IETF OAuth WG Draft

https://datatracker.ietf.org/doc/draft-fett-oauth-selective-disclosure-jwt/

Daniel Fett
yes.com → Authlete

Kristina Yasuda

Brian Campbell
Ping
‘Simple’ is a feature.
# Design Principles

<table>
<thead>
<tr>
<th><strong>SD-JWT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complexity</strong></td>
</tr>
<tr>
<td><strong>Algorithms</strong></td>
</tr>
<tr>
<td><strong>Format</strong></td>
</tr>
</tbody>
</table>
| **Security** | Security-by-design  
Easy to understand & verify  
Hardware binding possible  
Cryptographic agility |
| **Availability** | Widely-available JWT libraries can be leveraged  
Already five independent implementations |
| **Use Cases** | Universal (beyond identity use cases) |
SD-JWT in 5 Simple Steps

Step 1: Prepare User Data

{  
  "iss": "https://example.com",
  "type": "IdentityCredential",
  "cnf": "{"jwk": {"kty": "RSA","n": "Ovx....Kgw","e": "AQAB" } },
  "credentialSubject": {
    "given_name": "Max",
    "family_name": "Mustermann",
    "email": "mustermann@example.com",
    "address": {
      "street_address": "Musterstr. 23",
      "locality": "Berlin",
      "country": "DE"
    }
  }
}
SD-JWT in 5 Simple Steps

Step 2: Create Disclosures

```json
{
    "iss": "https://example.com",
    "type": "IdentityCredential",
    "cnf": {
        "jwk": {
            "kty": "RSA",
            "n": "0vx....Kgw",
            "e": "AQAB"
        }
    },
    "credentialSubject": {
        "given_name": "Max",
        "family_name": "Mustermann",
        "email": "mustermann@example.com",
        "address": {
            "street_address": "Musterstr. 23",
            "locality": "Berlin",
            "country": "DE"
        }
    }
}
```
SD-JWT in 5 Simple Steps

Step 3: Hash Disclosures & Replace Original Claims

```json
{
    "iss": "https://example.com",
    "type": "IdentityCredential",
    "cnf": {"jwk": {"kty": "RSA","n": "0vx....Kgw","e": "AQAB" } },
    "credentialSubject": {
        "_sd": [ "EW1o0egqa5mGcbytT5S-kAubcEjYEUwRkX1u2vCS120",
                 "FE-X-ITHt41I8_cn0SS-hvLoLneX_RG1Jo_8o2xRNhfdk",
                 "igg7H5fn2eBEMIEkE5Ckbm23QwDJ1TyORKrip08dYIc" ],
        "address": { "_sd": [ "gqB5kmAwyry88aHjaAeO-USX6JOMaojukKsheo3800c",
                             "w8InvxsPXdKoowuVpyBMg11b9_R2b6Xpa3OYOIjgQro",
                             "w0n1Ytcjr872fP3Wa75Oz17c-6_MOVDIUNtwLKKxZw0" ]
    }
}
```
SD-JWT in 5 Simple Steps
Step 4: Sign SD-JWT & Encode for Transport

```json
{"iss": "https://example.com",
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZNhOMwjgTmdYUEifQ,eyJpZCI6Iml>c7NiXCI6Ili1CJCI6Iiwdng3YWdvZ
YJr1FTd5uL4t1Y3NGQ3VvLWtFZ1UYdhcEp6S25xREtnyISICJCI6IjogIKFRQu1if
X0s1CJ0eXBlIjogIklkZw50aXSRQ3J1ZGVudG1hbm5ICjcmVkZw50aWFsU3ViamVjd
Ci6H1Sx3K1jogWjFvVFMGvncWE1bUdYn1oDVTLOtVwdWjRp2ZV3UmtYbHuydk
kM1bD1wiwogIkJFeCFjVEhONDFJQj9jbyBTUy1odm9MbmVXY1HBePvXzvhMnhSTmhmZ
GsiIaCIUxXhV1oyVj1O1jgbHRSnnWZqRTm3J1VTVhtkgS2RKejJVZG1Sb0kkXSISIC
CjhdFvUMVRZd1jJBdDRHTudZQZUVhWFnDjmNnHVJG1kcgODV3TTh2NjdfIiwigImZUT
XzcmdtrRUx3TDFTYnV2ZshIN3pCSO1Nd9v1aWY2MFnRszFeVhJVEiEaiCawhDN0g1Z
m4yZUJFTUtFa0U1Q2tibTizUX3REpsVFlvS1JpcDA4Zf1jYYyISICJ0cFV0bOdcwahBVX
3huncZAATBAhEdvlUlimx10MjxZ3Z2NUIZMEF4N0tj1I0sICJhZGRyZRNzIjogYjfc
2q0I1BdImdxqjVrBEF3xJSOdhSgQPQWPLVTWTDZKTo1hb2pa1OtaqVzMPGmIL
CAidk9ubFl0Y2pyODcyZlAzV2E3NU96bdjLTZTFTWZElvT3REttLep3MCISICJ3O
EludnhzUHyk5s29vd3WchHCTWdmsWl5X1iYyYcGeTZiLPSwnUJ119fSwgImIhelh
CI6IDE1M1TyYmZkwMjISICjLeHAi01AXNTE2MQ3MDIyLCAic2RfZGlzwnX0X2R1cm1c
YXRpb25fYXwnIjogInNoYS05ONTYiJfQ.1UHEPtULLUX0T51jH3g3c-3ZidWzSBu9n-VxmM
VldqtLhIBht6H6Jtt15p43yCXzdp1ZxtD16fr07TpoDy_Umg3QS_XFjFj4WhnsVvVzl
ASU8CFlg16xgH9D3W1g2hqepBS8DyQ5ba_p5Kn_tJKVoP1xWhcQujrRj8kkEKQsRia4F
hrBlB18F14wgu_ipPqNh1x4BV17GJC12Nz94nwPT7JUFkI6Y6kahlF356B0xtmLae
Y0gku88VeO2Nf1l cbo5k5kVtkarorfoL6D6Teiji__w-yY0PnIRjX0rYfOyohNI8LKL
AP38QYmpdR7z_rsvHpQHzFAPTmenvHDg
}
```
SD-JWT in 5 Simple Steps

Step 5: Base64url-encode Disclosures for Transport

```json
{
  "iss": "https://example.com",
  "type": "IdentityCredential",
  "cnf": {
    "jwk": {
      "kty": "RSA",
      "n": "0vx....Kgw",
      "e": "AQAB"
    }
  },
  "credentialSubject": {
    "address": {
      "street_address": "Musterstr. 23",
      "locality": "Berlin",
      "country": "DE"
    }
  }
}
```

→ Done!
Issuer

Issuance

SD-JWT plain-text claims + hashed Disclosures

Disclosures
salt + claim name + claim value

✓ signed by Issuer

End-User
(Holder)

Presentation

Selected Disclosures
salt + claim name + claim value

✓ signed by Issuer

Verifier

Holder-Binding JWT
nonce audience etc.

✓ signed by Holder

holder’s public key
Verification

- Verify SD-JWT signature
- Hash over disclosed Disclosures
- Find hash digests in SD-JWT
- Replace disclosed claims in SD-JWT
- Check holder binding, if required.

Done!
(Selected) Changes since -02

- **Nested disclosures** allow for arbitrary granularity for selectively disclosable claims
- **W3C VC-Data-Model** credential with JSON-LD
- **Improved various parts of the spec:**
  - Concepts section,
  - Security & Privacy Considerations,
  - Discussion on Holder Binding,
  - Context in Introduction,
  - Discussion on Canonicalization,
  - etc.
Discussion Points for -04

- Media Types
- Selective Disclosure of elements in the arrays
- SD-JWT with JWS using JSON serialization
Media Types

- As an identifier to signal it is an SD-JWT, since processing rules are different from JWS/JWT
- To be used in
  - HTTP requests and responses
  - typ in the Header of an SD-JWT
    - Payload of an SD-JWT can be any JSON.
    - W3C VC WG expected to define a media type for a payload (cty)
  - cty in the Header of JWE
- Proposals/ideas:
  - For the HTTP requests and responses, define sd-jwt-issuance and sd-jwt-presentation?
  - For the typ JOSE Header, define sd-jwt media type?
  - For general purpose, define +sd-jwt media type structured suffix?

GH Issues #236, #74; PR #229
Selective Disclosure for Individual Array Elements

Does the WG believe this feature is needed?

Some use cases for SD in arrays:

- Multi-value claims, like “nationalities”
- Selectively releasing a subset of the “evidence” documents supporting identity claims
- Using one holder binding method without releasing data for the others (e.g., release biometrics, but not public key)

GH Issue #194
Selective Disclosure of elements in the arrays?

Two approaches how this could be done, if the WG agreed the feature is needed:

Approach 1:

"nationalities": [
  "_sd",
  "7pHe1uQ5uSClgAxXdG0E6dKnBgXcxE01zvoQ09E5Lr4",
  "9-VdSnvRTZNDo-4Bxcp3X-V9VtLOCRUkR6oLWZQl81I"
]

New leading element indicates array of SD digests.

Approach 2:

"nationalities": [
  "_sd:7pHe1uQ5uSClgAxXdG0E6dKnBgXcxE01zvoQ09E5Lr4",
  "_sd:9-VdSnvRTZNDo-4Bxcp3X-V9VtLOCRUkR6oLWZQl81I"
]

String with defined prefix indicates SD hash.
SD-JWT with JWS using JSON Serialization?

Some use cases require JWS JSON serialization.

Example: ETSI JAdES signatures.

Proposal: Introduce JWS JSON serialization format for SD-JWT as optional feature (won’t be a breaking change)

GH Issue #198
SD-JWT with JWS using JSON serialization?

```
{
    "payload": "eyJpc3MiOiAiaHR0cHM6L...Z0NGpUOUYySFpRIn19fQ",
    "protected": "eyJhbGciOiAiRVMyNTYifQ",
    "header": {
        "kid": "e9bc097a-ce51-4036-9562-d2ade882db0d"
    },
    "signature": "mcndQ15m-4FbIzyfB...U2ZX7g",
    "disclosures": [
        "WyJkcVR2WE14UzBHYTNEb2FHbmU5eDBRIiwgInN1YiIsICJqb2huX2RvZV80MiJd",
        "WyIzanFjYjY3ejl3a3MwOHP3aUs3RX1RIiwgImdpdmVuX25hbWUiLCAiSm9obiJd",
        "WyJxUVdtakpsMXMxUjRscWhFTkxScnJ3IiwgImZhbWlseV9uYW11IiwgIkRvZSJd"
    ]
}
```

Payload as in SD-JWT

Disclosures
Unlinkability and Hash-based Selective Disclosure?

Inherent problem: Colluding verifiers can identify same user.

Solution: SD-JWTs are cheap!

- Issuer issues many SD-JWTs to Holder with same claims
- Holder uses a fresh SD-JWT per Verifier
Cryptographic Agility

SD-JWT does not prescribe any algorithms.

Signature Algorithm: All JOSE algorithms
Post-Quantum: see [draft-ietf-cose-post-quantum-signatures]

Hash Algorithm: Default SHA-256, algorithm identifier in SD-JWT
Compatibility

- Can be used with any JSON-based data format
  - W3C-VC Data Model
  - OpenID Connect for Identity Assurance (OIDC4IA)
- Flexibility regarding holder binding
  - External signature
  - Key distribution
- Makes no assumptions on the transport protocol
  - E.g., OIDC4VC
Available, Testable, Auditable

All examples in specification generated via reference implementation: oauthstuff/draft-selective-disclosure-jwt (Python)

Independent open-source implementations:

- Kotlin: IDunion/SD-JWT-Kotlin
- Rust: kushaldas/sd_jwt
- TypeScript: christianpaquin/sd-jwt
- TypeScript: chike0905/sd-jwt-ts
- Typescript: OR13/vc-sd-jwt NEW
- Java: authlete/sd-jwt NEW

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