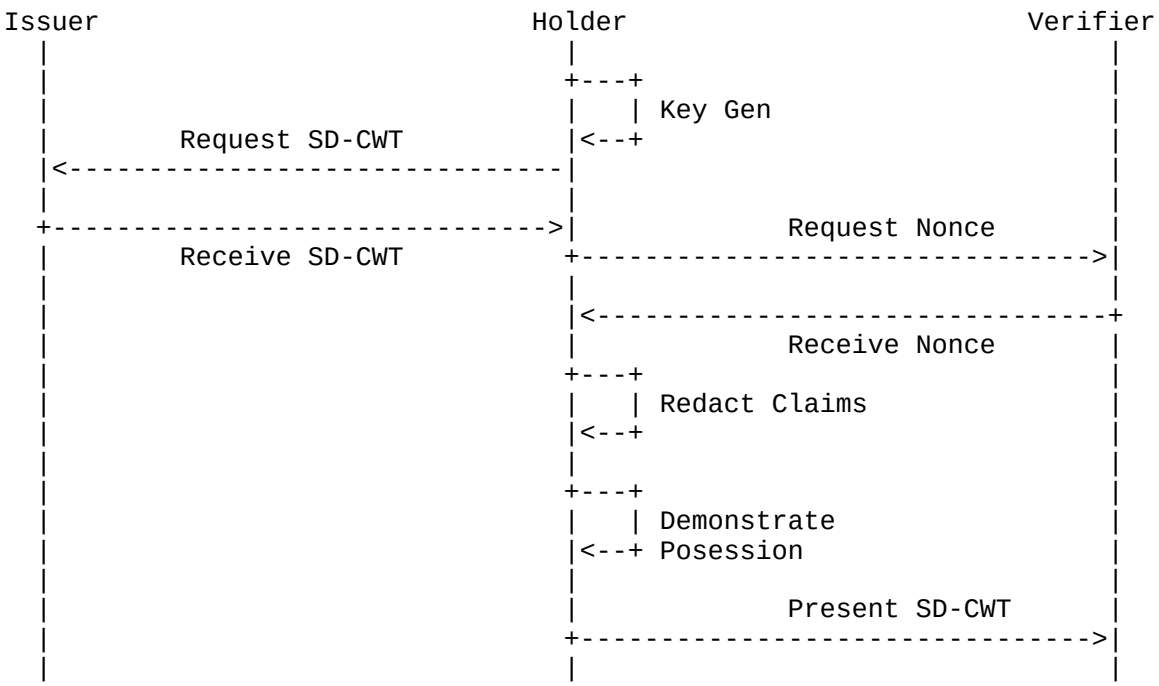
This specification creates a new format based on the CBOR Web Token (CWT) specification [RFC8392], enabling the Holder of a CWT to disclose or redact special claims marked as selectively disclosable by the Issuer of a CWT. The approach is modeled after SD-JWT [I-D.draft-ietf-oauth-selective-disclosure-jwt], with changes to align with conventions from CBOR Object Signing and Encryption (COSE) [RFC9052] and CWT. This specification enables Holders of CWT-based credentials to prove the integrity and authenticity of selected attributes asserted by an Issuer about a Subject to a Verifier.

Although techniques such as one time use and batch issuance can improve the confidentiality and security characteristics of CWT-based credential protocols, SD-CWTs remain traceable. 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. Credential types are distinguished by their attributes, for example, a license to operate a vehicle and a license to import a product will contain different attributes. The specification of credential types is out of scope for this specification, and the examples used in this specification are informative.

SD-CWT operates on CWT Claims Sets as described in [RFC8392]. CWT Claims Sets contain Claim Keys and Claim Values. SD-CWT enables Issuers to mark certain Claim Keys or Claim Values mandatory or optional for a Holder of a CWT to disclose. A Verifier that does not understand selective disclosure at all cannot process redacted Claim Keys sent by the Holder. However, Claim Keys and Claim Values that are not understood remain ignored, as described in Section 3 of [RFC8392].

1.1. High-Level Flow

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



This diagram captures the essential details necessary 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. However, the following guidance is generally recommended, regardless of protocol or transport.

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

2. 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.

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.

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 all claims are marked as mandatory to disclose 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.

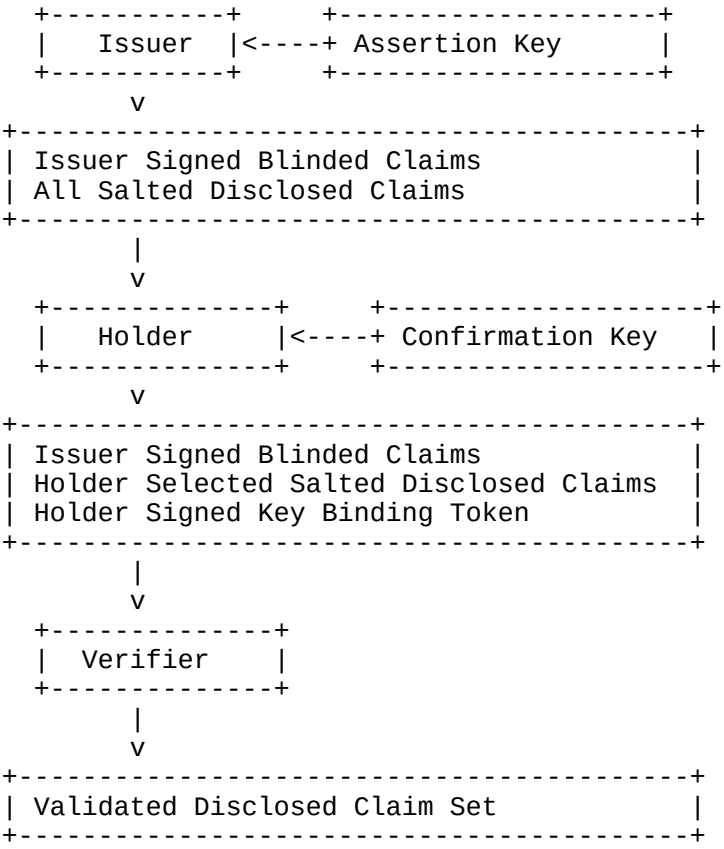
Redacted Claim Key: The hash of a claim redacted from a map data structure.

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

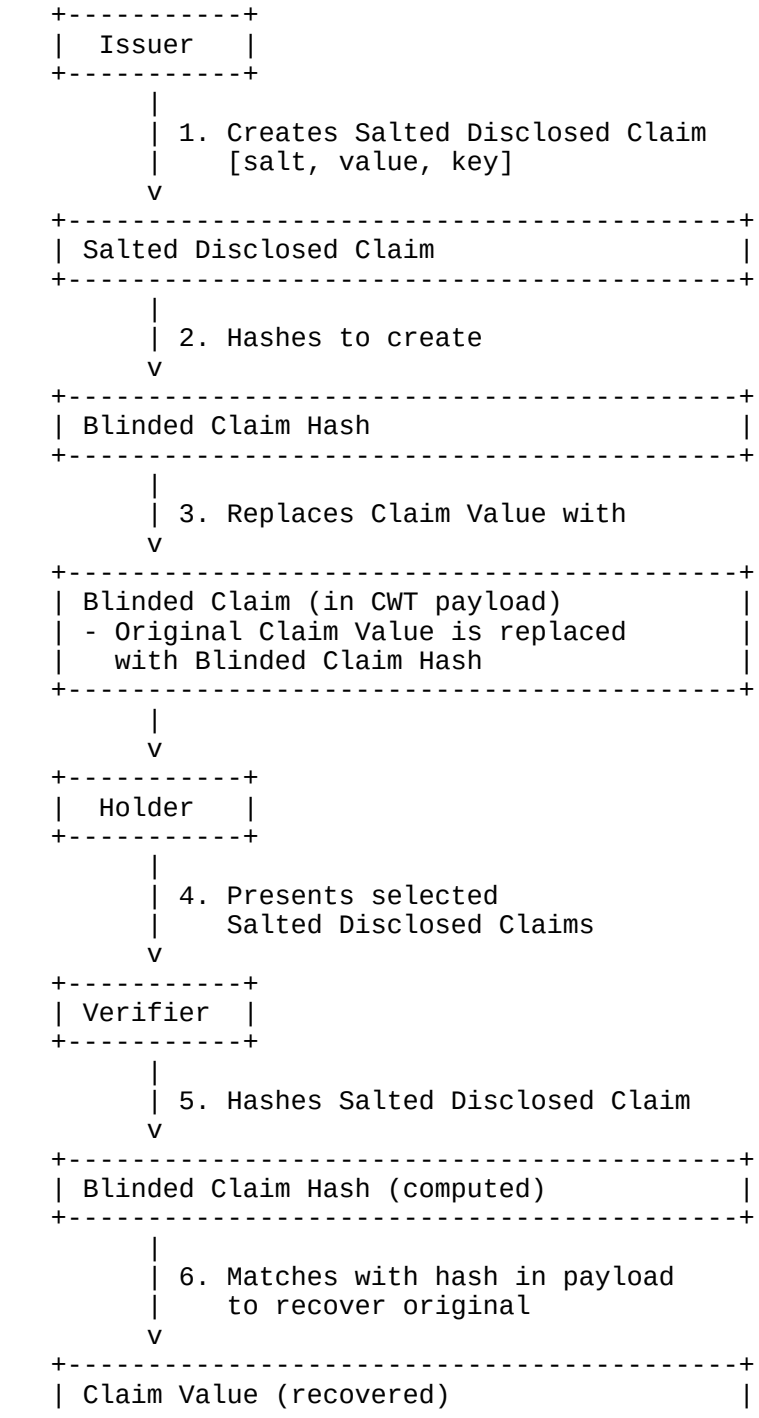
Presented Disclosed Claims Set: The CBOR map containing zero or more Redacted Claim Keys or Redacted Claim Elements.

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.

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



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


Issuer
1. Creates Salted Disclosed Claim
[salt, value, key]
Salted Disclosed Claim
2. Hashes to create
Blinded Claim Hash
3. Replaces Claim Value with
Blinded Claim (in CWT payload)
- Original Claim Value is replaced
with Blinded Claim Hash
Holder
4. Presents selected
Salted Disclosed Claims
Verifier
5. Hashes Salted Disclosed Claim
Blinded Claim Hash (computed)
6. Matches with hash in payload
to recover original
Claim Value (recovered)

```
+----------------------------------------+
```

3. Overview of Selective Disclosure CWT

3.1. A CWT without Selective Disclosure

Below is the payload of a standard CWT not using selective disclosure. It consists of standard CWT claims, the Holder confirmation key, and five specific custom claims. The payload is shown below in CBOR Extended Diagnostic Notation (EDN) [I-D.ietf-cbor-edn-literals]. Note that some of the CWT claim map keys shown in the examples have been invented for this example and do not have 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"
    }
}
```

The custom 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 (custom) 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 : "application/sd-cwt",
    / sd_alg / 18 : -16  / SHA256 /
  } >>,
  / CWT unprotected / {
    / sd_claims / 17 : [ / these are all the disclosures /
        <<[
            /salt/   h'bae611067bb823486797da1ebbb52f83',
            /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
               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'af375dc3fba1d082448642c00be7b2f7
            bb05c9d8fb61cfc230ddfdfb4616a693'
      ]
    } >>,
    / CWT signature / h'9c9022e57adb33c853f30b6e8a590f40
                       6ca55849d7b8cd2a2519d3aec03e61b9
                       ef0ecd85fe96103f916f58d73cd2f775
                       4c390401945f0683b144d3504e500f94
                       d30433c3445417dc3c920f7a155548e9
                       1994601827d0a46ead66ff450485e85f'
  ])
```

Figure 1: 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'bae611067bb823486797da1ebbb52f83',
            /value/  "ABCD-123456",
            /claim/  501   / inspector_license_number /
        ]>>,
```

Figure 2: 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)
      bae611067bb823486797da1ebbb52f83  # 16-byte salt
   6b                                   # text(11)
      414243442d313233343536            # "ABCD-123456"
   19 01f5                              # unsigned(501)
```

Figure 3: 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 4: 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 claims that are array elements are handled slightly differently, as described in Section 6.1.

```
    / redacted_claim_keys / simple(59) : [
        / redacted inspector_license_number /
        h'af375dc3fba1d082448642c00be7b2f7
          bb05c9d8fb61cfc230ddfdfb4616a693'
        / ... next redacted claim at the same level would go here /  ],
```

Figure 5: redacted inspector_license_number claim in the issued CWT payload

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.

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'bae611067bb823486797da1ebbb52f83',
            /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 /
        ]>>,
    ]
```

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:  "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'd895729e72a3a7c801d5a20e9daf5103
                      27858aecbb39b8b2e4bc11cbbd625ea8
                      c60b78da31fc9762c46b7cd61094d047
                      5ff1f19a7496cde53ab11600a5859d10'
])   / end of kbt /
```

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. Differences from the CBOR Web Token Specification

The CBOR Web Token Specification (Section 1.1 of [RFC8392]), uses text strings, negative integers, and unsigned integers as map keys. This specification also allows the CBOR simple value registered in this specification in Section 17.2, and CBOR tagged integers and text strings as map keys. As in CWTs, CBOR maps used in an SD-CWT or SD-KBT also cannot have duplicate keys. (An integer or text string map key is a distinct key from a tagged map key that wraps the corresponding integer or text string value).

When sorted, map keys in CBOR are arranged in bytewise lexicographic order of the key's deterministic encodings (see Section 4.2.1 of [RFC8949]). So, an integer key of 3 is represented in hex as 03, an integer key of -2 is represented in hex as 21, and a tag of 60 wrapping a 3 is represented in hex as D8 3C 03

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 key are not permitted.

Variability in serialization requirements impacts privacy.

See Section 16 for more details on the privacy impact of serialization and profiling.

6. 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 include the protected header parameter typ [RFC9596] with a value declaring that the object is an SD-CWT. This value MAY be the string content type value application/sd-cwt, the uint Constrained Application Protocol (CoAP) [RFC7252] content-format value TBD11, 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.

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

Optionally the salted Claim Values (and often 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 in the unprotected header is an empty array.

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.

6.1. Types of Blinded Claims

Salted Disclosed Claims for named claims are structured as a 128-bit salt, the disclosed value, and the name of the redacted element. For Salted Disclosed Claims of items in an array, the name is omitted.

```
salted = 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
]

; a collection of Salted Disclosed Claims
salted-array = [ +bstr .cbor salted ]
```

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 type TBD4 registered for that purpose (with the requested value of 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 TBD5 (requested tag number 60).

```
; redacted_claim_element = #6.<TBD5>( bstr ) -- RFC 9682 syntax
redacted_claim_element = #6.60( bstr )
```

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 13.2.

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.

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.

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 an empty array. The payload also MUST include a key confirmation element (cnf) [RFC8747] for the Holder’s public key.

In an SD-CWT, either the subject sub / 2 claim MUST be present, or the redacted form of the subject MUST be present. The iss / 1 claim SHOULD be present unless the protected header contains a certificate or certificate-like entity that fully identifies the Issuer. All other standard CWT claims (aud / 3, exp / 4, nbf / 5, iat / 6, and cti / 7) are OPTIONAL. The cnonce / 39 claim is OPTIONAL. The cnf / 8 claim, the cnonce / 39 claim, and the standard claims other than the subject MUST NOT be redacted. Any other claims are OPTIONAL and MAY be redacted. Profiles of this specification MAY specify additional claims that MUST, MUST NOT, and MAY be redacted.

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 5. 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;
* every blinded claim hash (some of which may be nested as in Section 13.2) 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) ^ => "application/sd-cwt" / TBD11,
   &(alg: 1) ^ => int,
   &(sd_alg: TBD2) ^ => int,        ; -16 for sha-256
   ? &(sd_aead: TBD7) ^ => uint .size 2
   * key => any
}

sd-unprotected = {
   ? &(sd_claims: TBD1) ^ => salted-array,
   ? &(sd_aead_encrypted_claims: TBD6) ^ => aead-encrypted-array,
   * key => any
}

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) ^ => int,  ; 1883000000
    ? &(nbf: 5) ^ => int,  ; 1683000000
    ? &(iat: 6) ^ => int,  ; 1683000000
    ? &(cti: 7) ^ => bstr,
      &(cnf: 8) ^ => { * key => any }, ; key confirmation
    ? &(cnonce: 39) ^ => bstr,
    ;
    ? &(redacted_claim_keys: REDACTED_KEYS) ^ => [ * bstr ],
    * key => any
}
```

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.

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 the iat (issued at) claim. The protected header of the SD-KBT MUST include the typ header parameter with the value application/kb+cwt or the uint value of TBD12.

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;
* the Holder's disclosure is linked to the creation time (iat) of the key binding.

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) ^ => "application/kb+cwt" / TBD12,
   &(alg: 1) ^ => int,
   &(kcwt: 13) ^ => sd-cwt-issued,
   * key => any
}

kbt-unprotected = {
   * key => any
}

kbt-payload = {
      &(aud: 3) ^ => tstr, ; "https://verifier.example/app"
    ? &(exp: 4) ^ => int,  ; 1883000000
    ? &(nbf: 5) ^ => int,  ; 1683000000
      &(iat: 6) ^ => int,  ; 1683000000
    ? &(cnonce: 39) ^ => bstr,
    * key => any
}
```

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

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].

3. The Verifier extracts the confirmation key from the cnf claim in the SD-CWT payload.

4. Using the confirmation key, the Verifier validates the SD-KBT as described in Section 7.2 of [RFC8392].

5. Finally, the Verifier MUST extract and decode the disclosed claims from the sd_claims header parameter in the unprotected header of the SD-CWT. 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. 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 could be: { &(digested-salted-disclosed-claim) => salted }

   a. 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.

   b. Next, the Verifier performs a breadth first or 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. By performing these steps, the recipient can cryptographically verify the integrity of the protected claims and verify they have not been tampered with.

   c. 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.

   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.

6. 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.

7. Otherwise, the SD-CWT is considered valid, and the Validated Disclosed Claims Set is now a CWT Claims Set with no claims marked for redaction.

8. Further validation logic can be applied to the Validated Disclosed Claims Set, just as it might be applied to a validated CWT Claims Set.

10. Decoy Digests

TODO

11. 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.

11.1. 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) . The second header parameter (sd_aead_encrypted_claims) contains a list of AEAD encrypted disclosures. Taking the first example disclosure from above:

```
        <<[
            /salt/   h'bae611067bb823486797da1ebbb52f83',
            /value/  "ABCD-123456",
            /claim/  501   / inspector_license_number /
        ]>>,
```

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.

```
/ sd_aead_encrypted_claims / 19 : [ / AEAD encrypted disclosures /
    [
        / nonce /      h'95d0040fe650e5baf51c907c31be15dc',
        / ciphertext / h'208cda279ca86444681503830469b705
                         89654084156c9e65ca02f9ac40cd62b5
                         a2470d',
        / tag /        h'1c6e732977453ab2cacbfd578bd238c0'
    ],
    ...
]
```

In the example above, the key in hex is a061c27a3273721e210d031863ad81b6.

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,              ; nonce value
  bstr,              ; the ciphertext output of a bstr-encoded-salted
                     ;   with a matching salt
  bstr               ; the corresponding authentication tag
]
;bstr-encoded-salted = bstr .cbor salted
```

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.

12. 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].

13. Examples

13.1. 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 : "application/sd-cwt",
        / sd_alg / 18 : -16  / SHA256 /
      } >>,
      / CWT unprotected / {
        / sd_claims / 17 : [ / these are the disclosures /
            <<[
                /salt/   h'bae611067bb823486797da1ebbb52f83',
                /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'af375dc3fba1d082448642c00be7b2f7
              bb05c9d8fb61cfc230ddfdfb4616a693'
        ]
      } >>,
      / CWT signature / h'9c9022e57adb33c853f30b6e8a590f40
                          6ca55849d7b8cd2a2519d3aec03e61b9
                          ef0ecd85fe96103f916f58d73cd2f775
                          4c390401945f0683b144d3504e500f94
                          d30433c3445417dc3c920f7a155548e9
                          1994601827d0a46ead66ff450485e85f'
    ]),
    / end of issuer SD-CWT /
    / typ /   16:  "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'd895729e72a3a7c801d5a20e9daf5103
                        27858aecbb39b8b2e4bc11cbbd625ea8
                        c60b78da31fc9762c46b7cd61094d047
                        5ff1f19a7496cde53ab11600a5859d10'
])   / end of kbt /
```

Figure 6: An EDN Example

13.2. 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: [                      / inspection history log /
        {
            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 /
            }
        },
    ]
}
```

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'2e9a833949c163ce845813c258a8f13c',
        /value/  {
                     500: true,
                     502: 17183928,
                     simple(59): [
                       h'3fc9748e00684e6442641e58ea965468
                         085024da253ed46b507ae56d4c204434',
                       h'a5124745703ea9023bf92a2028ba4547
                         b830ce9705161eaad56729cab8e1d807'
                     ]
                 }   / inspection 17-Jan-2023 /
    ]>>,
    <<[
        /salt/   h'bae611067bb823486797da1ebbb52f83',
        /value/  "ABCD-123456",
        /claim/  501   / inspector_license_number /
    ]>>,
    <<[
        /salt/   h'd5c7494eb16a8ff11fba507cbc7c816b',
        /value/  {
                     1: "us",
                     simple(59): [
                       h'3bf93977377099c66997303ddbce67b4
                         ca7ee95d2c8cf2b8b45f451362493460',
                       h'231e125d192de099e91bc59e2ae914f0
                         c891cbc3329b7fea70a3aa636c87a0a4'
                     ]
                 },
```

```
        /claim/  503   / San Francisco location /
    ]>>,
    <<[
        /salt/   h'52da9de5dc61b33775f9348b991d3d78',
        /value/  "ca",
        /claim/  2   / region=California /
    ]>>,
    <<[
        /salt/   h'9adcf14141f8607a44a130a4b341e162',
        /value/  {
                     500: true,
                     502: 1549560720,
                     simple(59): [
                       h'94d61c995d4fa25ad4d3cc4752f6ffaf
                         9e67f7f0b4836c8252a9ad23c20499f5',
                       h'4ff0ecad5f767923582febd69714f3f8
                         0cb00f58390a0825bc402febfa3548bf'
                     ]
                 }   / inspection 7-Feb-2019 /
    ]>>,
    <<[
        /salt/   h'591eb2081b05be2dcbb6f8459cc0fe51',
        /value/  "DCBA-101777",
        /claim/  501   / inspector_license_number /
    ]>>,
    <<[
        /salt/   h'95b006410a1b6908997eed7d2a10f958',
        /value/  {
                     1: "us",
                     simple(59): [
                       h'2bc86e391ec9b663de195ae9680bf614
                         21666bc9073b1ebaf80c77be3adb379f',
                       h'e11c93b44fb150a73212edec5bde46d3
                         d7db23d0d43bfd6a465f82ee8cf72503'
                     ]
                 },
        /claim/  503   / Denver location /
    ]>>,
]
```

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: [                      / inspection history log /
        {
            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 /
            }
        }
    ]
}
```

14. To Be Redacted Tag Definition

In order to indicate specific claims that should be redacted in a Claim Set, this specification defines a new CBOR tag "To be redacted". It 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.

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. Signature based linkability can be mitigated by leveraging batch issuance of single-use tokens, at a credential management complexity cost. 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.

Certificate transparency, as described in [RFC9162], or key transparency, as described in [I-D.draft-ietf-keytrans-protocol], can help detect and prevent such attacks by: - Enabling public observation of all issued certificates or key bindings - Detecting unauthorized or fraudulent bindings between verification keys and Issuer identifiers - Providing cryptographic proof of inclusion for legitimate keys - Creating an append-only audit trail that makes key substitution attacks discoverable

Verifiers SHOULD leverage transparency mechanisms where available to validate that the issuer’s keys have not been compromised or fraudulently substituted.

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.

16.3. Threat Model Development Guidance

This section provides guidance for developing threat models when applying SD-CWT to specific use cases. It is NOT a threat model itself, but rather a framework to help implementers create appropriate threat models for their particular contexts. Each use case will have unique security characteristics that MUST be analyzed before determining the applicability of SD-CWT-based credential types.

The following non-exhaustive list of questions and considerations should guide the development of a use-case-specific threat model:

1. Has there been a t-closeness, k-anonymity, and l-diversity assessment (see [t-Closeness]) assuming compromise of the one or more Issuers, Verifiers or Holders, for all relevant credential types?

2. Issuer questions:

    a. How many Issuers exist for the credential type?

    b. Is the size of the set of Issuers growing or shrinking over time?

    c. For a given credential type, will subjects be able to hold instances of the same credential type from multiple Issuers, or just a single Issuer?

    d. Does the credential type require or offer the ability to disclose a globally unique identifier?

    e. Does the credential type require high precision time or other claims that have sufficient entropy such that they can serve as a unique fingerprint for a specific subject?

    f. Does the credential type contain Personally Identifiable Information (PII), or other sensitive information that might have value in a market?

3. Holder questions:

    a. What steps has the Holder taken to improve their operation security regarding presenting credentials to verifiers?

    b. How can the Holder be convinced the Verifier that received presentations is legitimate?

c. How can the Holder be convinced the Verifier will not share, sell, leak, or otherwise disclose the Holder’s presentations or Issuer or Holder signed material?

d. What steps has the Holder taken to understand and confirm the consequences resulting from their support for the aggregate-use of digital credential presentations?

4. Verifier questions:

a. How many Verifiers exist for the credential type?

b. Is the size of the set of Verifiers growing or shrinking over time?

c. Are the Verifiers a superset, subset, or disjoint set of the Issuers or subjects?

d. Are there any legally required reporting or disclosure requirements associated with the Verifiers?

e. Is there reason to believe that a Verifier’s historic data will be aggregated and analyzed?

f. Assuming multiple Verifiers are simultaneously compromised, what knowledge regarding subjects can be inferred from analyzing the resulting dataset?

5. Subject questions:

a. How many subjects exist for the credential type?

b. Is the size of the set of subjects growing or shrinking over time?

c. Does the credential type require specific hardware, or algorithms that limit the set of possible subjects to owners of specific devices or subscribers to specific services?

16.4. Random Numbers

Each salt used to protect disclosed claims MUST be generated independently from the salts of other claims. The salts MUST be generated from a source of entropy that is acceptable to the Issuer. 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.

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.

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: TBD1 (requested assignment 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: TBD2 (requested assignment 18)
* 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: TBD6 (requested assignment 19)

* Value Type: 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 11.1 of this specification

17.1.4. sd_aead

The following completed registration template per RFC8152 is provided:

* Name: sd_aead

* Label: TBD7 (requested assignment 20)

* Value Type: int

* Value Registry: IANA AEAD Algorithm number

* Description: The AEAD algorithm used for encrypting disclosures.

* Reference: Section 11.1 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: TBD4 (requested assignment 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 6.1 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: TBD3 (requested assignment 58)
* Data Item: (any)
* Semantics: An array claim element, or map key and value intended to be redacted.
* Specification Document(s): Section 14 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: TBD5 (requested assignment 60)
* Data Item: byte string
* Semantics: A selective disclosure redacted (array) claim element.
* Specification Document(s): Section 6.1 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: TBD6 (request assignment 11)
* Claim Value Type(s): bstr
* Change Controller: IETF
* Specification Document(s): Section 12 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 6 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 6 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	-	TBD11	Section 6 of this specification
application/kb+cwt	-	TBD12	Section 8.1 of this specification

Table 1: New CoAP Content Formats

If possible, TBD11 and TBD12 should be assigned in the 256..9999 range.

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
0-9999	Specification Required
10000-64999	First Come First Served
65000-65535	Experimental Use (no operational use)

Table 2

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 is followed 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-18, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-edn-literals-18>.

[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>.

[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-12, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-cde-12>.

[I-D.draft-ietf-keytrans-protocol]
McMillion, B. and F. Linker, "Key Transparency Protocol", Work in Progress, Internet-Draft, draft-ietf-keytrans-protocol-02, 6 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-keytrans-protocol-02>.

[I-D.draft-ietf-oauth-sd-jwt-vc]
Terbu, O., Fett, D., and B. Campbell, "SD-JWT-based Verifiable Credentials (SD-JWT VC)", Work in Progress, Internet-Draft, draft-ietf-oauth-sd-jwt-vc-10, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-sd-jwt-vc-10>.

[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>.

[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>.

[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>.

[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>.

[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>.

[t-Closeness] "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity", 4 June 2007, <https://ieeexplore.ieee.org/document/4221659>.

Appendix A. Complete CDDL Schema

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

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

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

sd-protected = {
   &(typ: 16) ^ => "application/sd-cwt" / TBD11,
   &(alg: 1) ^ => int,
   &(sd_alg: TBD2) ^ => int,        ; -16 for sha-256
   ? &(sd_aead: TBD7) ^ => uint .size 2
   * key => any
}

kbt-protected = {
   &(typ: 16) ^ => "application/kb+cwt" / TBD12,
   &(alg: 1) ^ => int,
```

```
      &(kcwt: 13) ^ => sd-cwt-issued,
      * key => any
   }

   sd-unprotected = {
      ? &(sd_claims: TBD1) ^ => salted-array,
      ? &(sd_aead_encrypted_claims: TBD6) ^ => aead-encrypted-array,
      * key => any
   }

   kbt-unprotected = {
      * key => any
   }

   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) ^ => int,  ; 1883000000
       ? &(nbf: 5) ^ => int,  ; 1683000000
       ? &(iat: 6) ^ => int,  ; 1683000000
       ? &(cti: 7) ^ => bstr,
         &(cnf: 8) ^ => { * key => any }, ; key confirmation
       ? &(cnonce: 39) ^ => bstr,
       ;
       ? &(redacted_claim_keys: REDACTED_KEYS) ^ => [ * bstr ],
       * key => any
   }

   kbt-payload = {
         &(aud: 3) ^ => tstr, ; "https://verifier.example/app"
       ? &(exp: 4) ^ => int,  ; 1883000000
       ? &(nbf: 5) ^ => int,  ; 1683000000
         &(iat: 6) ^ => int,  ; 1683000000
       ? &(cnonce: 39) ^ => bstr,
       * key => any
   }

   salted-array = [ +bstr .cbor salted ]
   salted = salted-claim / salted-element / decoy
   salted-claim = [
     bstr .size 16,     ; 128-bit salt
     any,               ; claim value
     (int / text)       ; claim name
   ]
   salted-element = [
     bstr .size 16,     ; 128-bit salt
```

```
    any                ; claim value
  ]
  decoy = [
    bstr .size 16      ; 128-bit salt
  ]
  ;bstr-encoded-salted = bstr .cbor salted

  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
  ]

  header_map = {
      * key => any
  }
  empty_or_serialized_map = bstr .cbor header_map / bstr .size 0

  key = int / text
  TBD1 = 17
  TBD2 = 18
  TBD6 = 19
  TBD7 = 20

  ;TBD3 = 58;  CBOR tag wrapping to-be-redacted keys or elements

  TBD11 = 298
  TBD12 = 299

  ; REDACTED_KEYS is to be used in CDDL payloads that are meant to
  ; convey that a map key is redacted.
  REDACTED_KEYS = #7.59  ; #7.<TBD4>
  ;TBD4 = 59          ; for CBOR simple value 59

  ; redacted_claim_element is to be used in CDDL payloads that contain
  ; array elements that are meant to be redacted.
  redacted_claim_element = #6.60( bstr .size 16 )  ; #6.<TBD5>(bstr)
  ;TBD5 = 60; CBOR tag wrapping redacted_claim_element
```

Figure 7: 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.

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 : -9, /ESP256/
  /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": "hVTrJ13Nb70cesZBqiyQ2SAi_Q0wJLWvGMfMYa1Sei0",
  "y": "TceuLGd-ltDMgll2Vc6S1VA_VCk9h4ddHnnOR3AZQ0M",
  "d": "V1moblm7OwAt3kZ9pLUvPQbmws1DlFbPBIW5uGQpTOU"
}
```

Input to Holder public JWK thumbprint (ignore line breaks)

```
{"crv":"P-256","kty":"EC","x":"hVTrJ13Nb70cesZBqiyQ2SAi_Q0wJLWvGMfMYa1S
ei0","y":"TceuLGd-ltDMgll2Vc6S1VA_VCk9h4ddHnnOR3AZQ0M"}
```

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

```
59143645b6394582422317c340bf5a825f5f15209856ee17a1ca9beb37ab7353
```

Holder public JWK thumbprint

```
WRQ2RbY5RYJCIxfDQL9agl9fFSCYVu4Xocqb6zerc1M
```

Holder public key in PEM format

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEhVTrJ13Nb70cesZBqiyQ2SAi/Q0w
JLWvGMfMYa1Sei1Nx64sZ36W0MyCWXZVzpLVUD9UKT2Hh10eec5HcBlDQw==
-----END PUBLIC KEY-----
```

Holder private key in PEM format

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgV1moblm7OwAt3kZ9
pLUvPQbmws1DlFbPBIW5uGQpTOWhRANCAASFVOsnXc1vvRx6xkGqLJDZICL9DTAk
ta8Yx8xhrVJ6LU3HrixnfpbQzIJZdlXOktVQP1QpPYeHXR55zkdwGUND
-----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-HfltMOm3GelpdxrMHtyjDclkm
8qvl-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+SFLJ1CVIsFZ52T4cR4RYJVU
-----END PUBLIC KEY-----
```


Issuer private key in PEM format


```
-----BEGIN PRIVATE KEY-----
MIG2AgEAMBAGByqGSM49AgEGBSuBBAAiBIGeMIGbAgEBBDBxxU0iIZN+phLbEiHw
0933cck4HE475B1aoKidaF8Jz+90xLvxBHg/1X6HqyJ9B0yhZANiAATDF5iwx4hf
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)

Appendix E. Document History

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

E.1. 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

E.2. 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

E.3. 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)

E.4. 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

E.5. 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
Transmute
Email: orie@transmute.industries

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

Rohan Mahy
Rohan Mahy Consulting Services
Email: rohan.ietf@gmail.com