SD-JWT

Reference Implementation

A project proposal for the Open Wallet Foundation

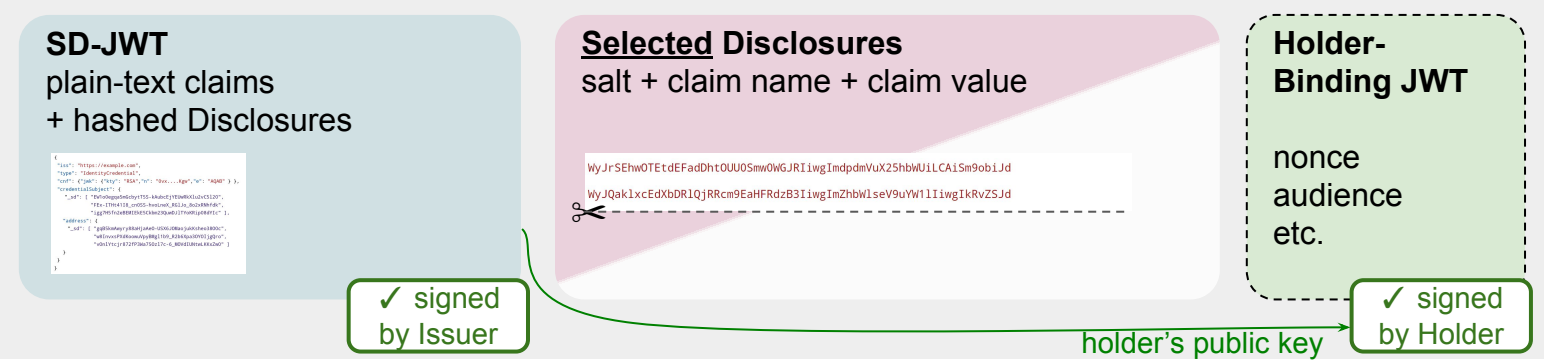
Dr. Daniel Fett
Authlete

AUTHLETE

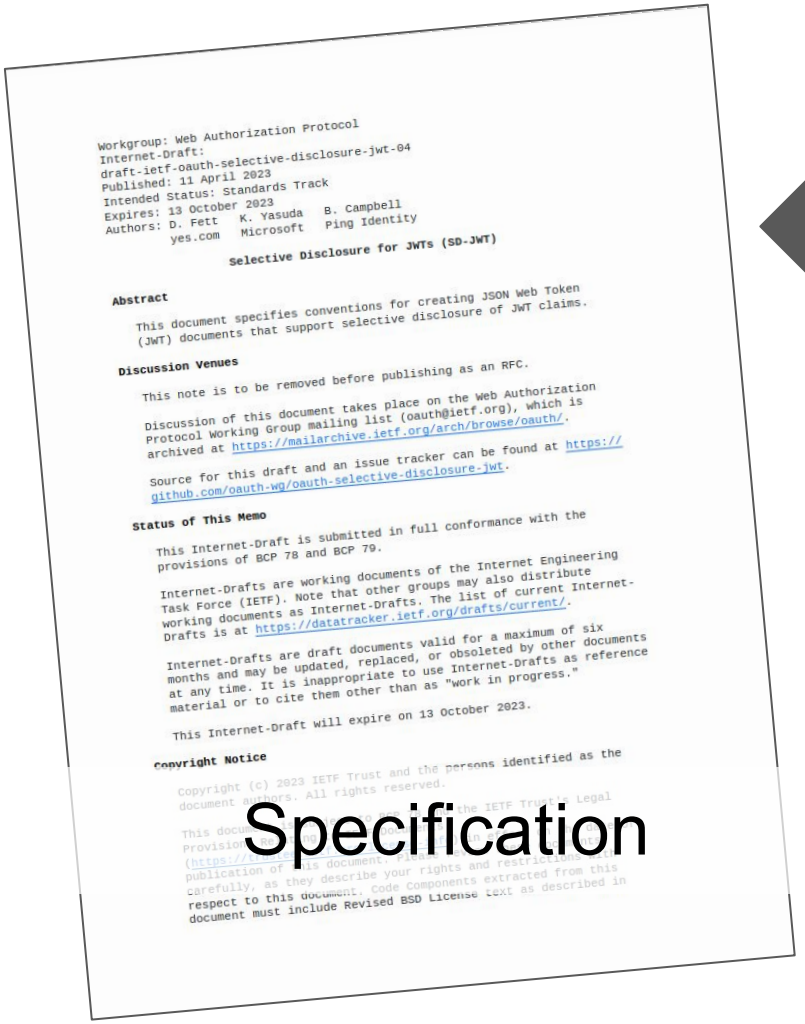
eic european identity and cloud conference 2023

‘Simple’ is a feature.

SD-JWT



A simple salted-hash based format enabling selective disclosure in JWTs for verifiable credentials and more.



Workgroup: Web Authorization Protocol
Internet-Draft:
draft-ietf-oauth-selective-disclosure-jwt-04
Published: 11 April 2023
Intended Status: Standards Track
Expires: 13 October 2023
Authors: D. Fett (yes.com), K. Yasuda (Microsoft), B. Campbell (Ping Identity)

Selective Disclosure for JWTs (SD-JWT)

Abstract

This document specifies conventions for creating JSON Web Token (JWT) documents that support selective disclosure of JWT claims.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/oauth-wg/oauth-selective-disclosure-jwt.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

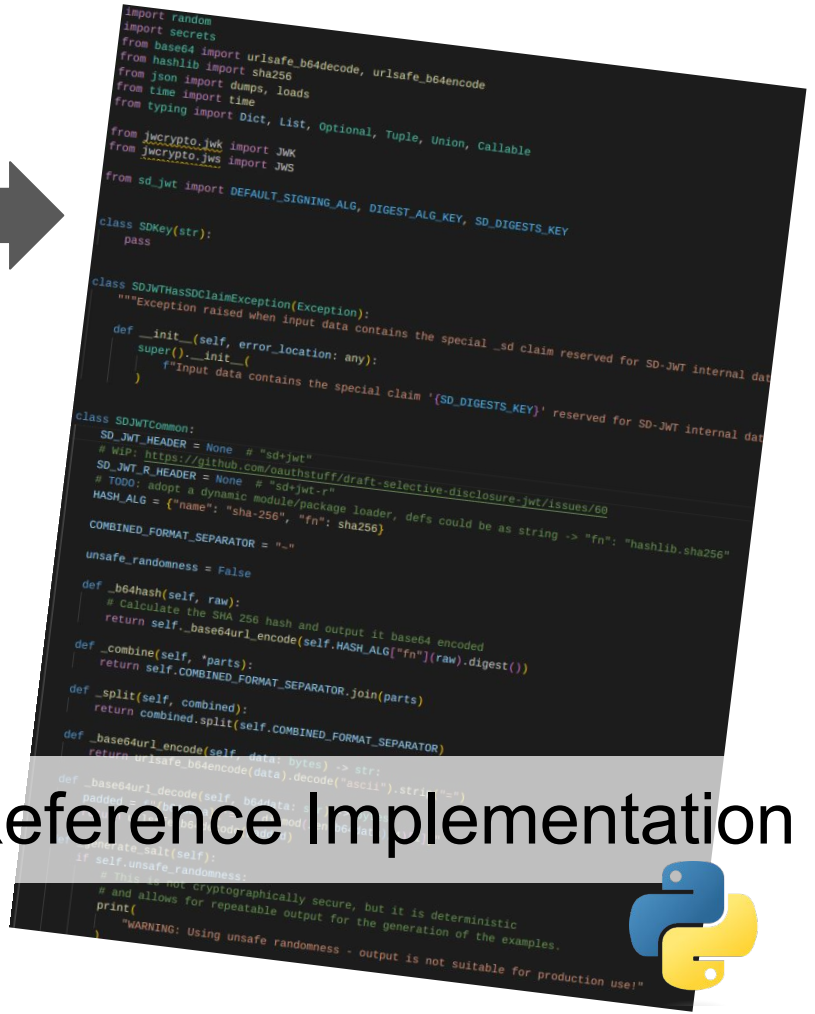
This Internet-Draft will expire on 13 October 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in

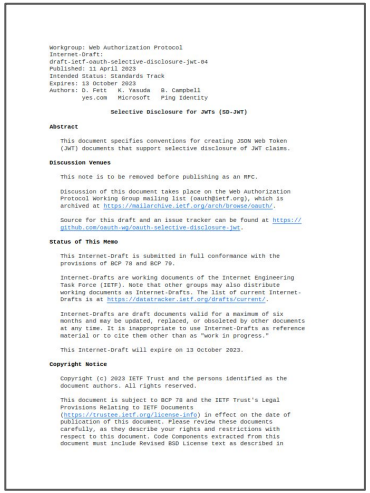
Specification



```python
import random
import secrets
from base64 import urlsafe_b64decode, urlsafe_b64encode
from hashlib import sha256
from json import dumps, loads
from time import time
from typing import Dict, List, Optional, Tuple, Union, Callable

from jwcrypto.jwk import JWK
from jwcrypto.jws import JWS

from sd_jwt import DEFAULT_SIGNING_ALG, DIGEST_ALG_KEY, SD_DIGESTS_KEY


class SDKey(str):
    pass


class SDJWTHasSDClaimException(Exception):
    """Exception raised when input data contains the special _sd claim reserved for SD-JWT internal dat

    def __init__(self, error_location: any):
        super().__init__(
            f"Input data contains the special claim '{SD_DIGESTS_KEY}' reserved for SD-JWT internal dat
        )


class SDJWTCommon:
    SD_JWT_HEADER = None  # "sd+jwt"
    # WIP: https://github.com/oauthstuff/draft-selective-disclosure-jwt/issues/60
    SD_JWT_R_HEADER = None  # "sd+jwt-r"
    # TODO: adopt a dynamic module/package loader, defs could be as string -> "fn": "hashlib.sha256"
    HASH_ALG = {"name": "sha-256", "fn": sha256}

    COMBINED_FORMAT_SEPARATOR = "~"

    unsafe_randomness = False

    def _b64hash(self, raw):
        # Calculate the SHA 256 hash and output it base64 encoded
        return self._base64url_encode(self.HASH_ALG["fn"](raw).digest())

    def _combine(self, *parts):
        return self.COMBINED_FORMAT_SEPARATOR.join(parts)

    def _split(self, combined):
        return combined.split(self.COMBINED_FORMAT_SEPARATOR)

    def _base64url_encode(self, data: bytes) -> str:
        return urlsafe_b64encode(data).decode("ascii").stri("=")

    def _base64url_decode(self, b64data: str) -> bytes:
        padded = b64data + "=" * (-len(b64data) % 4)
        return urlsafe_b64decode(padded)

    def _generate_salt(self):
        if self.unsafe_randomness:
            # This is not cryptographically secure, but it is deterministic
            # and allows for repeatable output for the generation of the examples.
            print(
                "WARNING: Using unsafe randomness - output is not suitable for production use!"
            )
```

Reference Implementation

Why a Reference Implementation?



Workgroup: Web Authorization Protocol
Internet-Draft:
draft-ietf-oauth-selective-disclosure-jwt-04
Published: 11 April 2023
Intended Status: Standards Track
Expires: 13 October 2023
Authors: D. Fett K. Yasuda B. Campbell
yes.com Microsoft Ping Identity

Selective Disclosure for JWTs (SD-JWT)

Abstract

This document specifies conventions for creating JSON Web Token (JWT) documents that support selective disclosure of JWT claims.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/oauth-wg/oauth-selective-disclosure-jwt.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 13 October 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in

<u>Production of examples</u>

for the specification.

<u>Test-driving</u>

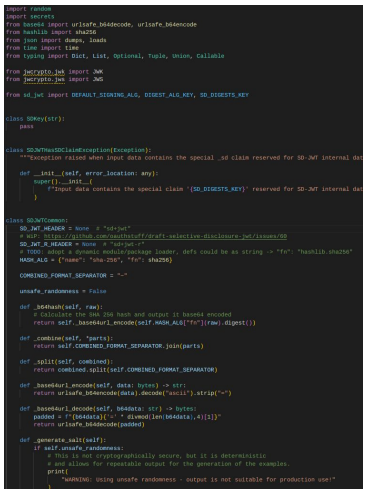
new features and modified syntax.

<u>Production of test-cases</u>

usable across projects.

<u>Demonstration</u>

of a production-ready library & interface.



```python
import random
import secrets
from base64 import urlsafe_b64decode, urlsafe_b64encode
from hashlib import sha256
from json import dumps, loads
from time import time
from typing import Dict, List, Optional, Tuple, Union, Callable

from jwcrypto.jwk import JWK
from jwcrypto.jws import JWS

from sd_jwt import DEFAULT_SIGNING_ALG, DIGEST_ALG_KEY, SD_DIGESTS_KEY


class SDKey(str):
    pass


class SDJWTHasSDClaimException(Exception):
    """Exception raised when input data contains the special _sd claim reserved for SD-JWT internal dat

    def __init__(self, error_location: any):
        super().__init__(
            f"Input data contains the special claim '{SD_DIGESTS_KEY}' reserved for SD-JWT internal dat
        )


class SDJWTCommon:
    SD_JWT_HEADER = None  # "sd+jwt"
    # alt: https://github.com/oauthstuff/draft-selective-disclosure-jwt/issues/69
    SD_JWT_R_HEADER = None  # "sd+jwt-r"
    # TODO: adopt a dynamic module/package loader. defs could be as string -> "fn": "hashlib.sha256"
    HASH_ALG = {"name": "sha-256", "fn": sha256}

    COMBINED_FORMAT_SEPARATOR = "~"

    unsafe_randomness = False

    def _b64hash(self, raw):
        # Calculate the SHA 256 hash and output it base64 encoded
        return self._base64url_encode(self.HASH_ALG["fn"](raw).digest())

    def _combine(self, *parts):
        return self.COMBINED_FORMAT_SEPARATOR.join(parts)

    def _split(self, combined):
        return combined.split(self.COMBINED_FORMAT_SEPARATOR)

    def _base64url_encode(self, data: bytes) -> str:
        return urlsafe_b64encode(data).decode("ascii").strip("=")

    def _base64url_decode(self, b64data: str) -> bytes:
        padded = f"{b64data}{'=' * divmod(len(b64data),4)[1]}"
        return urlsafe_b64decode(padded)

    def _generate_salt(self):
        if self.unsafe_randomness:
            # This is not cryptographically secure, but it is deterministic
            # and allows for repeatable output for the generation of the examples.
            print(
                "WARNING: Using unsafe randomness - output is not suitable for production use!"
            )
```

Production of Examples for the Specification

```
user_claims:
  !sd sub: john_doe_42
  !sd given_name: John
  !sd family_name: Doe
  !sd email: johndoe@example.com
  !sd phone_number: +1-202-555-0101
  !sd address:
       street_address: 123 Main St
       locality: Anytown
       region: Anystate
       country: US
  !sd birthdate: "1940-01-01"

holder_disclosed_claims:
  { "given_name": null,
    "family_name": null,
    "address": {} }

holder_binding: True
```

Example properties (example)

→ **SD-JWT Payload**

→ **Format for Issuance**

→ **Format for Presentation**

→ **Detailed output for explanation in specification text**

5.1.1.5.1. Option 1: Flat SD-JWT

The Issuer can decide to treat the `address` claim as a block that can either be disclosed completely or not at all. The following example shows that in this case, the entire `address` claim is treated as an object in the Disclosure.

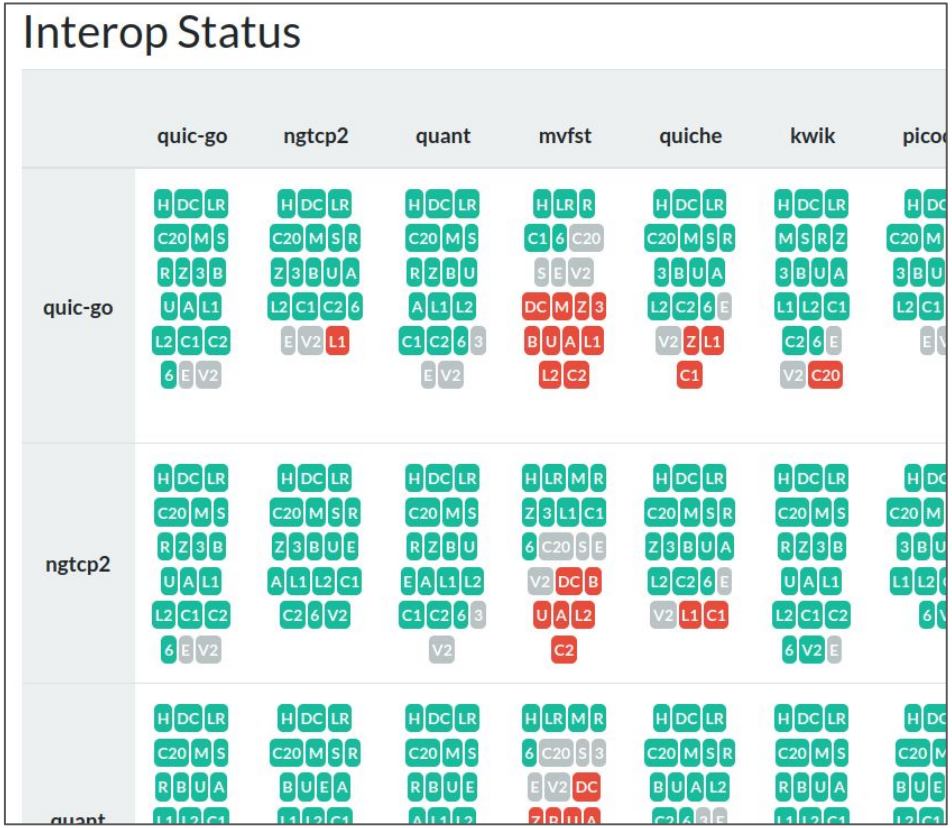
```
{
  "_sd": [
    "FphFFpj1vtr0rpYK-14fickGKMg3zf1fIpJXxTK8PAE"
  ],
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1735689661,
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c"
```

Production of Test-Cases

Test case data can be used to check other implementations.

Next step:
Separate test data repository.

Idea:
Centralized cross-implementation interoperability testing
— hosted at & supported by OWF?



Interop testing matrix of QUIC: A model for us?

SD-JWT Reference Implementation in the OWF

- Open source vs. open governance
- Project proposal submitted — “Labs” status
- Ongoing: Extraction of code into separate Github repository
- Ongoing: Merge with Python library by Hakan Yildiz (Accenture)
- Maintenance: For now, mostly as before — IETF draft editors & contributors
- There is room for more implementations!
  - Other languages
  - Other feature sets
  - Other frameworks/dependencies
  - …

SD-JWT IETF Draft

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

Daniel Fett
Authlete

Kristina Yasuda
Microsoft

Brian Campbell
Ping

Dr. Daniel Fett
Authlete Inc.
daniel.fett@authlete.com
Twitter: @dfett42
Linkedin:

Thank you!