SPICE SD-CWT

draft-ietf-spice-sd-cwt-03

M. Prorock , O. Steele , H. Birkholz , R. Mahy
IETF 122 Bangkok
17 March 2025

Agenda

- Recap on selective disclosure (Rohan)
- Changes since IETF 121 (Rohan)
- Open Issues (Henk)
- Next Steps (Henk)

Recap of Selective Disclosure

Intro to Selective Disclosure

- Solution statement
  - Allow a **Holder** to get a CBOR Web Token from an **Issuer** and then safely blind (“redact”) or reveal (“disclose”) some of its claims to a **Verifier**
- Holder can disclose *all, some, or none* of the disclosable claims.
- Issuer **dictates** which Claims can be blinded or disclosed
- Can include **DECOYS!**

- Every SD-CWT is a valid CWT, but not vice versa
  - Relaxes valid types of map keys to include simple values and tagged values.
- How Issuer decides which claims to selectively disclose is out-of-scope. Issuer does not have to make claims disclosable.
- Includes Holder’s public key or reference to it (`cnf`)

What can be disclosed?

No: Any CWT claims that are part of the validation of the CWT (iss, aud, exp, nbf, iat, cti, cnf, nonce), **except** the subject.

Yes: The subject; Any other claims at *any level of hierarchy*; Any array items of any CWT claims (ex: `swversion` below).

```
{
   / sub / 1: "https://device.holder.example",
   / cnf / 8 : {
      / ckt / 5 : h'496bd8a…3bda88fadd1669da253ec'
   },
   …
   / inspection passed /  500 : true,
   / inspector license /  501 : "ABCD-123456",
   / swversion / 271 : [
      "3.5.5",
      "4.1.7"
   ],
   / inspection location /  503 : {
      "country"     : "us",          / United States /
      "region"      : "ca",          / California /
      "locality"    : "San Francisco",
      "postal_code" : "94188"
}
```

Forming the disclosures — claim keys+values

For each CWT claim key, generate a random salt of at least 16-bytes.

Make a CBOR array with the salt and the selectively disclosed info

```
<<[
  /salt/ h'8d5c15fa86265d8ff77a0e92720ca837',
  /claim/  500,   / inspection_passed /
  /value/  true
]>>
```

```
  56 83                                   # bytes(22) array(3)
     50 8D5C15FA86265D8FF77A0E92720CA837     # bytes(16)
     19 01F4                              # unsigned(500)
     F5                                   # true
```

Forming the disclosures — Array items

For each array item, generate a random salt of at least 16-bytes.

Make a CBOR array with the salt and the selectively disclosed info

```
<<[
  /salt/ h'86c84b9c3614ba27073c7e5a475a2a13',
  /value/  "4.1.7"
]>>
```

```
58 18  82                               # bytes(24) array(2)
   50 86C84B9C3614BA27073C7E5A475A2A13    # bytes(16)
   65 342E312E37                        # text(5) "4.1.7"
```

Hashing the Disclosures

The **hash** of each disclosure becomes the **redacted form** of that claim, array element (or decoy).

These hashes are placed in the *payload* of the CWT, which is signed by the issuer. (Example syntax)

```
/ cose-sign1 / 18([
  / protected   / <<{
    / key / 4 : h'496b…53ec', # public key thumbprint
    / algorithm / 1 : -7,     # ES256
  }>>,
  / unprotected / {},
  / payload     /<<{
    / iss / 1 : "https://issuer.example",
    / sub / 2 : "https://subject.example",
    / iat / 6: 1729706150,
    / redacted_map_keys / simple(59) : [
      / hash of disclosure for inspection_passed / h'db6e…76e2',
      / hash of disclosure for license_number  / h'c234…3b04'
    ],
    /inspection_dates / 502: [ 1549560720, 60(h'7ad6…092f')
  }>>,
  / signature   / h'2544f2ed...5840893b'
])
```

Disclosing to a Verifier

Holder presents to a Verifier:

- the Issuer CWT
- any disclosures
- the Holder Key Binding Token (KBT) - *REQUIRED*

Can include an *audience* specific to the Verifier, and/or *nonce* scoped to the Verifier

Key Binding

Key Binding prevents copy and paste or replay attacks on selective disclosure.
It includes the Issued CWT and disclosures in the KBT protected header
The Holder proves possession of their private key and choice of disclosures.

```
/ sd_kbt    / 18([
  / protected / << {
      / alg / 1 : -7 / ES256 /,
      / typ / 16 : "application/kb+cwt"
      / kcwt / 13: 18([
            / protected / h’0fd2…6aa3’,
            / unprotected / {
                  /sd_claims/ 17: /disclosures go here /
            },
            / payload / h’ad49…1473’,
            /signature / h’28a9…83fd’
      ])
  } >>,
  / unprotected / {},
  / payload / << {
      / cnonce / 39      : h'e0a156...bb3f',
      / aud / 3    : "https://verifier.example",
      / iat / 6    : 1783000000,
  } >>,
  / signature / h'1237af2e...678945'
])
```

Nested disclosures

Disclosures can refer to nested data

```
{
  / name / 170: "Alice Smyth",
  / address / 187: {
    / region   / 2: "jp",
    / locality / 3: "Tokyo",
    / physical / 8: {
      / block / 1: "2 Chome-5-2",
      / bldg /  2: "Mitsubishi Building"
      / floor / 3: "B1F"
    }
  },
  / nationalities / 197: [
    "jp", "ie"
  ]
}
```

Nested disclosures

Disclosures can refer to nested data

```
{
  / name / 170: "Alice Smyth",
  / address / 187: {
    / region   / 2: "jp",
    / locality / 3: "Tokyo",
    / physical / 8: { simple(59) : [
      h'f41a...8246',
      h'1562...228d',
      h'277c...839f' ]
    }
  },
  / nationalities / 197: [
    "jp", "ie"
  ]
}
```

Nested disclosures

Disclosures can refer to nested data

```
{
  / name / 170: "Alice Smyth",
  / address / 187: { simple(59) : [
    h'ce5a...45fe',
    h'4d5a...cf14',
    h'08fc...a01b'




    ]
  },
  / nationalities / 197: [
    60(h'7ef9...2388'), 60(h'2782...65af')
  ]
}
```

278265af

Nested disclosures

Disclosures can refer to nested data

```
{ simple(59) : [
  h'6194...d892',
  h'56b8...0389',









  h'3ebc...c52e' ]


}
```

Forming Decoy disclosures

For each decoy, generate a random salt of at least 16-bytes.

Make a CBOR array with just the salt

Can be used to hide the information structure

```
<<[
  /salt/ h'613b281240e45f5d64b204555f91ba42'
]>>

  52 81                                  # bytes(18) array(1)
     50 613B281240E45F5D64B204555F91BA42    # bytes(16)
```

Changes since IETF 121

Changes Since 121 — Normative

- Updates CWT to allow tagged keys and simple value keys
- Use simple type 59 to indicate array of blinded claims (merged)
- sd_claims array is no longer bstr encoded, but kept the COSE style encoding of each individual disclosure (closed issue)
- Clarified optional/mandatory claims; dup keys not allowed; SD-CWT can have zero redacted claims.
- Encrypted disclosures

Now using simple value; updated CBOR Web Token

Main concerns about using a simple value for the list of redacted map keys were library support (since addressed) and RFC 8392 compatibility (text added warning about this).

```
/ cose-sign1 / 18([
  / protected   / <<{
    / key / 4 : h'496b…53ec', # public key thumbprint
    / algorithm / 1 : -7,     # ES256
  }>>,
  / unprotected / {},
  / payload     /<<{
    / iss / 1 : "https://issuer.example",
    / sub / 2 : "https://subject.example",
    / iat / 6: 1729706150,
    / redacted_map_keys / simple(59) : [
      / hash of disclosure for inspection_passed / h'db6e…76e2',
      / hash of disclosure for license_number  / h'c234…3b04'
    ],
    /inspection_dates / 502: [ 1549560720, 60(h'7ad6…092f')
  }>>,
  / signature   / h'2544f2ed...5840893b'
])
```

sd_claims format

Kept bstr encoding of individual claims since it is input to a hash and that is what COSE does.

Removed extra bstr encoding of the whole array, since there is no hash over the array and no significance to ordering/encoding choices.

```
/ CWT unprotected / {
  / sd claims / 17 : [ / these are the disclosures /
    <<[
        /salt/   h'bae611067bb823486797da1ebbb52f83',
        /claim/  501,   / inspector_license_number /
        /value/  "ABCD-123456"
    ]>>,
    <<[
        /salt/   h'8de86a012b3043ae6e4457b9e1aaab80',
        /value/  1549560720   / inspected 7-Feb-2019 /
    ]>>,
  ]
}
```

AEAD Encrypted Disclosures

Why? Some Verifiers have internal structure. Want to disclose claims to some parts of the verifier but not others.

- RATS workload appraiser vs. platform appraiser
- MLS group members vs. MLS Distribution Service (DS)

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

```
/ sd_encrypted_claims / 19 : [ / encrypted disclosures /
    <<[
        / salt /        h'bae611067bb823486797da1ebbb52f83',
        / alg /         1, / AEAD AES 128 GCM /
        / ciphertext / h'634ae7a62782fbdb978ab2d03ca6a9f0
                          0f123903f8ab59461fbd8ac3d86ef787
                          ba9b90',
        / mac /         h'8ce37110fc882bbf99c5f39e4adba636
    ]>>,
    ...
]
```


I E T F

Changes Since 121— Other

- Register `sd-cwt` and `sd-kbt` as media types and CoAP types
- Wrote walk through of the protocol
- Added Security and Privacy considerations
- Added nested disclosures example
- All examples regenerated with correct signatures / salts / hashes / dates
- NEW! Rust 🦀 implementation in Implementation Status section (merged)

Open Issues/PRs

Open PRs / Issues

- PRs
    -
- Issues
    - Support COSE key thumbprint as valid confirmation method when key is shared out-of-band #56
    - Always make value the 2nd item in a disclosure array #54 and #62
    - Verify EDN in draft matches corresponding CBOR — just work #53
    - Verify CBOR validates with CDDL — just work #52
    - Explain that holder has choice of disclosure order #50
    - Add decoy digest example — just work #11

Keep order of claims consistent #54 / #62

Make the “value” always the second item in the disclosure array. Makes parsing faster on strongly-typed languages.

Note: This is *different* from SD-JWT, which chose the order based on human readability.

```
<<[
    /salt/   h'8de86a012b3043ae6e4457b9e1aaab80',
    /value/  1549560720   / inspected 7-Feb-2019 /
]>>,
<<[
    /salt/   h'bae611067bb823486797da1ebbb52f83',
    /claim/  501,   / inspector_license_number /
    /value/  "ABCD-123456"
]>>

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

]>>
```

(array item)
value is 2nd

(map item) in -03
value is 2nd

(map item) in plan
value is 2nd

Next Steps

Next Steps

- Keep cranking on open issues
- Update implementations to use latest changes
- Interop testing of implementations