SD-JWT

Selective Disclosure for JWTs

draft-ietf-oauth-selective-disclosure-jwt-01

“Simple” is a feature.

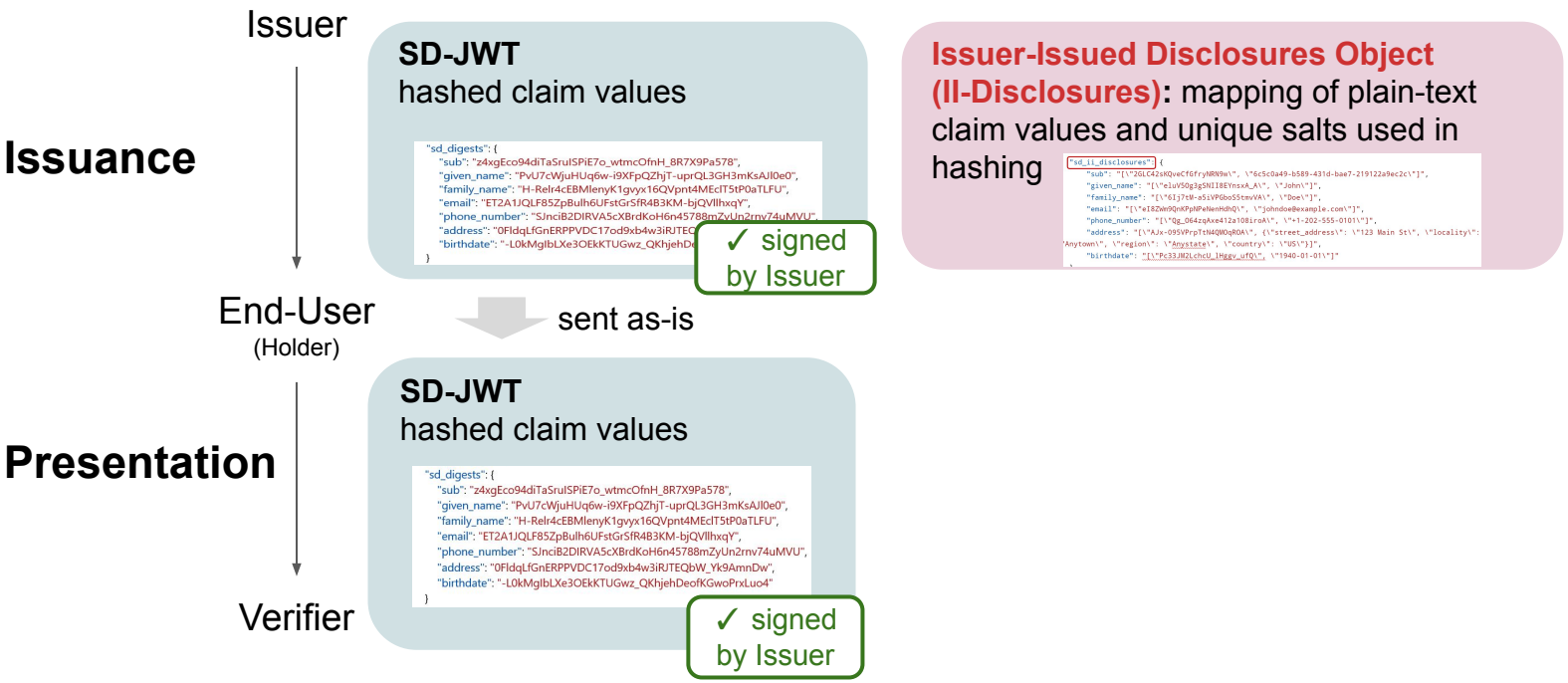
Overview of the Updates since IETF 114

- 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

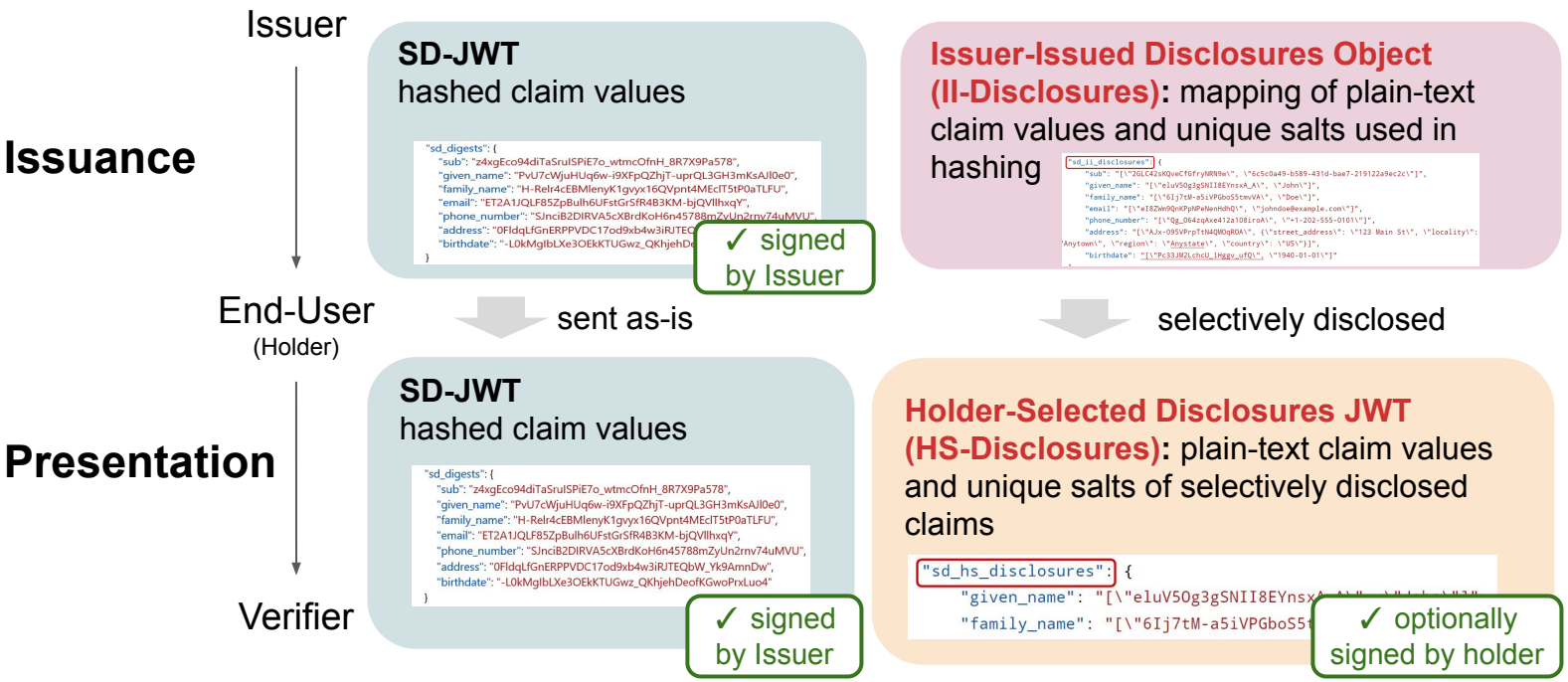
Presentation



New term

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.

Presentation



New term

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:


```
{
   "sd_ii_disclosures": {
        "sub": "[\"2GLC42sKQveCfGfryNRN9w\", \"6c5c0a49-b589-431d-bae7-219122a9ec2c\"]",
        "given_name": "[\"eluV5Og3gSNII8EYnsxA_A\", \"John\"]",
        "family_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"Doe\"]",
        "email": "[\"eI8ZWm9QnKPpNPeNenHdhQ\", \"johndoe@example.com\"]",
        "phone_number": "[\"Qg_O64zqAxe412a108iroA\", \"+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\"]"
   }
}
```

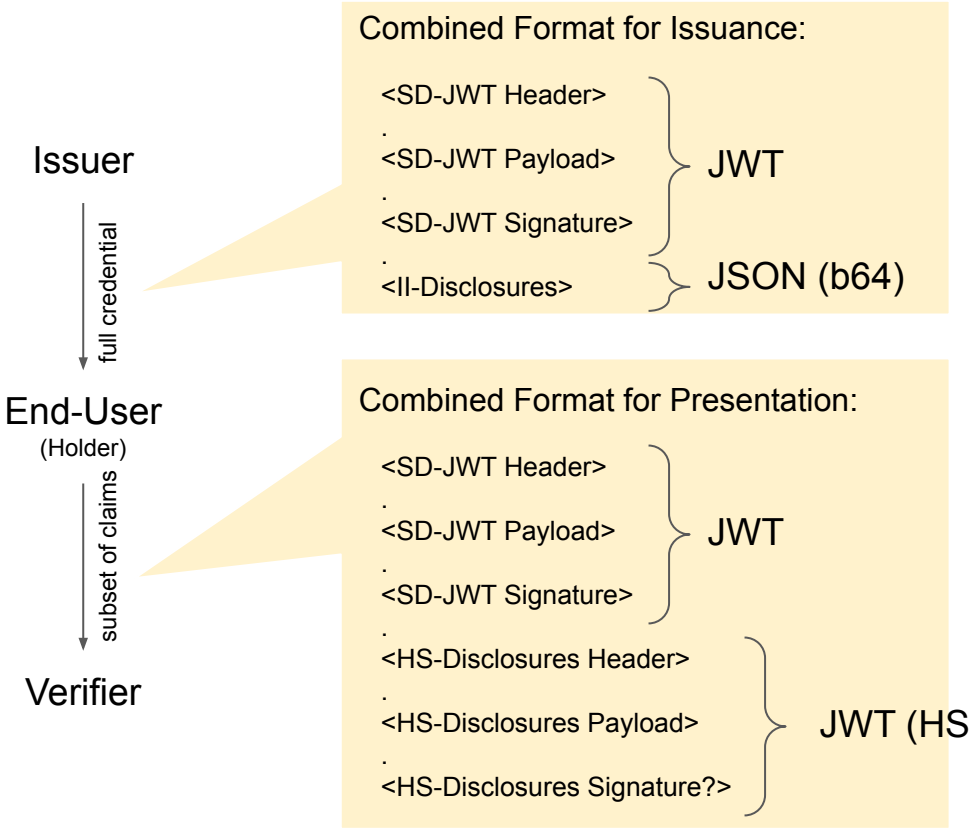
HS-Disclosures JWT: “sd_release” -> “sd_hs_disclosures”

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


```
{
   "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



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.

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.

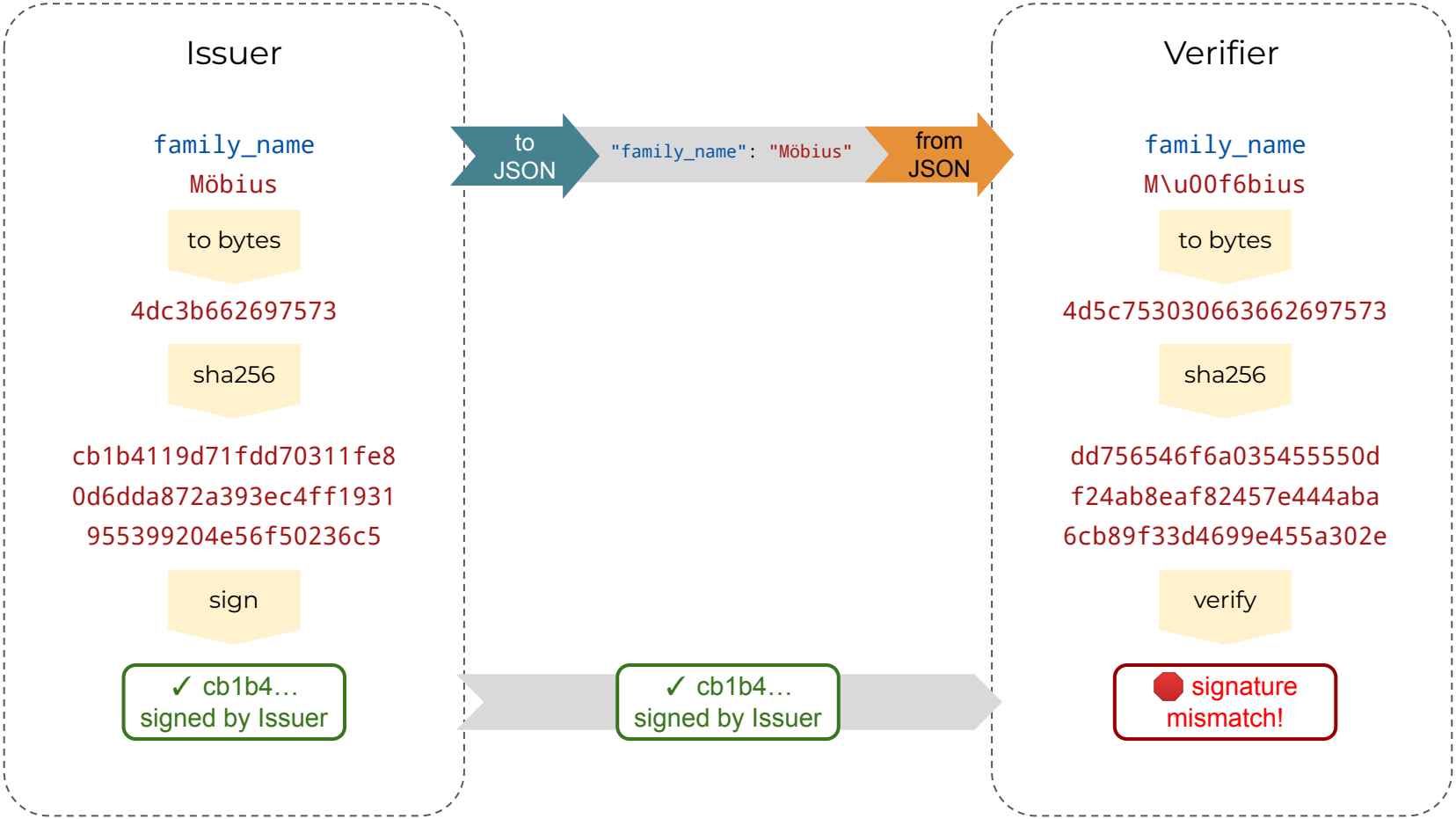
Signature Validation

Signature validation on the HS-Disclosures JWT

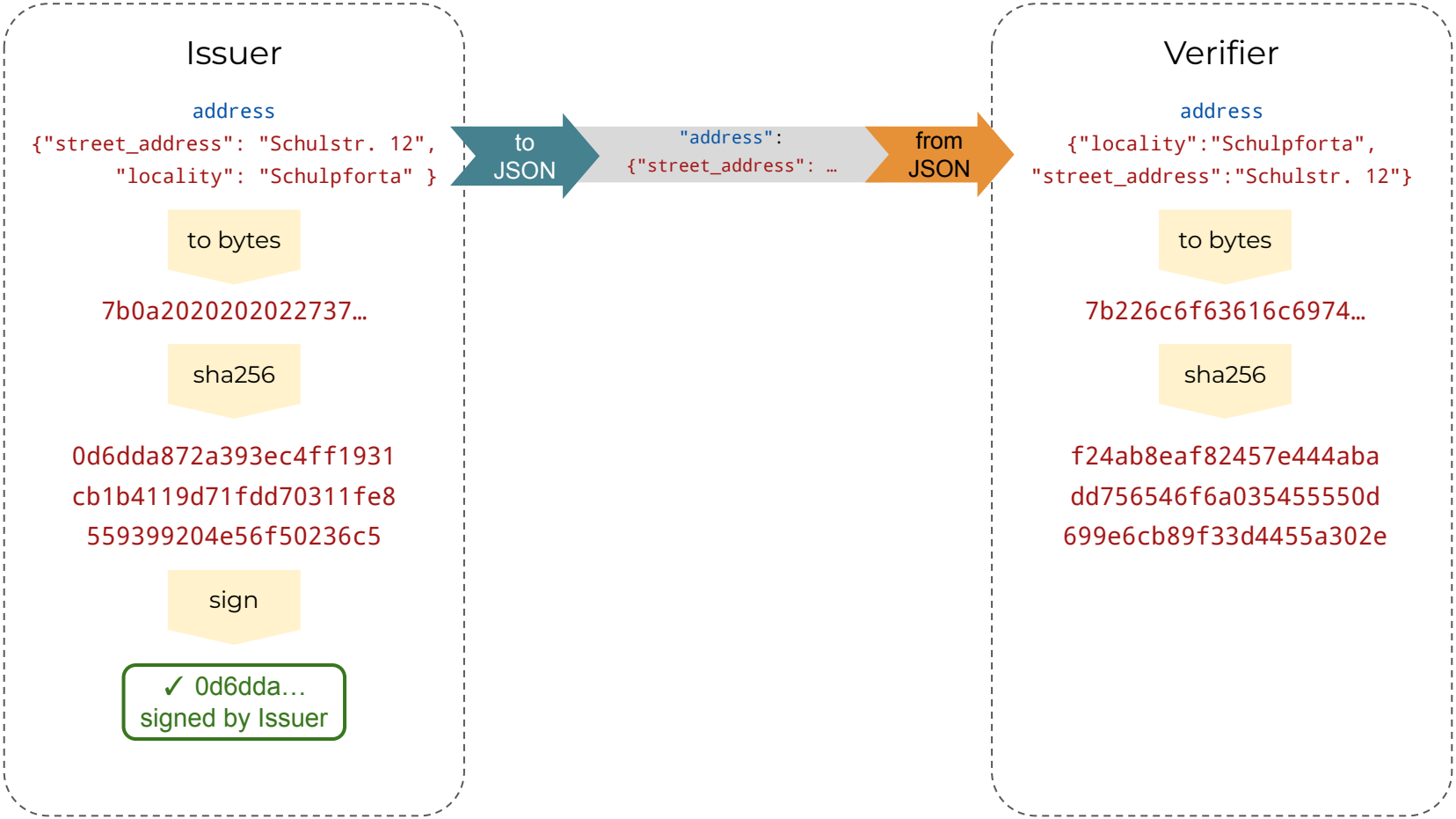
Whether to require **Holder Binding** is up to the Verifier's policy, based on the set of trust requirements such as trust frameworks it belongs to.

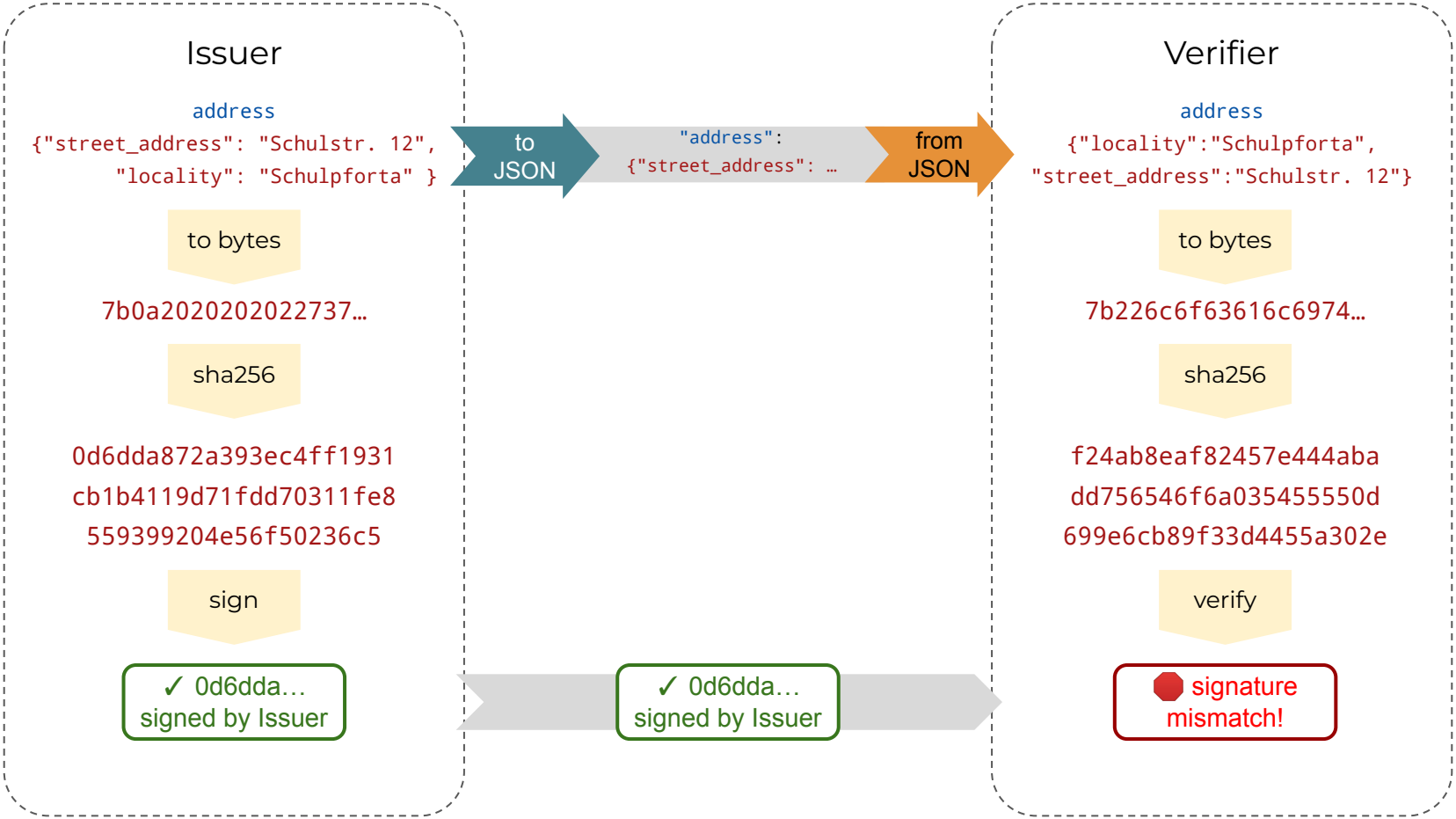
A Verifier MUST NOT accept HS-Disclosures JWTs using "none" algorithm, when the Verifier's policy **requires** Holder Binding, i.e., a signed HS-Disclosures JWT.

Disclosures Encoding


Issuer
family_name
Möbius
to bytes
4dc3b662697573
sha256
cb1b4119d71fdd70311fe8
0d6dda872a393ec4ff1931
955399204e56f50236c5
sign
✓ cb1b4…
signed by Issuer
to JSON
"family_name": "Möbius"
from JSON
✓ cb1b4…
signed by Issuer
Verifier
family_name
M\u00f6bius
to bytes
4d5c753030663662697573
sha256
dd756546f6a035455550d
f24ab8eaf82457e444aba
6cb89f33d4699e455a302e
verify
signature
mismatch!


Issuer
address
{"street_address": "Schulstr. 12",
"locality": "Schulpforta" }
to bytes
7b0a2020202022737…
sha256
0d6dda872a393ec4ff1931
cb1b4119d71fdd70311fe8
559399204e56f50236c5
sign
✓ 0d6dda…
signed by Issuer


Issuer
address
{"street_address": "Schulstr. 12",
"locality": "Schulpforta" }
to bytes
7b0a2020202022737…
sha256
0d6dda872a393ec4ff1931
cb1b4119d71fdd70311fe8
559399204e56f50236c5
sign
✓ 0d6dda…
signed by Issuer
to JSON
"address":
{"street_address": …
from JSON
Verifier
address
{"locality":"Schulpforta",
"street_address":"Schulstr. 12"}
to bytes
7b226c6f63616c6974…
sha256
f24ab8eaf82457e444aba
dd756546f6a035455550d
699e6cb89f33d4455a302e

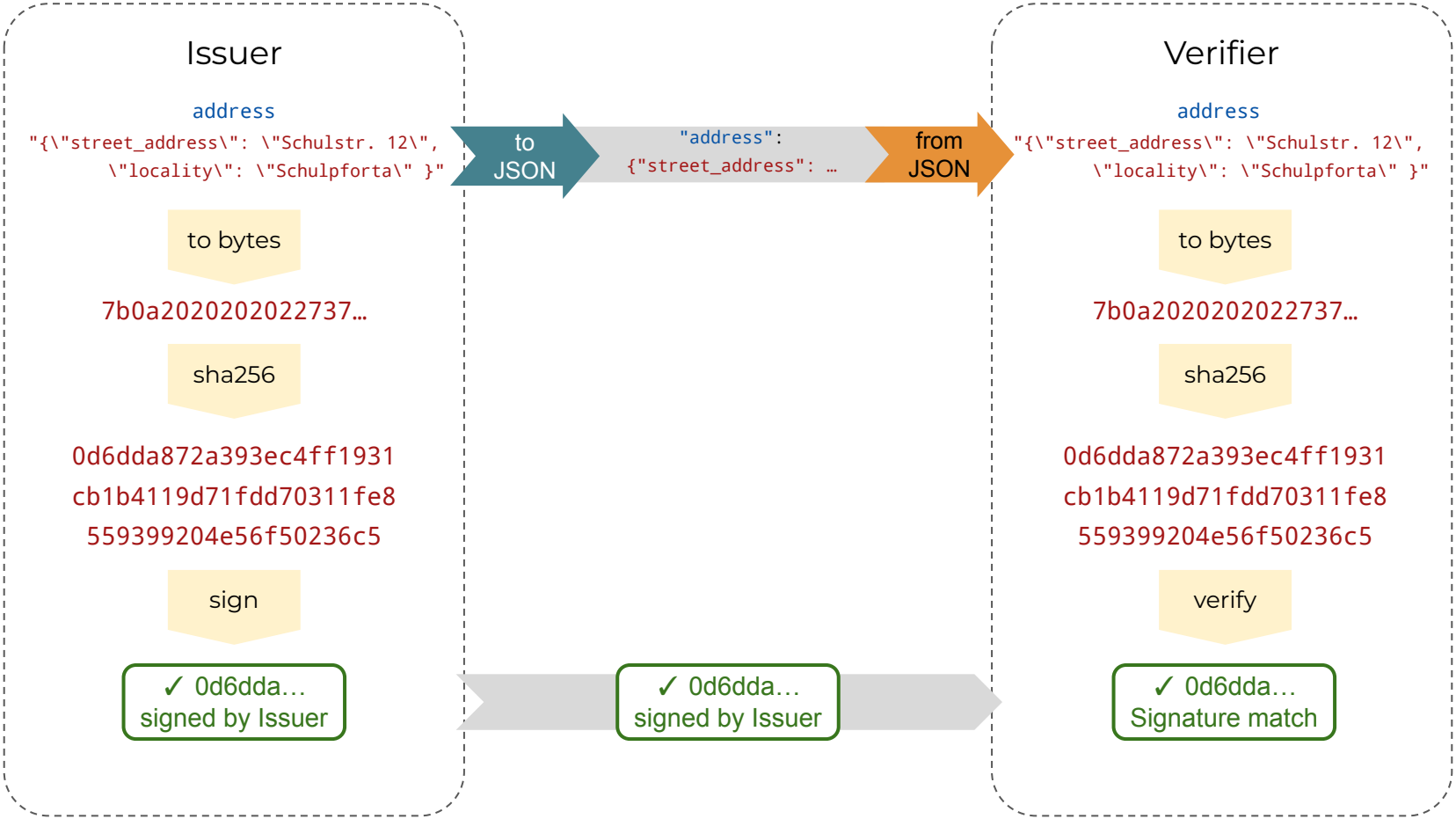

Issuer
address
{"street_address": "Schulstr. 12",
"locality": "Schulpforta" }
to bytes
7b0a2020202022737…
sha256
0d6dda872a393ec4ff1931
cb1b4119d71fdd70311fe8
559399204e56f50236c5
sign
✓ 0d6dda…
signed by Issuer
to JSON
"address":
{"street_address": …
from JSON
✓ 0d6dda…
signed by Issuer
Verifier
address
{"locality":"Schulpforta",
"street_address":"Schulstr. 12"}
to bytes
7b226c6f63616c6974…
sha256
f24ab8eaf82457e444aba
dd756546f6a035455550d
699e6cb89f33d4455a302e
verify
signature
mismatch!

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 JSON
"address":
{"street_address": …
from JSON
Verifier
address
"{\"street_address\": \"Schulstr. 12\",
\"locality\": \"Schulpforta\" }"
to bytes
7b0a2020202022737…
sha256
0d6dda872a393ec4ff1931
cb1b4119d71fdd70311fe8
559399204e56f50236c5
sign
✓ 0d6dda…
signed by Issuer
✓ 0d6dda…
signed by Issuer
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:


```
{
 "iss": "https://example.com/issuer",
 "cnf": { [...] },
 "iat": 1516239022,
 "exp": 1516247022,
 "sd_digest_derivation_alg": "sha-256",
 "sd_digests": {
   "sub": "2EDXXZ1JcE6aTcM70fZopFneYAS9-hY3lalaoLuWD1s",
   "given_name": "pC56LWpTgec18Ll1kps3koXapnw6SOiI0d1ba34t-mY",
   "family_name": "EySQc316Ln3ZGJXwioELWSyylm_6OXV6rcL6LyPb7oI",
   "email": "qHv6gGaq4oFmIXyKh9ZlFjQ5rOClS-dXHiPMZyl2FaU",
   "phone_number": "jhr_PsauT4xsYZS_OxBW8y_1MLULOovKseRvF9CE0TM",
   "address": "eQXgmowqkT_ORkedoqeW0wBUy4vzkWG1VhvOjh3tl_o",
   "birthdate": "qgDxFuNpf83MkKe4GCaiLuL_XZdzO4pYD7lQKbv4zos"
 }
}
```


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\": \"QRamZSB5KyOMeJyz4EAleA\", \"v\": \"johndoe@example.com\"}",
   "phone_number": "{\"s\": \"xniP4JZtNWIH-Lk_Dt-o-A\", \"v\": \"+1-202-555-0101\"}",
   "address": "{\"s\": \"KtfsxxTm2mwOYLUcKZU8tA\", \"v\": {\"street_address\": \"123 Main St\", \"locality\":
     \"Anytown\", \"region\": \"Anystate\", \"country\": \"US\"}}",
   "birthdate": "{\"s\": \"Ozd4wBLBwqGzJhJvTmQwdQ\", \"v\": \"1940-01-01\"}"
 }
}
```


s: Salt v: Value

String hashes to digest in the SD-JWT:


```
"address": "eQXgmowqkI_ORkedoqeWOWBUy4VzkWGIVhVOJh3tI_o",
"birthdate": "qgDxFuNpf83MkKe4GCaiLuL_XZdzO4pYD7lQKbv4zos"
```

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

HS-Disclosures

Subset of II-Disclosures selected by Holder, plus additional data, potentially signed:


```
{
 "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:

```
{
 "cnf": { [...] },
 "iat": 1516239022,
 "iss": "https://example.com/issuer",
 "sd_digest_derivation_alg": "sha-256",
 "sd_digests": {
   "HS4QoeE9ty-I8BZTEupSzw": "emp2qhunGPulOGvtgor5dFwNSasDewLqNdqXCkYl4Nw",
   "birthdate": "1IjWWzdrXEs7iXUbsahdx_-8CIJsz2bcHHH_ccwgTBg",
   "email": "gszmttjNfSw7_uL31KyJRvWgL1gHM6O3LFAzqxluWDQ",
   "family_name": "Xbz5qK4Fqg-bS_CdwQYd_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\": \"OErzfd2Gy6jw1atlcCpr6A\", \"v\": \"1940-01-01\"}",
   "HS4QoeE9ty-I8BZTEupSzw":
             "{\"s\": \"iq6rolXFOSyWSsdCeaETNg\", \"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!

```
### Produce SD-JWT and SVC

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

- **Python: oauthstuff/draft-selective-disclosure-jwt**
- Kotlin: IDunion/SD-JWT-Kotlin
- Rust: kushaldas/sd_jwt
- TypeScript: christianpaquin/sd-jwt
- TypeScript: chike0905/sd-jwt-ts **(New!)**

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_itcMxoQZCE1QAEwyLJcibEpH114KiE",
        "phone_number": "Jv2nwOC1wP5ASutYNAxrWEnaDRIpiF0eTUAkUOp8F6Y",
        "address": "ZrjKs-RmEAVeAYSzSw6GPFrMpcgctCfaJ6t9qQhbfJ4",
        "birthdate": "qXPRRPdpNaebP8jtbEpO-skF4n7v7ASTh8oLg0mkAdQ"
      }
    }
  }
}
```


JWT-VP (= HS-Disclosures JWT)


```
{
  "iss": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCc",
  "aud": "s6BhdRkqt3",
  "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\": \"lklxF5jMYlGTPUovMNIvCA\", \"v\":
          \"+1-202-555-0101\"}",
        "address": "{\"s\": \"5bPs1IquZNa0hkaFzzzZNw\", \"v\":
          {\"street_address\": \"123 Main St\", \"locality\":
          \"Anytown\", \"region\": \"Anystate\", \"country\":
          \"US\"}}",
        "birthdate": "{\"s\": \"y1sVU5wdfJahVdgwPgS7RQ\", \"v\":
          \"1940-01-01\"}"
      }
    }
  }
}
```


Note: specific syntax is under discussion.