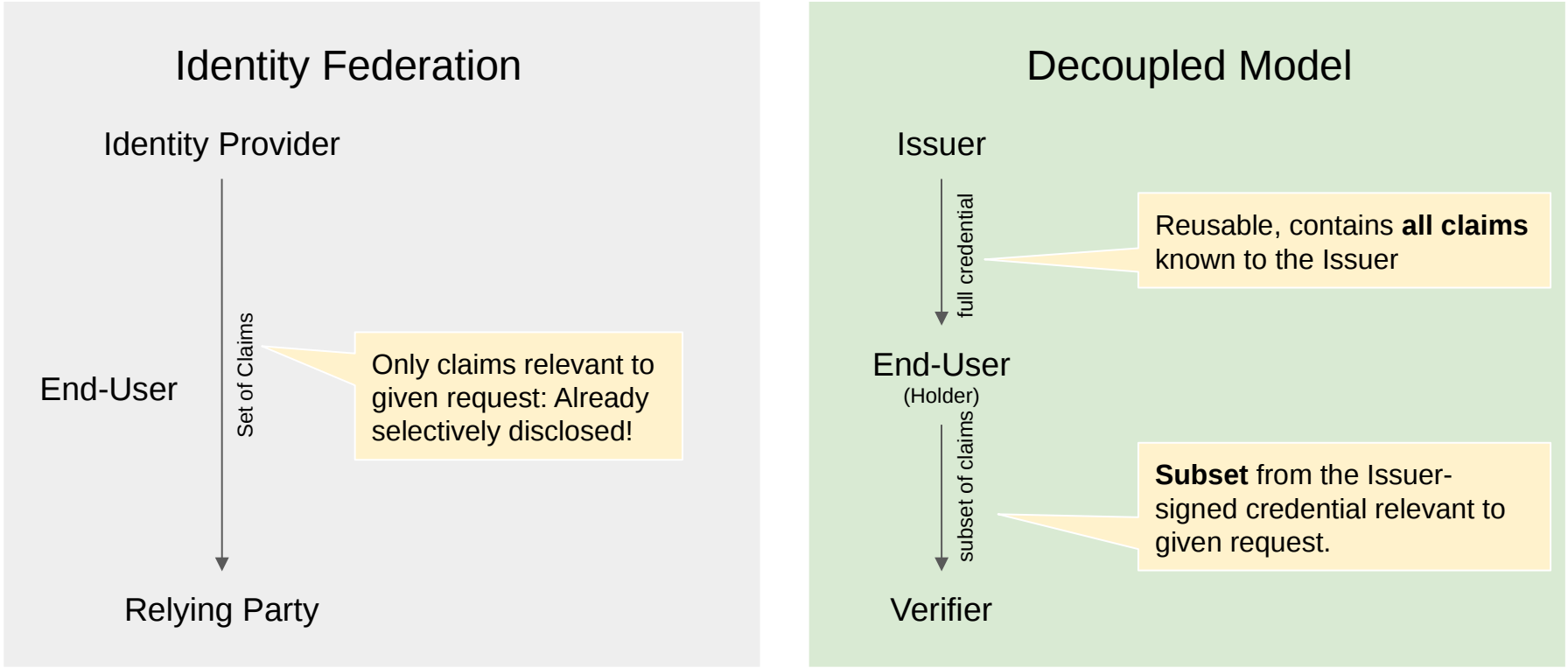
SD-JWT

Selective Disclosure for JWTs

draft-fett-selective-disclosure-jwt

oauthstuff/draft-selective-disclosure-jwt

Decoupling Issuance & Presentation



RP chooses which claims are required to be released to receive service.

Selective Disclosure

Issuer issued a whole set of claims:

```
{
   "iss": "https://server.example.com",
   "sub": "some-user-identifier",
   "aud": "s6BhdRkqt3",
   "given_name": "John",
   "family_name": "Doe",
   "email": "johndoe@example.com",
   "phone_number": "+1-202-555-0101",
   "address": {
       "street_address": "123 Main St",
       "locality": "Anytown",
       "region": "Anystate",
       "country": "US"
   },
   "birthdate": "1940-01-01"
}
```

But Verifier needs only a subset in a given request:

```
{
   "iss": "https://server.example.com",
   "sub": "some-user-identifier",
   "aud": "s6BhdRkqt3",
   "given_name": "John",
   "family_name": "Doe",
   "email": "johnd
   "phone_number": "+1-202-555-0101",
   "ad
       "country": "US"
   },
   "birthdate": "1940-01-01"
}
```

SD-JWT

Selective Disclosure for JWTs

“Simple” is a feature.

Why another Spec?

	Existing SD schemes	**SD-JWT**
Complexity	(Very) Complex	As simple as possible
Algorithms	Advanced cryptography	Standard cryptography with salted hashes
Format	Binary formats	JSON & JWT
Security	Hard to audit	Easy to understand & verify
Availability	Lack of implementations	Widely-available JWT libraries can be leveraged; Already 4 independent implementations
Use Cases	often limited to the identity use cases	Universal (beyond identity use cases)

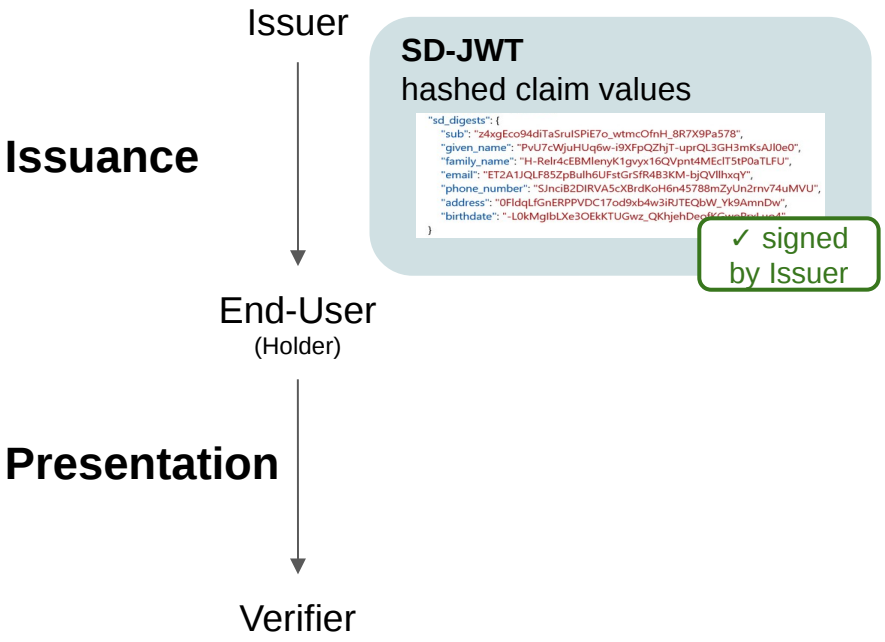
Salted Hash Approach

Idea:

- Issuer hashes each claim value together with a random salt
  - `"John" → hash(salt, "John") → "PvU7cWjuHUq6w-i9XFpQZhjT-uprQL3GH3mKsAJl0e0"`
- Issuer-signed credential only contains digests
  - `"given_name": "PvU7cWjuHUq6w-i9XFpQZhjT-uprQL3GH3mKsAJl0e0"`
- Holder selectively releases salt + plain-text value
  - `"given_name" = hash("eluV5Og3gSNII8EYnsxA_A", "John")`
- Verifier checks by calculating hashes & issuer’s signature

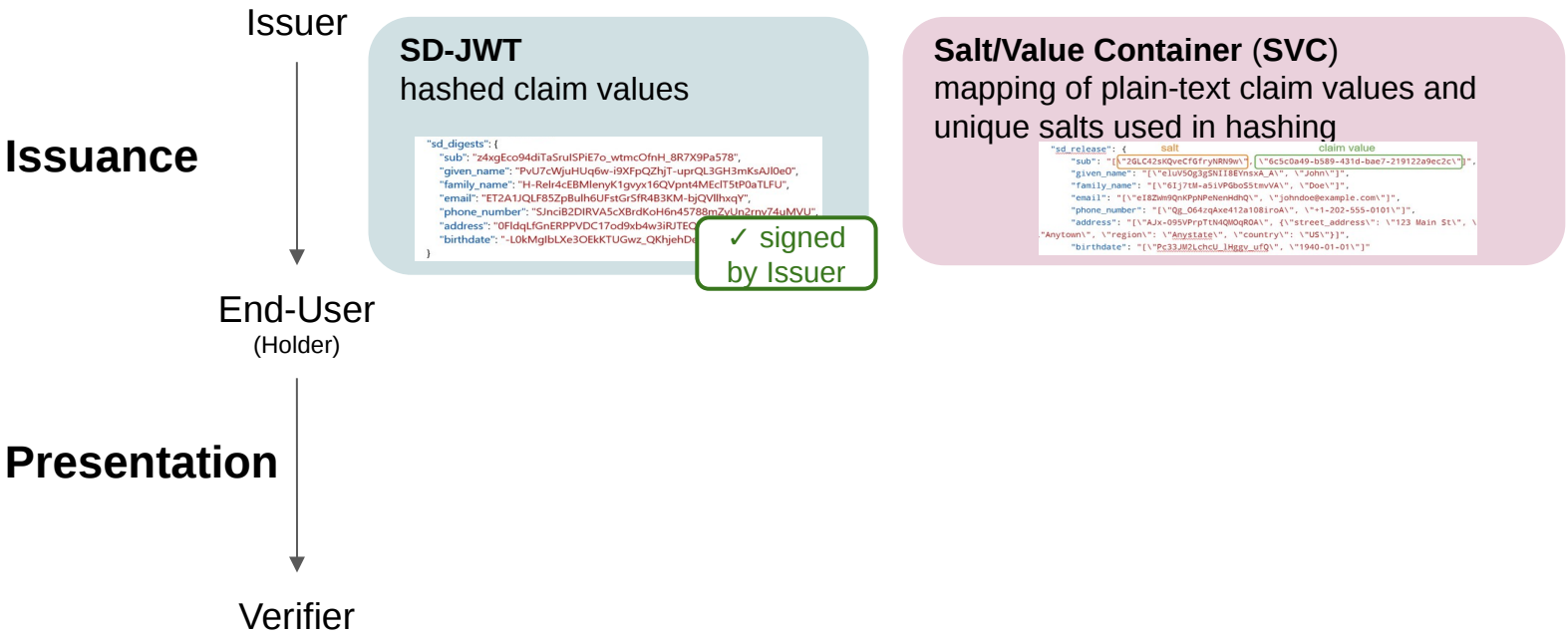
Overview: Salted Hash Approach

Issuance (1/2)



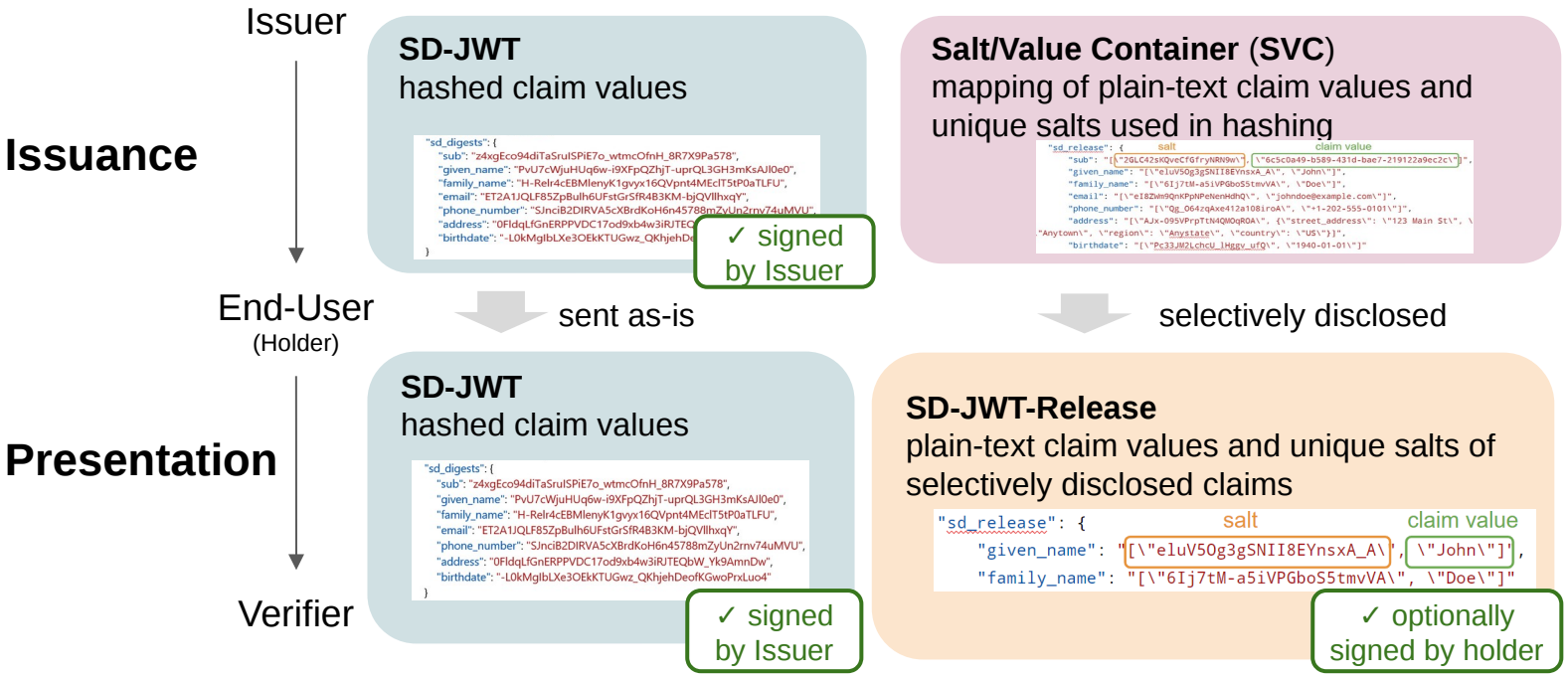
- Issuer hashes each claim value together with a random salt
  `-> "John" → hash(salt, "John") → "PvU7cWjuHUq6w-i9XFpQZhjT-uprQL3GH3mKsAJl0e0"`
- Issuer-signed credential only contains digests
  `-> "given_name": "PvU7cWjuHUq6w-i9XFpQZhjT-uprQL3GH3mKsAJl0e0"`

Issuance (2/2)



- Issuer sends a mapping of plain-text claim values and unique salts
  `-> given_name": "[\"eluV5Og3gSNII8EYnsxA_A\", \"John\"]`

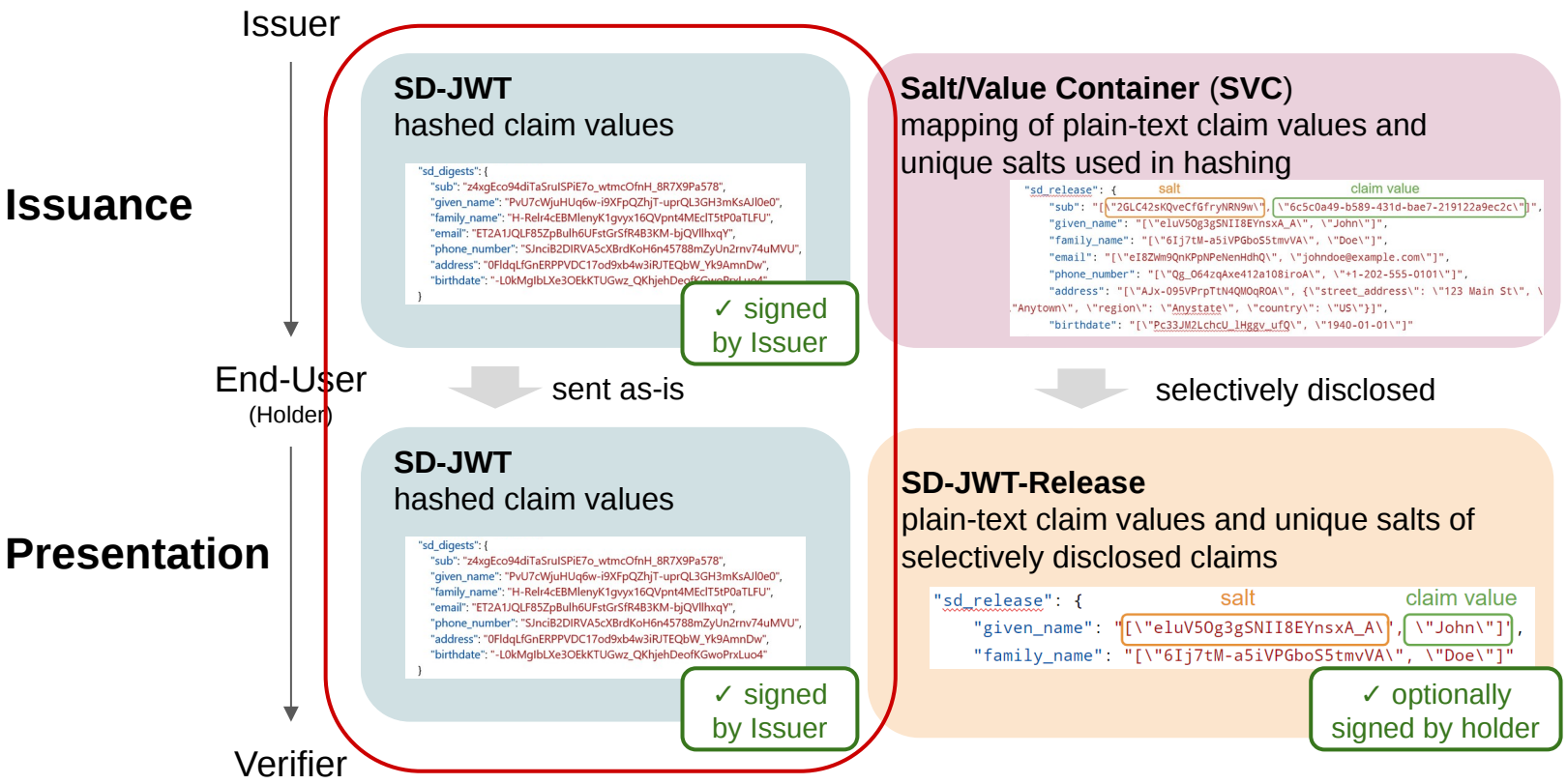
Presentation



- Holder selectively releases salt + plain-text value
  `-> "given_name" = hash("eluV5Og3gSNII8EYnsxA_A", "John")`
- Verifier checks by calculating hashes & issuer's signature

Deep-dive

Deep-dive on SD-JWT



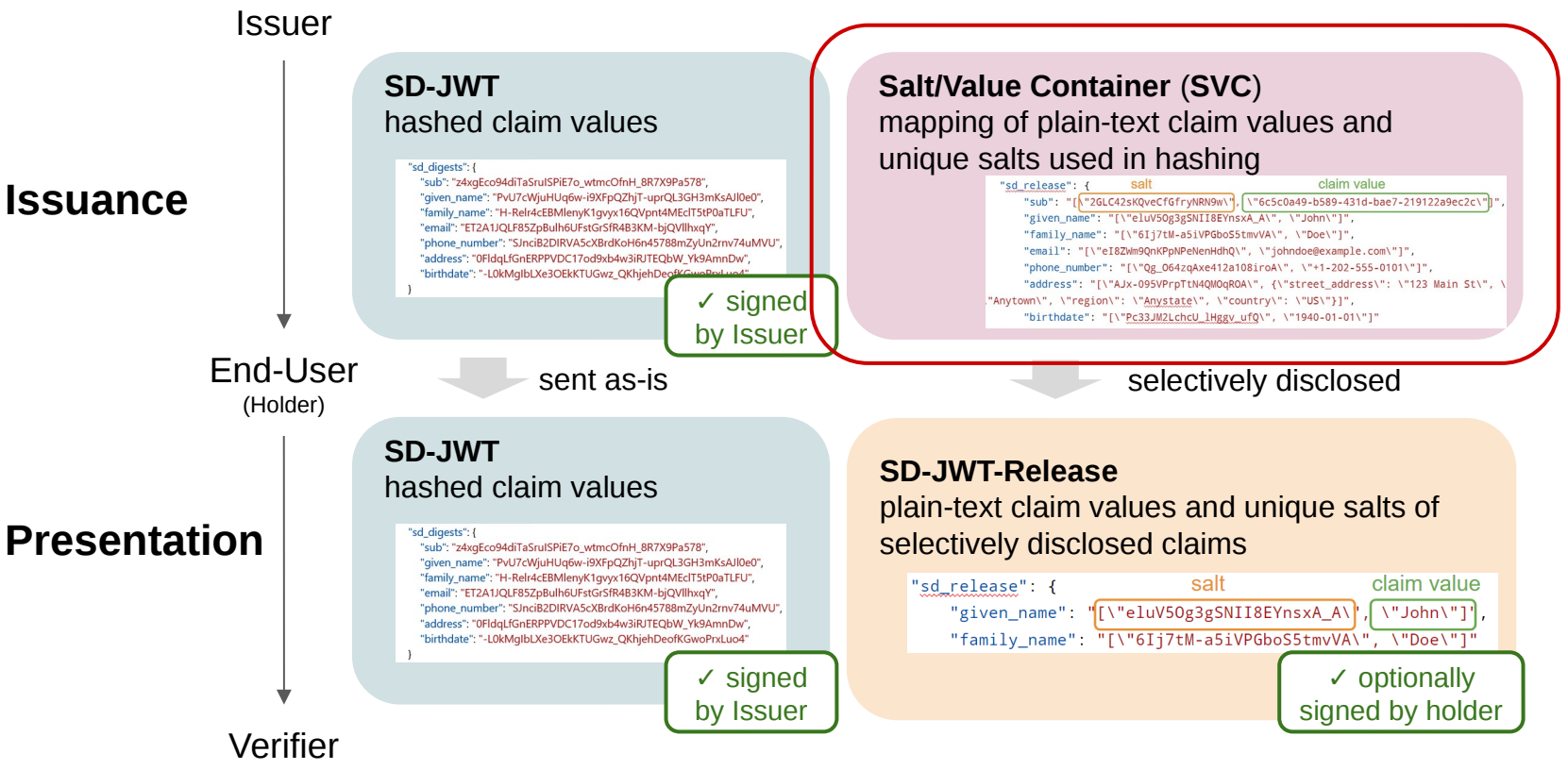
Example: SD-JWT

Issuer creates & sends to holder:

```
{
   "iss": "https://example.com/issuer",
   "sub_jwk": {  # optional: public key of holder
       "kty": "RSA",
       "n": "pm4bOHBg-oYhAyP(...)7ihcw",
       "e": "AQAB"
   },
   "iat": 1516239022,
   "exp": 1516247022,
   "sd_hash_alg": "sha-256",
   "sd_digests": {
       "sub": "z4xgEco94diTaSruISPiE7o_wtmcOfnH_8R7X9Pa578",
       "given_name": "PvU7cWjuHUq6w-i9XFpQZhjT-uprQL3GH3mKsAJl0e0",
       "family_name": "H-Relr4cEBMlenyK1gvyx16QVpnt4MEclT5tP0aTLFU",
       "email": "ET2A1JQLF85ZpBulh6UFstGrSfR4B3KM-bjQVllhxqY",
       "phone_number": "SJnciB2DIRVA5cXBrdKoH6n45788mZyUn2rnv74uMVU",
       "address": "0FldqLfGnERPPVDC17od9xb4w3iRJTEQbW_Yk9AmnDw",
       "birthdate": "-L0kMgIbLXe3OEkKTUGwz_QKhjehDeofKGwoPrxLuo4"
   }
}
```

Digests of all the claims issued to the holder by the Issuer

Deep-dive on SVC



Example: Salt/Value Container

JSON strings?!

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

```
{
   "sd_release": {
       "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\"]"
   }
}
```

salt

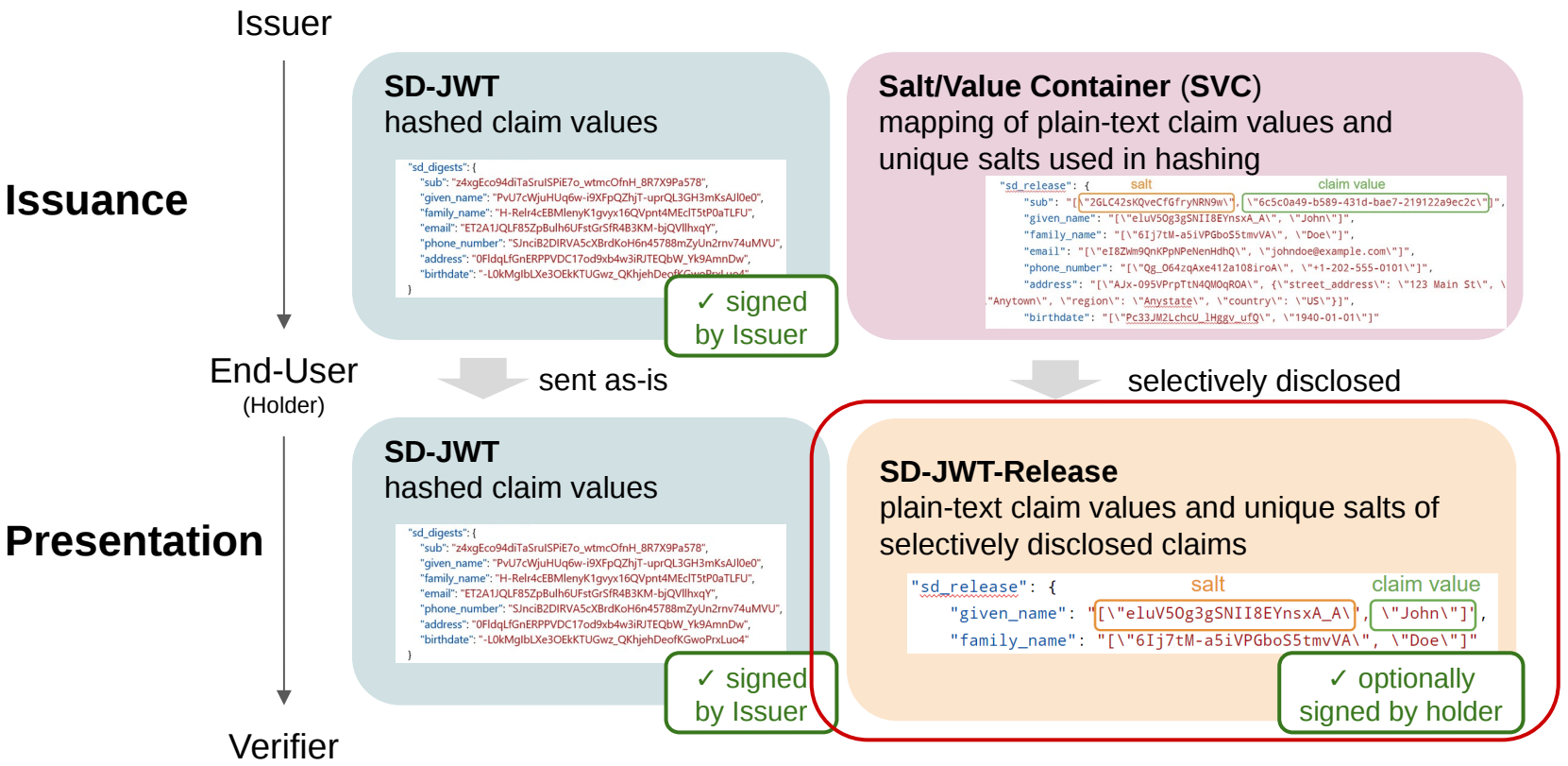
claim value

Mapping of salts and claim values of all the claims issued by the Issuer

JSON string literals…

- …ensure all parties hash the same values
- …obsolete canonicalization even for complex claim values
- …with extremely simple implementation

Deep-dive on SD-JWT-Release



Example: SD-JWT-Release

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

```
{
   "nonce": "XZOUco1u_gEPknxS78sWWg",
   "aud": "https://example.com/verifier",
   "sd_release": {
       "given_name": "[\"eluV5Og3gSNII8EYnsxA_A\", \"John\"]",
       "family_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"Doe\"]"
   }
}
```

salt

claim value

Salts and claim values of the claims that the Holder is releasing to the Verifier

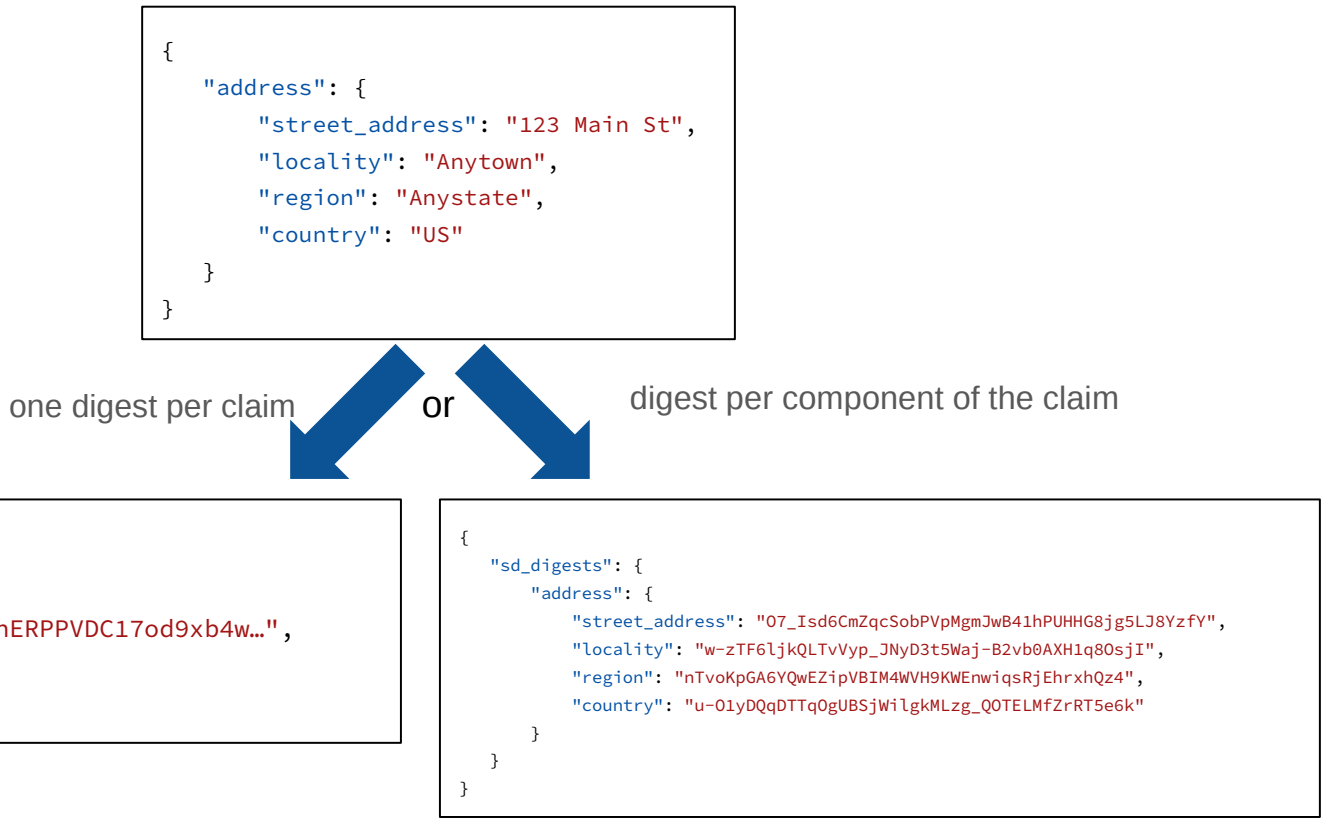
Verifier checks:

- SD-JWT is valid (signature, etc.)
- SD-JWT-Release is valid (if signed: signature, nonce, etc.)
- digests in SD-JWT match hashes of released claims

Then: Extract plain-text values from released claims

Foot-gun: Verifier MUST calculate and check digests before using claims!

Granularity of hashing in SD-JWT



Works with Complex Objects!

Example of OpenID for Identity Assurance

For full example, see the spec


```
{
   "sd_digests": {
       "verified_claims": {
           "verification": {
               "trust_framework": "w1mP4oPc_J9thBex0TaQi1vgxFmruQJxZYLFnkNFMaI",
               "time": "Pu3i0CWrPVLJW-LT30yF1bFBPP15B6-uKk3PnGDflv8",
               "verification_process": "8HqIXRmczsdYOZzGcLqI5-l9xN5QbK2XDtXmdfH7z-4",
               "evidence": [{
                   "type": "TnLuqGGQm6jfeOoa5uX1diKANUPuh-zHrpBFdX9MR-g",
                   "method": "SagmakoSu-X-XUPIC3EgdrEEwIWxRWXX4-i68X9TyEo",
                   "time": "ld2c5oYDRtQcfU6PzogPkx_95WYqhqIJNVRMnfcsicY",
                   "document": {
                       "type": "ufWjDaAa54MnHeji2ZUUHDdnpZ9zx6CUG6uR28VMtsQ",
                       "issuer": {
                           "name": "a4GMucU7Zb060r0Svd7huY6Qho1bIf3v1U5BvPR8q6Y",
                           "country": "135k9M0m2SCnYRuOfHuYScYVS2q3eeY7IItgyRsaBT8"
                       },
                       "number": "cUvOxLUp8RV7TTVliEiu-TQIel-LsE8E-XfUgfqk5gk",
                       "date_of_issuance": "NIs8olJnJOv4J1qIEBKuTs2sEFs4fgGJhNqM6xdQt7E",
                       "date_of_expiry": "HTR37vLtANT6MWk-9dBqekFpCvaTG7zNf1ze56rnV64"
                   }
               }]
           },
           "claims": {
               "given_name": "NB9XH_yJKqKOhXDmXkZKpMCkRbOmOTd8bqJFYDJYQnQ",
               "family_name": "hAUbJ66ZYL9VJLbjsDpmSs2e9Ff_Ohim_WR4bwZyvoQ",
               "birthdate": "6XOR4k56BgWk5tnNismbmEHvoGX7RRfy6Z8HENl96cU",
               "place_of_birth": {
                   "country": "CLTlhuy13WWc3_ISon1kEypFwvCmfhLSpGUMCyAUg68",
                   "locality": "AQoX8ixGpz-ipweEGlC-2umqwyQdhjIeiUB_TKWcE2E"
               },
               "nationalities": "nfoc__QKlMUHodmxwlY-Kp-6ewgX3CdK7Ia0RJHIXVo",
               "address": "ngnO4uQeOktM7YdFD8x82doS7WJnlZnq-rQE_RfuBSI"
           }
       },
       "birth_middle_name": "FeFSwd9drypEPtWVgIZ42N9j_yostt1Ds5PBpxT3Rng",
       "salutation": "57CMhvASQMNuzuQ0a_B1_VX5XdH73TcuPxyWGiorj5g",
       "msisdn": "leKbB0ro6q3jrVraCqt443uaGZVZisD3iGrKuKE2mqM"
   }
}
```

Running Code:
4 independent implementations!

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

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

- **Python: https://github.com/oauthstuff/draft-selective-disclosure-jwt**
- Kotlin: https://github.com/IDunion/SD-JWT-Kotlin
- Rust: https://github.com/kushaldas/sd_jwt
- TypeScript: https://github.com/christianpaquin/sd-jwt

- Reference implementation in python: ~500 LoC.
- Evolves with the spec.
- Examples in the spec generated from the code.

Use-Case: W3C VC-Data-Model

JWT-VC (= SD-JWT)


```
{
   "sub": "did:example:ebfeb1f712ebc6f1c276e12ec21",
   "jti": "http://example.edu/credentials/3732",
   "iss": "https://example.com/keys/foo.jwk",
   "nbf": 1541493724,
   "iat": 1541493724,
   "exp": 1573029723,
   "vc": {
       "@context": [
           "https://www.w3.org/2018/credentials/v1",
           "https://www.w3.org/2018/credentials/examples/v1"
       ],
       "type": [
           "VerifiableCredential",
           "UniversityDegreeCredential"
       ],
       “credentialSubject”: {
           "sd_digests": {
               "given_name": "fUMdn8aoyKTHrvZd6AuLmPraGhPJ0zF5r_JhxCVZs",
               "family_name": "9h5vgv6TpFV6GmnPtugiMLl5tHeeb5X_2cKHjN7cw",
               "birthdate": "fvLCnDm3r4VSYcBF3pIlXP4ulEoHuHOfG_YmFZEuxpQ"
           }
       }
   }
}
```


JWT-VP (= SD-JWT-Release)


```
{
   "iss": "did:example:ebfeb1f712ebc6f1c276e12ec21",
   "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_release": {
       "given_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"John\"]",
       "family_name": "[\"eI8ZWm9QnKPpNPeNenHdhQ\", \"Doe\"]"
   }
}
```


Note: specific syntax is under discussion.

Current Status &
Next Steps

Draft Status

- Functional specification complete and somewhat stable
- New: Approach to blinding claim names
- Some missing parts in security and privacy considerations
- Already considered a serious alternative to many existing formats
- Adopted in EU-projects & by Microsoft

OAuth WG?

- JWT was developed in OAuth WG
- Expected usage in OAuth and OAuth-based technologies
- SD-JWT is universal: Selectively disclosable access tokens?

Call for adoption?

Additional Slides

Blinding Claim Names

Hashing not only claim values but also claim names.

Credential

```
{
   "iss": "https://server.example.com",
   "sub": "some-user-identifier",
   "aud": "s6BhdRkqt3",
   "given_name": "John",
   "family_name": "Doe",
   "email": "johndoe@example.com",
   "phone_number": "+1-202-555-0101",
   "secret_club_membership_no": "42-23-23"
}
```

✓ signed by Issuer

```
{
   "iss": "https://server.example.com",
   "sub": "some-user-identifier",
   "aud": "s6BhdRkqt3",
   "given_name": "John",
   "family_name": "Doe",
   "email": "johndoe@example.com",
   "phone_number": "+1-202-555-0101",
   [redacted]: [redacted]
}
```

✓ signed by Issuer