Selective Disclosure for JWTs

draft-ietf-oauth-selective-disclosure-jwt-01
“Simple” is a feature.
Overview of the Updates since IETF 114

- Working group adoption
- Updated terminology
- Introduced Combined Formats
- Extended to Digest Derivation Algorithms
- Clarified signature validation
- Clarified the rationale of encoding disclosures as JSON
- Optional blinding of claim names
- Described the processing model

+ Moved our GH repo to
https://github.com/oauth-wg/oauth-selective-disclosure-jwt
Updated Terminology
Disclosure A combination of a cleartext claim value, a cleartext claim name, a salt, and optionally a blinded claim name value that is used to calculate a digest for a certain claim.
Disclosure A combination of a cleartext claim value, a cleartext claim name, a salt, and optionally a blinded claim name value that is used to calculate a digest for a certain claim.
Updated Terminology

- Salt/Value Container (SVC) → Issuer-Issued Disclosures (II-Disclosures)
- sd_release in II-Disclosures document → sd_ii_disclosures

- SD-JWT-Release (SD-JWT-R) → Holder-Selected Disclosures (HS-Disclosures)
- sd_release in HS-Disclosures JWT → sd_hs_disclosures

- sd_hash_alg → sd_digest_derivation_alg
  (The digest derivation algorithm used by the Issuer to generate the digests over the salts and the claim values.)
II-Disclosures Object: “sd_release” → “sd_ii_disclosures”

Issuer creates & sends to holder together with SD-JWT:

```json
{  
  "sd_ii_disclosures": {  
    "sub": "["2GLC42sKQveCfGfryNRRN9w", "6c5c0a49-b589-431d-bae7-219122a9ec2c"],  
    "given_name": "["eluV5Og3gSNIII8EYnsxA_A", "John"],  
    "family_name": "["6Ij7tM-a5iVPGboS5tmvVA", "Doe"],  
    "email": "["eI8ZWm9QnKPnPqNenhDhQ", "johndoe@example.com"],  
    "phone_number": "["Qg_064zqAx412a108iroA", "+1-202-555-0101"],  
    "address": "["AJx-095VPrpTtN4QMOqROA", {"street_address": "123 Main St", "locality": "Anytown", "region": "Anystate", "country": "US"}],  
    "birthdate": "["Pc33JM2LchcU_lHggv_ufQ", "1940-01-01"]
  }
}
```
HS-Disclosures JWT: “sd_release” -> “sd_hs_disclosures”

Holder creates from SVC & sends to verifier together with SD-JWT:

```json
{
    "nonce": "XZOUco1u_gEPknxS78sWwG",
    "aud": "https://example.com/verifier",
    "sd_hs_disclosures": {
        "given_name": "["eluV5Og3gSNII8EYnsxA_A", "John"]",
        "family_name": "["6Ij7tM-a5iVPGboS5tmvVA", "Doe"]"
    }
}
```
Combined Format
Combined Format

**Combined Format for Issuance:**
- `<SD-JWT Header>`
- `<SD-JWT Payload>`
- `<SD-JWT Signature>`
- `<II-Disclosures>`

**Combined Format for Presentation:**
- `<SD-JWT Header>`
- `<SD-JWT Payload>`
- `<SD-JWT Signature>`
- `<II-Disclosures>`

Format for combining
- SD-JWT and II-Disclosures, or
- SD-JWT and HS-Disclosures

for transport to Holder or Verifier.

Transport protocol may support sending multiple such documents.

Issuer

End-User (Holder)

Verifier

- full credential
- subset of claims

JWT

JSON (b64)

JWT

JWT (HSD signature can be omitted)
Digest Derivation Algorithm
Digest Derivation Algorithm

Generalized Hash Algorithm to a Digest Derivation Algorithm, so that HMAC can also be used to generate digests.

For example, an interest to use PBKDF2-HMAC-SHA256-310000 with potentially smaller salt values has been expressed.

SHA-256 is a mandatory to implement hash algorithm.
Signature Validation
Signature validation on the HS-Disclosures JWT

Whether to require **Holder Binding** is up to the Verifier's policy, based on the set of trust requirements such as trust frameworks it belongs to.

A Verifier MUST NOT accept HS-Disclosures JWTs using "none" algorithm, when the Verifier's policy **requires** Holder Binding, i.e., a signed HS-Disclosures JWT.
Disclosures Encoding
Issuer

address
{"street_address": "Schulstr. 12",
 "locality": "Schulpforta" }
Issuer

address
{"street_address": "Schulstr. 12", "locality": "Schulpforta"}

to bytes
7b0a202020202273...

sha256
0d6dda872a393ec4ff1931 cb1b4119d71fdd70311fe8 559399204e56f50236c5

sign
✓ 0d6dda...
signed by Issuer

Verifier

address
{"locality": "Schulpforta", "street_address": "Schulstr. 12"}

to bytes
7b226c6f63616c6974...

sha256
f24ab8eaf82457e444aba dd756546f6a035455550d 699e6cb89f33d4455a302e
Solution:

Ensure that Issuer and Verifier hash the same data:

- Send the bytes that were hashed ("source string") from Issuer to Verifier,
  or
- Apply a transformation before hashing that ensures same input to hash
  function at Issuer and Verifier - Canonicalization (C18N)

→ Needs to be defined in the spec to ensure interoperability!
Issuer

```
address
{"street_address": "Schulstr. 12", 
 "locality": "Schulpforta" }
```

to bytes

```
7b0a2020202022737...
```

sha256

```
0d6dda872a393ec4ff1931
cb1b4119d71fdd70311fe8
559399204e56f50236c5
```

sign

✓ 0d6dda...
signed by Issuer

Verifier

```
address
{"street_address": "Schulstr. 12", 
 "locality": "Schulpforta" }
```

to bytes

```
7b0a2020202022737...
```

sha256

```
0d6dda872a393ec4ff1931
cb1b4119d71fdd70311fe8
559399204e56f50236c5
```

verify

✓ 0d6dda...
signature match
Example: Canonicalization

SD-JWT with C18N

Disclosure
Canonicalization

+ Clean data structure

- Adds a non-trivial dependency

- Hard to debug (errors at Issuer not transparent to Verifier)
Source String Encoding

+ Easy to implement with any JSON library
+ No new dependencies
+ JWS-like approach

- Looks strange

- Raw values in disclosure not accessible to JSON Schema, typing, etc.
Example: SD-JWT

Issuer creates & sends to holder:

```json
{
  "iss": "https://example.com/issuer",
  "cnf": { [...] },
  "iat": 1516239022,
  "exp": 1516247022,
  "sd_digest_derivation_alg": "sha-256",
  "sd_digests": {
    "sub": "2EDXXZ1JcE6aTcM70fZopFneYAS9-hY3lalaoLuWD1s",
    "given_name": "pC56LwpTgec18L11kps3koXapnw6S0i0d1ba34t-mY",
    "family_name": "EySQc316Ln3ZGJXwioELWSyy1m_60XV6rcL6LyPb7oI",
    "email": "qHv6gGaq40FmIXyKh9Z1FqJ5rOClS-dXHiPMZyl2FaU",
    "phone_number": "jhr_PsauT4xsYZS_0xBW8y_1MLUL0ovKseRvF9CE0TM",
    "address": "eQXgmowqkTORKedoqeqW0wBUy4vzKG1Vhv0jh3t1_o",
    "birthdate": "qgDxFuNpf83MkKe4GCaiLuL_XZdz04pYD71Qkbv4zos"
  }
}
```

Digests of all the claims issued to the holder by the Issuer.
II-Disclosures

Issuer creates & sends to holder alongside SD-JWT:

```
"sd_ii_disclosures": {
    "sub": "{"s": "YZSmzeu7lFHUbZ8Z1QqH9Q", "v": "6c5c0a49-b589-431d-bae7-219122a9ec2c""},
    "given_name": "{"s": "kHHp91-tAZt8m9E4Jl4XbQ", "v": "John""},
    "family_name": "{"s": "PjIqpGWl4eB4QroDhqQw0w", "v": "Doe""},
    "email": "{"s": "QRamZSB5Ky0MeJyz4EAl4E", "v": "johndoe@example.com""},
    "phone_number": "{"s": "xniP4JZtNWIH-LkDt-o-A", "v": "+1-202-555-0101""},
    "address": "{"s": "KtfsxxTm2mw0YLUCKZU8tA", "v": {"street_address": "123 Main St", "locality": "Anytown", "region": "Anystate", "country": "US"}}",
    "birthdate": "{"s": "Ozd4wBLBwqGzJhJvTmQwdQ", "v": "1940-01-01""}
}
```

String hashes to digest in the SD-JWT:

```
"address": "eQXgmowqki_UKkedoqewUwBUy4VZkwG1VnVOjhn3tL_o",
"birthdate": "qqDxFuNpf83MkKe4GCailuL_XZdzO4pYD7lQkvv4z0s"
```
Recap: JSON-encoded Disclosures

Encoding of disclosures as JSON:

- ... encodes arbitrary objects
- ... ensures same digest at issuer and verifier
- ... cleanly separates salt and value
- ... is easy to implement
- ... is safe to parse
- ... is extensible
Subset of II-Disclosures selected by Holder, plus additional data, potentially signed:

```json
{
  "nonce": "XZOUco1u_gEPknxS78sWWg",
  "aud": "https://example.com/verifier",
  "sd_hs_disclosures": {
    "given_name": "{\"s\": \"kHHp91-tAZt8m9E4Jl4XbQ\", \"v\": \"John\"}",
    "family_name": "{\"s\": \"PjIqpGWL4eB4QroDhqQw0w\", \"v\": \"Doe\"}",
    "address": "{\"s\": \"KtfsxxTm2mw0YLUCKZU8tA\", \"v\": {\"street_address\": \"123 Main St\", \"locality\": \"Anytown\", \"region\": \"Anystate\", \"country\": \"US\"}}"
  }
}
```

Document serves two purposes:

1. Transfer disclosures to Verifier
2. Extra information (nonce/aud/…), potentially signed for holder binding.

Could be separated - let's discuss.
Claim Name Blinding
Example: SD-JWT with Blinded Claim Names

Issuer creates & sends to holder:

```json
{
    "cnf": { [...] },
    "iat": 1516239022,
    "iss": "https://example.com/issuer",
    "sd_digest_derivation_alg": "sha-256",
    "sd_digests": {
        "HS4QoeE9ty-I8BZTEupSzw": "emp2qhunGPu10Gvtgor5dFwNSasDewLqNdqXCKY14Nw",
        "birthdate": "1IjWzdrXE7niXUsahdx__-8CIJsz2bcHHH_ccwTBg",
        "email": "gszmttjNfSw7_uL31KyJRvWgL1gHM6O3LFAzqxluWDQ",
        "family_name": "Xbz5qK4Fgg-bS_CdwYyd_7qiNS9w810mRn42-FTHMPo",
    }
}
```

Claim name replaced by random string
Blinded Claim Names: Disclosure

HS-Disclosures:
{
  "nonce": "XZOUco1u_gEPknxS78sWwg",
  "aud": "https://example.com/verifier",
  "sd_hs_disclosures": {
    "family_name": "{\"s\": \"ZXPEdf3K8mtRBKDAMjEcBQ\", \"v\": \"Doe\"}",
    "birthdate": "{\"s\": \"OErzfd2Gy6jw1atlCpr6A\", \"v\": \"1940-01-01\"}",
    "HS4QoeE9ty-I8BZTEupSzw":
      
      "{\"s\": \"iq6rolXFOSyWsdCeaETNg\", \"v\": \"23\", \"n\": \"secret_club_membership_no\"}"
  }
}

n: Original claim name
Processing Model
SD-JWT Processing Model

- **Verify SD-JWT**
  - Issuer signature
  - nbf/iat/...
  - structure
- **Verify HS-Disclosures JWT**
  - (Holder Binding)
  - Transaction
  - Digests matching signed digests in SD-JWT

  For disclosed, blinded claims: **Replace placeholder claim name by disclosed original claim name**

  Merge disclosed claims into non-SD claims in SD-JWT
  - SD claims take precedence

  Pass result to application for further processing.

  Application receives verified JWT body without any SD-JWT elements.
Current Status &
Next Steps
Running Code:
5 independent implementations!

- Python: [oauthstuff/draft-selective-disclosure-jwt](https://github.com/oauthstuff/draft-selective-disclosure-jwt)
- Kotlin: [IDunion/SD-JWT-Kotlin](https://github.com/IDunion/SD-JWT-Kotlin)
- Rust: [kushaldas/sd_jwt](https://github.com/kushaldas/sd_jwt)
- TypeScript: [christianpaquin/sd-jwt](https://github.com/christianpaquin/sd-jwt)
- TypeScript: [chike0905/sd-jwt-ts](https://github.com/chike0905/sd-jwt-ts) (New!)

```python
sdjwt = SDJWT(
    user_claims,
    issuer,
    ISSUER_KEY,
    HOLDER_KEY,
    claims_structure,
    blinded_claim_names,
    iat,
    exp,
)
```

Thank you for updating the implementations as specification progresses!
Next Steps

- mapping between existing credential formats and SD-JWT functionality (i.e. W3C VC-DATA-MODEL and ISO/IEC 18013-5)
- keep discussing security/privacy considerations

- Would love to do a scoped interoperability test between the libraries
Appendix
Use-Case: W3C VC-Data-Model

JWT-VC (= SD-JWT)

```
{
    "sub": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCc",
    "jti": "http://example.edu/credentials/3732",
    "iss": "https://example.com/keys/foo.jwk",
    "nbf": 1541493724,
    "iat": 1541493724,
    "exp": 1573029723,
    "cnf": {
        "jwk": {
            ...
        }
    },
    "vc": {
        "@context": ["https://www.w3.org/2018/credentials/v1"],
        "type": ["VerifiableCredential",
                  "UniversityDegreeCredential"],
        "credentialSubject": {
            "first_name": "Jane",
            "last_name": "Doe"
        }
    },
    "sd_digests": {
        "vc": {
            "credentialSubject": {
                "email": "-Rcr4fDyjwlM_itcMxoQZC21QEWyLJcbEpH114KiE",
                "phone_number": "Jv2nwOC1wPSASutYNAxFwEnaDR1piF0e7UAkU0p8F6Y",
                "address": "ZrjKs-RmEAVeAY5zSw6GPPFrMpcgctCfaJ6g7qQhbfJ4",
                "birthdate": "qXPRRPdpNaebP8jtbEp0-skF4n7v7ASTh80Lg0mkAdQ"
            }
        }
    }
}
```

JWT-VP (= HS-Disclosures JWT)

```
{
    "iss": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCc",
    "aud": "s6BhdRkt3",
    "nbf": 1560415047,
    "iat": 1560415047,
    "exp": 1573029723,
    "nonce": "660!6345FSer",
    "vp": {
        "@context": ["https://www.w3.org/2018/credentials/v1"],
        "type": ["VerifiablePresentation"],
        "verifiableCredential": ["eyJhb...npyXw"]
    },
    "sd_hs_disclosures": {
        "vc": {
            "credentialSubject": {
                "email": "{"s": "Pc33JM2LchcU_lHggv_ufQ", "v": "johndoe@example.com"}",
                "phone_number": "{"s": "lk1xK5jMY1GTPUovMNvCA", "v": "+1-202-555-0101"}",
                "address": "{"s": "5bPs1luZNa0hkaFzzzZNgw", "v": "{"street_address": "123 Main St", "locality": "Anytown", "region": "Anystate", "country": "US"}}}",
                "birthdate": "{"s": "y1sV5wdfJahVdgwPs7RQ", "v": "1940-01-01"}
            }
        }
    }
}
```

Note: specific syntax is under discussion.