SD-JWT

draft-ietf-oauth-selective-disclosure-jwt-03

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

	SD-JWT
Complexity	Selective disclosure, as simple as possible
Algorithms	Standard cryptography: JWS Signature + Hash function
Format	JWT & JSON
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": "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 2: Create *Disclosures*

```
{
 "iss": "https://example.com",
 "type": "IdentityCredential",
 "cnf": {"jwk": {"kty": "RSA","n": "0vx....Kgw","e": "AQAB" } },
 "credentialSubject": {
   "given_name": "Max",·····  ["GO0r26nO-iW50ZcAoOilFw", "given_name", "Max"]
   "family_name": "Mustermann",·····  ["cSlbR135iONjhsouMxrjjg", "family_name", "Mustermann"]
   "email": "mustermann@example.com",·····  ["oHDt43Vwuhpo8mzaprgCcw", "email", "mustermann@example.com"]
   "address": {
     "street_address": "Musterstr. 23",·····  ["rGcOKtY6WmflywTTKEWIEQ", "street_address", "Musterstr. 23"]
     "locality": "Berlin",·····  ["pGQMQx-2tH2XwC_eQCFn4g", "locality", "Berlin"]
     "country": "DE"·····  ["TI15M8G5UIxPiWNZ-VLYBA", "country", "DE"]
   }
  }
}
```

salt claim name claim value

SD-JWT in 5 Simple Steps

Step 3: Hash Disclosures & Replace Original Claims

```
{
 "iss": "https://example.com",
 "type": "IdentityCredential",
 "cnf": {"jwk": {"kty": "RSA","n": "0vx....Kgw","e": "AQAB" } },
 "credentialSubject": {
    "_sd": [ "EW1oOegqa5mGcbytT5S-kAubcEjYEUwRkXlu2vC5l2O",        ← ["GOOr26nO-iW5OZcAoOilFw", "given_name", "Max"]
             "FEx-ITHt41I8_cnOSS-hvoLneX_RGlJo_8o2xRNhfdk",        ← ["cSlbR135iONjhsouMxrjjg", "family_name", "Mustermann"]
             "igg7H5fn2eBEMIEkE5Ckbm23QuwDJlTYoKRip08dYIc" ],      ← ["oHDt43Vwuhpo8mzaprgCcw", "email", "mustermann@example.com"]
   "address": {
     "_sd": [ "gqB5kmAwyry88aHjaAeO-USX6JOMaojukKsheo38OOc",       ← ["rGcOKtY6WmflywTTKEWIEQ", "street_address", "Musterstr. 23"]
              "w8InvxsPXdKoowuVpyBMgl1b9_R2b6Xpa3OYOIjgQro",       ← ["pGQMQx-2tH2XwC_eQCFn4g", "locality", "Berlin"]
              "vOnlYtcjr872fP3Wa75Ozl7c-6_MOVdIUNtwLKKxZw0" ]      ← ["TI15M8G5UIxPiWNZ-VLYBA", "country", "DE"]
   }
 }
}
```

SD-JWT in 5 Simple Steps

Step 4: Sign SD-JWT & Encode for Transport

```
{
  "iss": "https://example.com",
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc
3N1ZXIiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJSU0EiLCAibiI6ICIwdng3YWdvZ
WJHY1FTdS4uLi4tY3NGQ3VyLWtFZ1U4YXdhcEp6S25xREtndyIsICJlIjogIkFRQUIif
X0sICJ0eXBlIjogIklkZW50aXR5Q3JlZGVudGlhbCIsICJjcmVkZW50aWFsU3ViamVjd
CI6IHsiX3NkIjogWyJFVzFvMGVncWE1bUdjYnl0VDVTLWtBdWJjRWpZRVV3UmtYbHUyd
kM1bDIwIiwgIkZFeC1JVEhONDFJOF9jbjBTUy1odm9MbmVYX1JHbEpvXzhvMnhSTmhmZ
GsiLCAiUXhKVi0yVjFIOG1jbHRSNnZWQzRtM3JlVTVhTkg5d2RKejJVZG1SbOkxRSIsI
CJhdFVuMVRZd1JBbDRHUTdQZUV0WGFNdzJmNHVJVGlKclgOODV3TTh2NjdFIiwgImZUT
XczdmtrRUx3TDFYTnVZSzhIN3pCS0NIdV91aWY2MFNsRzFweVhJVVEiLCAiaWdnNOg1Z
m4yZUJFTUlFa0U1Q2tibTIzUXV3REpsVFlvS1JpcDA4ZFlJYyIsICJ0cFV0bDcwaHBVX
3hucnZaaTBHaEdvUlIxam1OMXpZZ3Z2NUlZMEF4N0tjIl0sICJhZGRyZXNzIjogeyJfc
2QiOiBbImdxQjVrbUF3eXJ5ODhhSGphQWVPLVVTWDZKT01hb2p1a0tzaGVvMzhPMGMiL
CAidk9ubFlOY2pyODcyZlAzV2E3NU96bDdjLTZfTU9WZElVTnR3TEtLeFp3MCIsICJ3O
EludnhzUFhkS29vd3VWcHlCTWdsMWI5X1IyYjZYcGEzT1lPSWpnUXJvIl19fSwgImlhd
CI6IDE1MTYyMzkwMjIsICJleHAiOiAxNTE2MjQ3MDIyLCAic2RfZGlnZXN0X2Rlcml2Y
XRpb25fYWxnIjogInNoYS0yNTYifQ.1UHEPtLLUXOT51jH3gg-3C-ZidWzsB9Un-VxmM
VdQtTbLLhwDTB6HJtt15p43yCXTzdpiZxtDI6fr07TpODy_Umg3Q5_FxFj4WHnsVuVzu
ASU8cFlGPi6xgH9D3w1G2hqepBS8DyQ5bA_p5kN_tKJVoP1xWhcQujRJ8kkEKQsRia4F
hrBldl8f41wgu_ipPqh1Ix4BVI7GJClZNx94nWPT7JUFkI6Y6JkahLf3S6gBOMxtmLAe
YOqkuz8VeOZNfl_CDog55kVTkArorfoL6D6TEjI__-w6YyUOPnIRJXJOwrYfoyhNl8LK
AP38QYMpdR7z_rsvHpQHzFAPTmevnHDg
  ]
}
```

← ["GOOr26nO-iW50ZcAoOilFw", "given_name", "Max"]
← ["cSlbR135iONjhsouMxrjjg", "family_name", "Mustermann"]
← ["oHDt43Vwuhpo8mzaprgCcw", "email", "mustermann@example.com"]

← ["rGcOKtY6WmflywTTKEWIEQ", "street_address", "Musterstr. 23"]
← ["pGQMQx-2tH2XwC_eQCFn4g", "locality", "Berlin"]
← ["TI15M8G5UIxPiWNZ-VLYBA", "country", "DE"]

SD-JWT in 5 Simple Steps

Step 5: Base64url-encode Disclosures for Transport

```
{
  "iss": "https://example.com",
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc
3N1ZXIiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJSU0EiLCAibiI6ICIwdng3YWdvZ
WJHY1FTdS4uLi4tY3NGQ3VyLWtFZ1U4YXdhcEp6S25xREtndyIsICJlIjogIkFRQUIif
X0sICJ0eXBlIjogIklkZW50aXR5Q3JlZGVudGlhbCIsICJjcmVkZW50aWFsU3ViamVjd
CI6IHsiX3NkIjogWyJFVzFvMGVncWE1bUdjYnl0VDVTLWtBdWJjRWpZRVV3UmtYbHUyd
kM1bDIwIiwgIkZFeC1JVEh0NDFJOF9jbjBTUy1odm9MbmVYX1JHbEpvXzhvMnhSTmhmZ
GsiLCAiUXhKViOyVjFIOG1jbHRSNnZWQzRtM3JlVTVhTkg5d2RKejJVZG1SbOkxRSIsI
CJhdFVuMVRZd1JBbDRHUTdQZUVOWGFNdzJmNHVJVGlKclg0ODV3TTh2NjdFIiwgImZUT
XczdmtrRUx3TDFYTnVZSzhIN3pCS0NIdV91aWY2MFNsRzFweVhJVVEiLCAiaWdnNOg1Z
m4yZUJFTUlFaOU1Q2tibTIzUXV3REpsVFlvS1JpcDA4ZFlJYyIsICJOcFVObDcwaHBVX
3hucnZaaTBHaEdvUlIxam1OMXpZZ3Z2NUlZMEF4NOtjIlOsICJhZGRyZXNzIjogeyJfc
2QiOiBbImdxQjVrbUF3eXJ5ODhhSGphQWVPLVVTWDZKT01hb2p1aOtzaGVvMzhPMGMiL
CAidk9ubFlOY2pyODcyZlAzV2E3NU96bDdjLTZfTU9WZElVTnR3TEtLeFp3MCIsICJ3O
EludnhzUFhkS29vd3VWcHlCTWdsMWI5X1IyYjZYcGEzT1lPSWpnUXJvIl19fSwgImlhd
CI6IDE1MTYyMzkwMjIsICJleHAiOiAxNTE2MjQ3MDIyLCAic2RfZGlnZXNOX2Rlcml2Y
XRpb25fYWxnIjogInNoYS0yNTYifQ.1UHEPtLLUXOT51jH3gg-3C-ZidWzsB9Un-VxmM
VdQtTbLLhwDTB6HJtt15p43yCXTzdpiZxtDI6fr07TpODy_Umg3Q5_FxFj4WHnsVuVzu
ASU8cFlGPi6xgH9D3w1G2hqepBS8DyQ5bA_p5kN_tKJVoP1xWhcQujRJ8kkEKQsRia4F
hrBldl8f41wgu_ipPqh1Ix4BVI7GJClZNx94nWPT7JUFkI6Y6JkahLf3S6gBOMxtmLAe
YOqkuz8VeOZNfl_CDog55kVTkArorfoL6D6TEjI__-w6YyUOPnIRJXJOwrYfoyhNl8LK
AP38QYMpdR7z_rsvHpQHzFAPTmevnHDg
  }
}
```

```
← ["GO0r26nO-iW50ZcAoOilFw", "given_name", "Max"]
~WyJHTzByMjZuTy1pVzUwWmNBbO9pbEZ3IiwgImdpdmVuX25hbWUiLCAiTWF4Il0
~WyJjU2xiUjEzNWkwTmpoc291TXhyampnIiwgImZhbWlseV9uYW1lIiwgIk11c3R
lcm1hbm4iXQ
~WyJvSER0NDNWd3VocG84bXphcHJnQ2N3IiwgImVtYWlsIiwgIm11c3Rlcm1hbm5
AZXhhbXBsZS5jb20iXQ
~WyJyR2MwS3RZNldtZmx5d1RUS0VXSUVRIiwgInN0cmVldF9hZGRyZXNzIiwgIk1
1c3RlcnN0ci4gMjMiXQ
~WyJwR1FNUXgtMnRIMlh3Q19lUUNGbjRnIiwgImxvY2FsaXR5IiwgIkJlcmxpbiJ
d
~WyJUSTE1TThHNVVJeFBpV05aLVZMWUJBIiwgImNvdW50cnkiLCAiREUiXQ
```

→ Done!

Issuer

Issuance

SD-JWT
plain-text claims
+ hashed Disclosures

✓ signed by Issuer

Disclosures
salt + claim name + claim value

End-User
(Holder)

Presentation

Verifier

Issuer

Issuance

End-User
(Holder)

Presentation

Verifier

SD-JWT
plain-text claims
+ hashed Disclosures

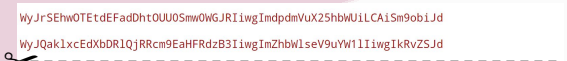
✓ signed by Issuer

Disclosures
salt + claim name + claim value

SD-JWT
plain-text claims
+ hashed Disclosures

✓ signed by Issuer

<u>Selected</u> Disclosures
salt + claim name + claim value



Issuer

Issuance

End-User
(Holder)

Presentation

Verifier

SD-JWT
plain-text claims
+ hashed Disclosures

✓ signed
by Issuer

Disclosures
salt + claim name + claim value

SD-JWT
plain-text claims
+ hashed Disclosures

✓ signed
by Issuer

<u>Selected</u> Disclosures
salt + claim name + claim value

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.

Verification requires hash check!

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",
   "7pHe1uQ5uSClgAxXdG0E6dKnBgXcxEO1zvoQO9E5Lr4",
   "9-VdSnvRTZNDo-4Bxcp3X-V9VtLOCRUkR6oLWZQl81I"
]
```

New leading element indicates array of SD digests.

Approach 2:

```
"nationalities": [
   "_sd:7pHe1uQ5uSClgAxXdG0E6dKnBgXcxEO1zvoQO9E5Lr4",
   "_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

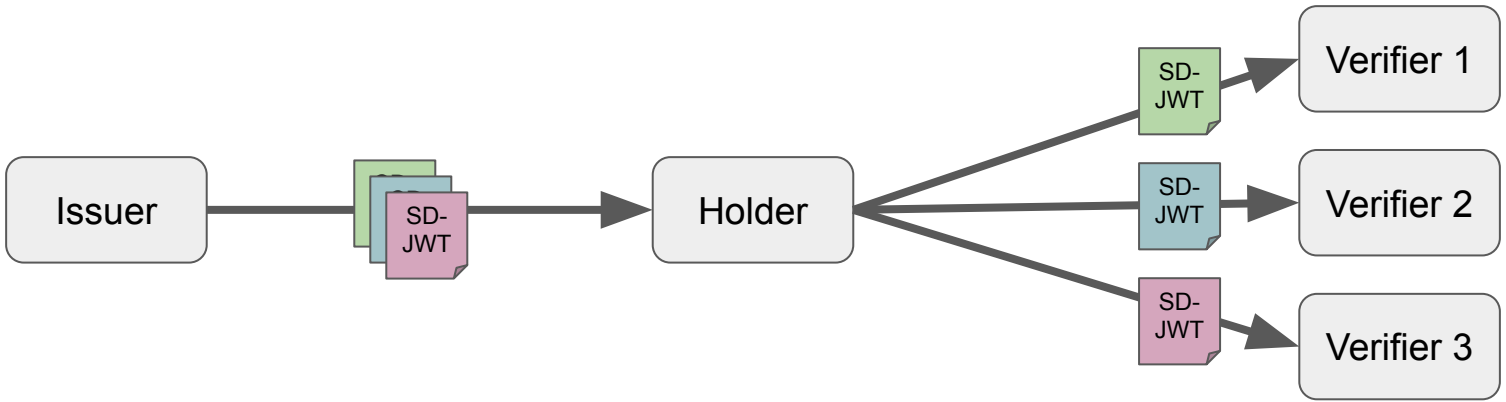
SD-JWT with JWS using JSON serialization?

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)

tooling might be separated into another GH repo in the future

```
### Produce SD-JWT
sdjwt = SDJWT(
   user_claims,
   issuer,
   ISSUER_KEY,
   HOLDER_KEY,
   iat,
   exp,
)
```

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**