Workgroup: Web Authorization Protocol
Internet-Draft:
draft-fett-oauth-selective-disclosure-jwt-01
Published: 23 June 2022
Intended Status: Standards Track
Expires: 25 December 2022
Authors: D. Fett K. Yasuda
yes.com Microsoft

Selective Disclosure JWT (SD-JWT)

Abstract

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

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 25 December 2022.

Copyright Notice

Table of Contents

1. Introduction
   1.1. Conventions and Terminology
2. Terms and Definitions
3. Flow Diagram
4. Concepts
   4.1. Creating an SD-JWT
   4.2. Creating an SD-JWT Release
   4.3. Optional Holder Binding
   4.4. Verifying an SD-JWT Release
5. Data Formats
   5.1. Format of an SD-JWT
      5.1.1. sd_digests Claim (Digests of Selectively Disclosable Claims)
      5.1.2. Hash Function Claim
      5.1.3. Holder Public Key Claim
   5.2. Example 1: SD-JWT
   5.3. Format of a SD-JWT Salt/Value Container (SVC)
   5.4. Example: SVC for the Flat SD-JWT in Example 1
   5.5. Sending SD-JWT and SVC during Issuance
   5.6. Format of an SD-JWT Release
   5.7. Example: SD-JWT Release for Example 1
   5.8. Sending SD-JWT and SD-JWT-R during Presentation
6. Verification
7. Security Considerations
   7.1. Mandatory signing of the SD-JWT
   7.2. Entropy of the salt
   7.3. Minimum length of the salt
   7.4. Choice of a hash function
   7.5. Holder Binding
8. Privacy Considerations
   8.1. Claim Names
   8.2. Unlinkability
9. Acknowledgements
10. IANA Considerations
11. Normative References
12. Informative References

Appendix A. Additional Examples
   A.1. Example 2 - Structured SD-JWT
   A.2. Example 3 - Complex Structured SD-JWT
   A.3. Example 4 - W3C Verifiable Credentials Data Model

Appendix B. Document History

Authors' Addresses

1. Introduction

The JSON-based claims in a signed JSON Web Token (JWT) [RFC7519] document are secured against modification using JSON Web Signature

(JWS) [RFC7515] digital signatures. A consumer of a signed JWT document that has checked the document's signature can safely assume that the contents of the document have not been modified. However, anyone receiving an unencrypted JWT can read all of the claims and likewise, anyone with the decryption key receiving an encrypted JWT can also read all of the claims.

This document describes a format for signed JWTs that support selective disclosure (SD-JWT), enabling sharing only a subset of the claims included in the original signed JWT instead of releasing all the claims to every verifier. During issuance, an SD-JWT is sent from the issuer to the holder alongside an SD-JWT Salt/Value Container (SVC), a JSON object that contains the mapping between raw claim values contained in the SD-JWT and the salts for each claim value.

This document also defines a format for SD-JWT Releases (SD-JWT-R), which convey a subset of the claim values of an SD-JWT to the verifier. For presentation, the holder creates an SD-JWT-R and sends it together with the SD-JWT to the verifier. To verify claim values received in SD-JWT-R, the verifier uses the salts values in the SD-JWT-R to compute the hashes of the claim values and compare them to the hashes in the SD-JWT.

One of the common use cases of a signed JWT is representing a user's identity created by an issuer. As long as the signed JWT is one-time use, it typically only contains those claims the user has consented to release to a specific verifier. However, when a signed JWT is intended to be multi-use, it needs to contain the superset of all claims the user might want to release to verifiers at some point. The ability to selectively disclose a subset of these claims depending on the verifier becomes crucial to ensure minimum disclosure and prevent verifiers from obtaining claims irrelevant for the transaction at hand.

One example of such a multi-use JWT is a verifiable credential, a tamper-evident credential with a cryptographically verifiable authorship that contains claims about a subject. SD-JWTs defined in this document enable such selective disclosure of claims.

While JWTs for claims describing natural persons are a common use case, the mechanisms defined in this document can be used for many other use cases as well.

This document also describes holder binding, or the concept of binding SD-JWT to key material controlled by the subject of SD-JWT. Holder binding is optional to implement.

1.1. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

base64url denotes the URL-safe base64 encoding without padding defined in Section 2 of [RFC7515].

2. Terms and Definitions

Selective Disclosure JWT (SD-JWT) A JWT [RFC7515] created by the issuer, which is signed as a JWS [RFC7515], that supports selective disclosure as defined in this document.

SD-JWT Salt/Value Container (SVC) A JSON object created by the issuer that contains mapping between raw claim values contained in the SD-JWT and the salts for each claim value.

SD-JWT Release (SD-JWT-R) A JWT created by the holder that contains a subset of the claim values of an SD-JWT in a verifiable way.

Holder binding Ability of the holder to prove legitimate possession of SD-JWT by proving control over the same private key during the issuance and presentation. SD-JWT signed by the issuer contains a public key or a reference to a public key that matches to the private key controlled by the holder.

Issuer An entity that creates SD-JWTs (2.1).

Holder An entity that received SD-JWTs (2.1) from the issuer and has control over them.

Verifier An entity that requests, checks and extracts the claims from SD-JWT-R (2.2)

Note: discuss if we want to include Client, Authorization Server for the purpose of ensuring continuity and separating the entity from the actor.

3. Flow Diagram

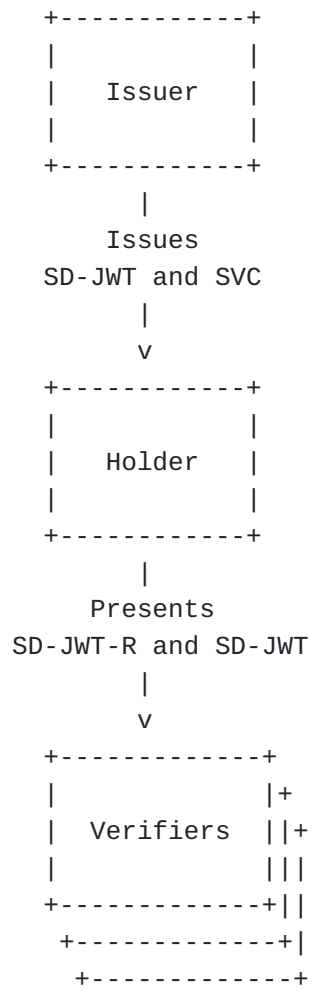

Figure 1: SD-JWT Issuance and Presentation Flow

4. Concepts

In the following, the contents of SD-JWTs and SD-JWT Releases are described at a conceptual level, abstracting from the data formats described afterwards.

4.1. Creating an SD-JWT

An SD-JWT, at its core, is a digitally signed document containing hashes over the claim values with unique salts and other metadata. It MUST be digitally signed using the issuer's private key.

```
SD-JWT-DOC = (METADATA, SD-CLAIMS)
SD-JWT = SD-JWT-DOC | SIG(SD-JWT-DOC, ISSUER-PRIV-KEY)
```

SD-CLAIMS can be a simple object with claim names mapped to hashes over the claim values with unique salts:

```
SD-CLAIMS = (
    CLAIM-NAME: HASH(SALT | CLAIM-VALUE)
)*
```

SD-CLAIMS can also be nested deeper to capture more complex objects, as will be shown later.

SD-JWT is sent from the issuer to the holder, together with the mapping of the plain-text claim values, the salt values, and potentially some other information.

4.2. Creating an SD-JWT Release

To disclose to a verifier a subset of the SD-JWT claim values, a holder creates a JWT such as the following:

```
SD-JWT-RELEASE-DOC = (METADATA, SD-RELEASES)
SD-JWT-RELEASE = SD-JWT-RELEASE-DOC
```

SD-RELEASES follows the structure of SD-CLAIMS and can be a simple object with claim names mapped to values and salts:

```
SD-RELEASES = (
    CLAIM-NAME: (DISCLOSED-SALT, DISCLOSED-VALUE)
)
```

Just as SD-CLAIMS, SD-RELEASES can be more complex as well.

SD-JWT-RELEASE is sent together with SD-JWT from the holder to the verifier.

4.3. Optional Holder Binding

Some use-cases may require holder binding.

If holder binding is desired, SD-JWT must contain information about key material controlled by the holder:

```
SD-JWT-DOC = (METADATA, HOLDER-PUBLIC-KEY, SD-CLAIMS)
```

Note: How the public key is included in SD-JWT is out of scope of this document. It can be passed by value or by reference. Examples in this document use sub_jwt Claim to include raw public key by value in SD-JWT.

With holder binding, the SD-JWT-RELEASE is signed by the holder using its private key. It therefore looks as follows:

```
SD-JWT-RELEASE = SD-JWT-RELEASE-DOC | SIG(SD-JWT-RELEASE-DOC, HOLDER-PRI
```

4.4. Verifying an SD-JWT Release

A verifier checks that

- for each claim in SD-JWT-RELEASE, the hash HASH(DISCLOSED-SALT | DISCLOSED-VALUE) matches the hash under the given claim name in SD-JWT.

- if holder binding is desired, the SD-JWT-RELEASE was signed by the private key belonging to HOLDER-PUBLIC-KEY.

The detailed algorithm is described below.

5. Data Formats

This section defines data formats for SD-JWTs (containing hashes of the salted claim values), SD-JWT Salt/Value Containers (containing the mapping of the plain-text claim values and the salt values), and SD-JWT Releases (containing a subset of the same mapping).

5.1. Format of an SD-JWT

An SD-JWT is a JWT that MUST be signed using the issuer's private key. The payload of an SD-JWT MUST contain the sd_digests and hash_alg claims described in the following, and MAY contain a holder's public key or a reference thereto, as well as further claims such as iss, iat, etc. as defined or required by the application using SD-JWTs.

5.1.1. sd_digests Claim (Digests of Selectively Disclosable Claims)

An SD-JWT MUST include hashes of the salted claim values that are included by the issuer under the property sd_digests.

The issuer MUST choose a unique salt value for each claim value. Each salt value MUST contain at least 128 bits of pseudorandom data, making it hard for an attacker to guess. The salt value MUST then be encoded as a string. It is RECOMMENDED to base64url-encode at least 16 pseudorandom bytes.

The issuer MUST build the hashes by hashing over a string that is formed by JSON-encoding an ordered array containing the salt and the claim value, e.g.: ["6qMQvRL5haj","Peter"]. The hash value is then base64url-encoded. Note that the precise JSON encoding can vary, and therefore, the JSON encodings MUST be sent to the holder along with the SD-JWT, as described below.

5.1.1.1. Flat and Structured sd_digests objects

The sd_digests object can be a 'flat' object, directly containing all claim names and hashed claim values without any deeper structure. The sd_digests object can also be a 'structured' object, where some claims and their respective hashes are contained in places deeper in the structure. It is at the issuer's discretion whether to use a 'flat' or 'structured' sd_digests SD-JWT object, and how to structure it such that it is suitable for the use case.

Example 1 below is a non-normative example of an SD-JWT using a 'flat' sd_digests object and Example 2 in the appendix shows a non-normative example of an SD-JWT using a 'structured' sd_digests object. The difference between the examples is how the address claim is disclosed.

Appendix 2 shows a more complex example using claims from eKYC (todo: reference).

5.1.2. Hash Function Claim

The claim hash_alg indicates the hash algorithm used by the Issuer to generate the hashes of the salted claim values. The hash algorithm identifier MUST be a value from the "Hash Name String" column in the IANA "Named Information Hash Algorithm" registry [IANA.Hash.Algorithms]. SD-JWTs with hash algorithm identifiers not found in this registry are not considered valid and MUST NOT be accepted by verifiers.

5.1.3. Holder Public Key Claim

If the issuer wants to enable holder binding, it MAY include a public key associated with the holder, or a reference thereto.

It is out of the scope of this document to describe how the holder key pair is established. For example, the holder MAY provide a key pair to the issuer, the issuer MAY create the key pair for the holder, or holder and issuer MAY use pre-established key material.

Note: need to define how holder public key is included, right now examples are using sub_jwk I think.

5.2. Example 1: SD-JWT

This example and Example 2 in the appendix use the following object as the set of claims that the Issuer is issuing:

```
{
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "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"
}
```

The following non-normative example shows the payload of an SD-JWT. The issuer is using a flat structure, i.e., all of the claims the address claim can only be disclosed in full.

```
{
  "iss": "https://example.com/issuer",
  "sub_jwk": {
    "kty": "RSA",
    "n": "pm4bOHBg-oYhAyPWzR56AWX3rUIXp11_ICDkGgS6W3ZWLts-hzwI3x65659kg4hVo9dbGoCJE3ZGF_eaetE
    "e": "AQAB"
  },
  "hash_alg": "sha-256",
  "iat": 1516239022,
  "exp": 1516247022,
  "sd_digests": {
    "sub": "LbnhkOr5oS7KjeUrxezAu8TG0CpWz0jSixy6tffuo04",
    "given_name": "fUMdn88aaoyKTHrvZd6AuLmPraGhPJ0zF5r_JhxCVZs",
    "family_name": "9h5vgv6TpFV6GmnPtugiMLl5tHetHeb5X_2cKHjN7cw",
    "email": "fPZ92dtYMCN2Nb-2ac_zSH19p4yakUXrZl_-wSgaazA",
    "phone_number": "QdSffzNzzd0n60MsSmuiKj6Y6Enk2b-BS-KtEePde5M",
    "address": "JFu99NUXPq55f6DFBZ22rMkxMNHayCrfPG0FDsqbyDs",
    "birthdate": "Ia1Tc6_Xnt5CJc2LtKcu6Wvqr42glBGGcjGOye8Zf3U"
  },
  "hash_alg": "sha-256"
}
```

The SD-JWT is then signed by the issuer to create a document like the following:

eyJhbGciOiAiUlMyNTYifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXI
iLCAic3ViX2p3ayI6IHsia3R5IjogIlJTQSIsICJuIjogInBtNGJPSEJnLW9ZaEF5UFd6U
jU2QVdYM3JVSVhwMTFfSUNEa0dnUzZXM1pXTHRzLWh6d0kzeDY1NjU5a2c0aFZvOWRiR29
DSkUzWkdGX2VhZXRFMzBVaEJVRWdwR3dyRHJRaUo5enFwcm1jRmZyM3F2dmtHanR0aDhaZ
2wxZU0yYkpjT3dFN1BDQkhXVEtXWXMxNTJSN2c2SmcyT1ZwaC1hOHJxLXE3OU1oS0c1UW9
XX21UejEwUVRfNkg0YzdQaldHMWZqaDhocFdObmJQX3B2NmQxelN3WmZjNWZsNnlWUkwwR
FYwVjNsR0hLZTJXcWZfZU5HakJyQkxWa2xEVGs4LXN0WF9NV0xjUi1FR21YQU92MFVCV2l
0U19kWEpLSnUtdlhKeXcxNG5IU0d1eFRJSzJoeDFwdHRNZnQ5Q3N2cWltWEtlRFRVMTRxU
UwxZUU3aWhjdyIsICJlIjogIkFRQUIifSwgImlhdCI6IDE1MTYyMzkwMjIsICJleHAiOiA
xNTE2MjQ3MDIyLCAiX3NkIjogeyJzdWIiOiAiTGJuaGtPcjVvUzdLamVVcnhlekF1OFRHM
ENwV3owalNpeHk2dGZmdW8wNCIsICJnaXZlbl9uYW1lIjogImZVTWRuODhhYW95S1RIcnZ
aZDZBdUxtUHJhR2hQSjB6RjVyX0poeENWWnMiLCAiZmFtaWx5X25hbWUiOiAiOWg1dmd2N
lRwRlY2R21uUHR1Z2lNTGw1dEhldEhlYjVYXzJjS0hqTjdjdyIsICJlbWFpbCI6ICJmUFo
5MmR0WU1DTjJOYi0yYWNfelNIMTlwNHlha1VYclpsXy13U2dhYXpBIiwgInBob25lX251b
WJlciI6ICJRZFNmZnpOenpkMG42ME1zU211aUtqNlk2RW5rMmItQlMtS3RFZVBkZTVNIiw
gImFkZHJlc3MiOiAiSkZ1OTlOVVhQcTU1ZjZERkJaMjJyTWt4TU5IYXlDcmZQRzBGRHNxY
nlEcyIsICJiaXJ0aGRhdGUiOiAiSWExVGM2X1hudDVDSmMyTHRLY3U2V3ZxcjQyZ2xCR0d
jakdPeWU4WmYzVSJ9LCAiaGFzaF9hbGciOiAic2hhLTI1NiJ9.FfUbeF_py0aTQ6XHamsX
PNd0LopmZPXdA3iOaCpK9G11SUrzEo8BrmzsqHCbEIDXUxM_CXjslvBYaFF0J374Xwyuxw
t-tLsnXUEglqze_mzEKyvgC9rpHp18Fmcuv3KYYMD6-c6_yBPMS5ocf4v-Hn-VBzJLs81g
BP1QiCgHIoFE8OHBBiZK3ynoqWEWskx_4Q1-8cDUprPmWYF0ik7LgnQwgoZKig-_AVqTma
ckYaRT1PjLW8ULgKBb6jgmkq-ge_yMGz0SM_i-ZbjPUIJb9WC5jqb7dXyXDALFEDiArv6n
a59mbieucYA0wKIWleuww0ci4hOfrnA-dQxPcMKYvw

(Line breaks for presentation only.)

5.3. Format of a SD-JWT Salt/Value Container (SVC)

Besides the SD-JWT itself, the holder needs to learn the raw claim values that are contained in the SD-JWT, along with the precise input to the hash calculation, and the salts. There MAY be other information the issuer needs to communicate to the holder, such as a private key if the issuer selected the holder key pair.

A SD-JWT Salt/Value Container (SVC) is a JSON object containing at least the top-level property sd_release. Its structure mirrors the one of sd_digests in the SD-JWT, but the values are the inputs to the hash calculations the issuer used, as strings.

The SVC MAY contain further properties, for example, to transport the holder private key.

5.4. Example: SVC for the Flat SD-JWT in Example 1

The SVC for Example 1 is as follows:

```
{
  "sd_release": {
    "sub": "[\"eluV5Og3gSNII8EYnsxA_A\", \"6c5c0a49-b589-431d-bae7-219122a9ec2c\"]",
    "given_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"John\"]",
    "family_name": "[\"eI8ZWm9QnKPpNPeNenHdhQ\", \"Doe\"]",
    "email": "[\"Qg_O64zqAxe412a108iroA\", \"johndoe@example.com\"]",
    "phone_number": "[\"AJx-095VPrpTtN4QMOqROA\", \"+1-202-555-0101\"]",
    "address": "[\"Pc33JM2LchcU_lHggv_ufQ\", {\"street_address\": \"123 Main St\", \"locality
    "birthdate": "[\"G02NSrQfjFXQ7Io09syajA\", \"1940-01-01\"]"
  }
}
```

Important: As described above, hashes are calculated over the string formed by serializing a JSON array containing the salt and the claim value. This ensures that issuer and verifier use the same input to their hash functions and avoids issues with canonicalization of JSON values that would lead to different hash values. The SVC therefore maps claim names to JSON-encoded arrays.

5.5. Sending SD-JWT and SVC during Issuance

For transporting the SVC together with the SD-JWT from the issuer to the holder, the SVC is base64url-encoded and appended to the SD-JWT using a period character . as the separator. For Example 1, the combined format looks as follows:

eyJhbGciOiAiUlMyNTYifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXI
iLCAic3ViX2p3ayI6IHsia3R5IjogIlJTQSIsICJuIjogInBtNGJPSEJnLW9ZaEF5UFd6U
jU2QVdYM3JVSVhwMTFfSUNEa0dnUzZXM1pXTHRzLWh6d0kzeDY1NjU5a2c0aFZvOWRiR29
DSkUzWkdGX2VhZXRFMzBVaEJVRWdwR3dyRHJRaUo5enFwcm1jRmZyM3F2dmtHanR0aDhaZ
2wxZU0yYkpjT3dFN1BDQkhXVEtXWXMxNTJSN2c2SmcyT1ZwaC1hOHJxLXE3OU1oS0c1UW9
XX21UejEwUVRfNkg0YzdQaldHMWZqaDhocFdObmJQX3B2NmQxelN3WmZjNWZsNnlWUkwwR
FYwVjNsR0hLZTJXcWZfZU5HakJyQkxWa2xEVGs4LXN0WF9NV0xjUi1FR21YQU92MFVCV2l
0U19kWEpLSnUtdlhKeXcxNG5IU0d1eFRJSzJoeDFwdHRNZnQ5Q3N2cWltWEtlRFRVMTRxU
UwxZUU3aWhjdyIsICJlIjogIkFRQUIifSwgImlhdCI6IDE1MTYyMzkwMjIsICJleHAiOiA
xNTE2MjQ3MDIyLCAiX3NkIjogeyJzdWIiOiAiTGJuaGtPcjVvUzdLamVVcnhlekF1OFRHM
ENwV3owalNpeHk2dGZmdW8wNCIsICJnaXZlbl9uYW1lIjogImZVTWRuODhhYW95S1RIcnZ
aZDZBdUxtUHJhR2hQSjB6RjVyX0poeENWWnMiLCAiZmFtaWx5X25hbWUiOiAiOWg1dmd2N
lRwRlY2R21uUHR1Z2lNTGw1dEhldEhlYjVYXzJjS0hqTjdjdyIsICJlbWFpbCI6ICJmUFo
5MmR0WU1DTjJOYi0yYWNfelNIMTlwNHlha1VYclpsXy13U2dhYXpBIiwgInBob25lX251b
WJlciI6ICJRZFNmZnpOenpkMG42ME1zU211aUtqNlk2RW5rMmItQlMtS3RFZVBkZTVNIiw
gImFkZHJlc3MiOiAiSkZ1OTlOVVhQcTU1ZjZERkJaMjJyTWt4TU5IYXlDcmZQRzBGRHNxY
nlEcyIsICJiaXJ0aGRhdGUiOiAiSWExVGM2X1hudDVDSmMyTHRLY3U2V3ZxcjQyZ2xCR0d
jakdPeWU4WmYzVSJ9LCAiaGFzaF9hbGciOiAic2hhLTI1NiJ9.FfUbeF_py0aTQ6XHamsX
PNd0LopmZPXdA3iOaCpK9G11SUrzEo8BrmzsqHCbEIDXUxM_CXjslvBYaFF0J374Xwyuxw
t-tLsnXUEglqze_mzEKyvgC9rpHp18Fmcuv3KYYMD6-c6_yBPMS5ocf4v-Hn-VBzJLs81g
BP1QiCgHIoFE8OHBBiZK3ynoqWEWskx_4Q1-8cDUprPmWYF0ik7LgnQwgoZKig-_AVqTma
ckYaRT1PjLW8ULgKBb6jgmkq-ge_yMGz0SM_i-ZbjPUIJb9WC5jqb7dXyXDALFEDiArv6n
a59mbieucYA0wKIWleuww0ci4hOfrnA-dQxPcMKYvw.ewogICAgIl9zZCI6IHsKICAgICA
gICAic3ViIjogIltcImVsdVY1T2czZ1NOSUk4RVluc3hBX0FcIiwgXCI2YzVjMGE0OS1iN
Tg5LTQzMWQtYmFlNy0yMTkxMjJhOWVjMmNcIl0iLAogICAgICAgICJnaXZlbl9uYW1lIjo
gIltcIjZJajd0TS1hNWlWUEdib1M1dG12VkFcIiwgXCJKb2huXCJdIiwKICAgICAgICAiZ
mFtaWx5X25hbWUiOiAiW1wiZUk4WldtOVFuS1BwTlBlTmVuSGRoUVwiLCBcIkRvZVwiXSI
sCiAgICAgICAgImVtYWlsIjogIltcIlFnX082NHpxQXhlNDEyYTEwOGlyb0FcIiwgXCJqb
2huZG9lQGV4YW1wbGUuY29tXCJdIiwKICAgICAgICAicGhvbmVfbnVtYmVyIjogIltcIkF
KeC0wOTVWUHJwVHRONFFNT3FST0FcIiwgXCIrMS0yMDItNTU1LTAxMDFcIl0iLAogICAgI
CAgICJhZGRyZXNzIjogIltcIlBjMzNKTTJMY2hjVV9sSGdndl91ZlFcIiwge1wic3RyZWV
0X2FkZHJlc3NcIjogXCIxMjMgTWFpbiBTdFwiLCBcImxvY2FsaXR5XCI6IFwiQW55dG93b
lwiLCBcInJlZ2lvblwiOiBcIkFueXN0YXRlXCIsIFwiY291bnRyeVwiOiBcIlVTXCJ9XSI
sCiAgICAgICAgImJpcnRoZGF0ZSI6ICJbXCJHMDJOU3JRZmpGWFE3SW8wOXN5YWpBXCIsI
FwiMTk0MC0wMS0wMVwiXSIKICAgIH0KfQ

(Line breaks for presentation only.)

5.6. Format of an SD-JWT Release

SD-JWT-R contains claim values and the salts of the claims that the holder has consented to release to the Verifier. This enables the Verifier to verify the claims received from the holder by computing the hash sof the claims values and the salts revealed in the SD-JWT-R using the hashing algorithm specified in SD-JWT and comparing them to the hash valued included in SD-JWT.

For each claim, an array of the salt and the claim value is contained in the _sd object. The structure of _sd object in the SD-JWT-R is the same as in SD-JWT.

The SD-JWT-R MAY contain further claims, for example, to ensure a binding to a concrete transaction (in the example the nonce and aud claims).

When the holder sends the SD-JWT-R to the Verifier, the SD-JWT-R MUST be a JWS represented as the JWS Compact Serialization as described in Section 7.1 of [RFC7515].

If holder binding is desired, the SD-JWT-R is signed by the holder. If no holder binding is to be used, the none algorithm is used, i.e., the document is not signed. TODO: Change to plain base64 to avoid alg=none issues

5.7. Example: SD-JWT Release for Example 1

The following is a non-normative example of the contents of an SD-JWT-R for Example 1:


```
{
  "nonce": "2GLC42sKQveCfGfryNRN9w",
  "aud": "https://example.com/verifier",
  "sd_release": {
    "given_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"John\"]",
    "family_name": "[\"eI8ZWm9QnKPpNPeNenHdhQ\", \"Doe\"]",
    "address": "[\"Pc33JM2LchcU_lHggv_ufQ\", {\"street_address\": \"123 Main St\", \"locality
  }
}
```


For each claim, an array of the salt and the claim value is contained in the sd_release object.

Again, the SD-JWT-R follows the same structure as the sd_digests in the SD-JWT.

Below is a non-normative example of a representation of the SD-JWT-R JWS Compact Serialization:

eyJhbGciOiAiUlMyNTYifQ.eyJub25jZSI6ICIyR0xDNDJzS1F2ZUNmR2ZyeU5STjl3Iiw
gImF1ZCI6ICJodHRwczovL2V4YW1wbGUuY29tL3ZlcmlmaWVyIiwgIl9zZCI6IHsiZ2l2Z
W5fbmFtZSI6ICJbXCI2SWo3dE0tYTVpVlBHYm9TNXRtdlZBXCIsIFwiSm9oblwiXSIsICJ
mYW1pbHlfbmFtZSI6ICJbXCJlSThaV205UW5LUHBOUGVOZW5IZGhRXCIsIFwiRG9lXCJdI
iwgImFkZHJlc3MiOiAiW1wiUGMzM0pNMkxjaGNVX2xIZ2d2X3VmUVwiLCB7XCJzdHJlZXR
fYWRkcmVzc1wiOiBcIjEyMyBNYWluIFN0XCIsIFwibG9jYWxpdHlcIjogXCJBbnl0b3duX
CIsIFwicmVnaW9uXCI6IFwiQW55c3RhdGVcIiwgXCJjb3VudHJ5XCI6IFwiVVNcIn1dIn1
9.b0hG3v71rzHvtoDTdroZ9m-lt9tf8nobFKb2YGiyGOjIklfcKc2KWj72oi_tBKcOCqZh
dX6IV4BRXIw-aspQfLh-xBrNLuGqiC-Y3rZBBlWw0WWnbbtsy1tj8yZOiXBr8vO6mCgZGA
d4MgPYPd-QzOr9ukObYDRB4I24xHrqlAEYPJIzSw9MI_dEmIkNnAuIfLQKiuyTqVVVp6Ly
pBIz6fBLm6NOLC4-uVXlOzI91iT4zlkrhP0-vj8TmfB-XL9aD3-xqytvLBHTESct49OSRZ
FrwkLUKTM56_6KW3pG7Ucuv8VnpHXHIka0SGRaOh8x6v5-rCQJl_IbM8wb7CSHvQ

(Line breaks for presentation only.)

5.8. Sending SD-JWT and SD-JWT-R during Presentation

The SD-JWT and the SD-JWT-R can be combined into one document using period character . as a separator (here for Example 1):

eyJhbGciOiAiUlMyNTYifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXI
iLCAic3ViX2p3ayI6IHsia3R5IjogIlJTQSIsICJuIjogInBtNGJPSEJnLW9ZaEF5UFd6U
jU2QVdYM3JVSVhwMTFfSUNEa0dnUzZXM1pXTHRzLWh6d0kzeDY1NjU5a2c0aFZvOWRiR29
DSkUzWkdGX2VhZXRFMzBVaEJVRWdwR3dyRHJRaUo5enFwcm1jRmZyM3F2dmtHanR0aDhaZ
2wxZU0yYkpjT3dFN1BDQkhXVEtXWXMxNTJSN2c2SmcyT1ZwaC1hOHJxLXE3OU1oS0c1UW9
XX21UejEwUVRfNkg0YzdQaldHMWZqaDhocFdObmJQX3B2NmQxelN3WmZjNWZsNnlWUkwwR
FYwVjNsR0hLZTJXcWZfZU5HakJyQkxWa2xEVGs4LXN0WF9NV0xjUi1FR21YQU92MFVCV2l
0U19kWEpLSnUtdlhKeXcxNG5IU0d1eFRJSzJoeDFwdHRNZnQ5Q3N2cWltWEtlRFRVMTRxU
UwxZUU3aWhjdyIsICJlIjogIkFRQUIifSwgImlhdCI6IDE1MTYyMzkwMjIsICJleHAiOiA
xNTE2MjQ3MDIyLCAiX3NkIjogeyJzdWIiOiAiTGJuaGtPcjVvUzdLamVVcnhlekF1OFRHM
ENwV3owalNpeHk2dGZmdW8wNCIsICJnaXZlbl9uYW1lIjogImZVTWRuODhhYW95S1RIcnZ
aZDZBdUxtUHJhR2hQSjB6RjVyX0poeENWWnMiLCAiZmFtaWx5X25hbWUiOiAiOWg1dmd2N
lRwRlY2R21uUHR1Z2lNTGw1dEhldEhlYjVYXzJjS0hqTjdjdyIsICJlbWFpbCI6ICJmUFo
5MmR0WU1DTjJOYi0yYWNfelNIMTlwNHlha1VYclpsXy13U2dhYXpBIiwgInBob25lX251b
WJlciI6ICJRZFNmZnpOenpkMG42ME1zU211aUtqNlk2RW5rMmItQlMtS3RFZVBkZTVNIiw
gImFkZHJlc3MiOiAiSkZ1OTlOVVhQcTU1ZjZERkJaMjJyTWt4TU5IYXlDcmZQRzBGRHNxY
nlEcyIsICJiaXJ0aGRhdGUiOiAiSWExVGM2X1hudDVDSmMyTHRLY3U2V3ZxcjQyZ2xCR0d
jakdPeWU4WmYzVSJ9LCAiaGFzaF9hbGciOiAic2hhLTI1NiJ9.FfUbeF_py0aTQ6XHamsX
PNd0LopmZPXdA3iOaCpK9G11SUrzEo8BrmzsqHCbEIDXUxM_CXjslvBYaFF0J374Xwyuxw
t-tLsnXUEglqze_mzEKyvgC9rpHp18Fmcuv3KYYMD6-c6_yBPMS5ocf4v-Hn-VBzJLs81g
BP1QiCgHIoFE8OHBBiZK3ynoqWEWskx_4Q1-8cDUprPmWYF0ik7LgnQwgoZKig-_AVqTma
ckYaRT1PjLW8ULgKBb6jgmkq-ge_yMGz0SM_i-ZbjPUIJb9WC5jqb7dXyXDALFEDiArv6n
a59mbieucYA0wKIWleuww0ci4hOfrnA-dQxPcMKYvw.eyJhbGciOiAiUlMyNTYifQ.eyJu
b25jZSI6ICIyR0xDNDJzS1F2ZUNmR2ZyeU5STjl3IiwgImF1ZCI6ICJodHRwczovL2V4YW
1wbGUuY29tL3ZlcmlmaWVyIiwgIl9zZCI6IHsiZ2l2ZW5fbmFtZSI6ICJbXCI2SWo3dE0t
YTVpVlBHYm9TNXRtdlZBXCIsIFwiSm9oblwiXSIsICJmYW1pbHlfbmFtZSI6ICJbXCJlST
haV205UW5LUHBOUGVOZW5IZGhRXCIsIFwiRG9lXCJdIiwgImFkZHJlc3MiOiAiW1wiUGMz
M0pNMkxjaGNVX2xIZ2d2X3VmUVwiLCB7XCJzdHJlZXRfYWRkcmVzc1wiOiBcIjEyMyBNYW
luIFN0XCIsIFwibG9jYWxpdHlcIjogXCJBbnl0b3duXCIsIFwicmVnaW9uXCI6IFwiQW55
c3RhdGVcIiwgXCJjb3VudHJ5XCI6IFwiVVNcIn1dIn19.b0hG3v71rzHvtoDTdroZ9m-lt
9tf8nobFKb2YGiyGOjIklfcKc2KWj72oi_tBKcOCqZhdX6IV4BRXIw-aspQfLh-xBrNLuG
qiC-Y3rZBBlWw0WWnbbtsy1tj8yZOiXBr8vO6mCgZGAd4MgPYPd-QzOr9ukObYDRB4I24x
HrqlAEYPJIzSw9MI_dEmIkNnAuIfLQKiuyTqVVVp6LypBIz6fBLm6NOLC4-uVXlOzI91iT
4zlkrhP0-vj8TmfB-XL9aD3-xqytvLBHTESct49OSRZFrwkLUKTM56_6KW3pG7Ucuv8Vnp
HXHIka0SGRaOh8x6v5-rCQJl_IbM8wb7CSHvQ

(Line breaks for presentation only.)

6. Verification

Verifiers MUST follow [RFC8725] for checking the SD-JWT and, if signed, the SD-JWT Release.

Verifiers MUST go through (at least) the following steps before trusting/using any of the contents of an SD-JWT:

1. Determine if holder binding is to be checked for the SD-JWT. Refer to Section 7.5 for details.

2. Check that the presentation consists of six period-separated (.) elements; if holder binding is not required, the last element can be empty.

3. Separate the SD-JWT from the SD-JWT Release.

4. Validate the SD-JWT:

    1. Ensure that a signing algorithm was used that was deemed secure for the application. Refer to [RFC8725], Sections 3.1 and 3.2 for details.

    2. Validate the signature over the SD-JWT.

    3. Validate the issuer of the SD-JWT and that the signing key belongs to this issuer.

    4. Check that the SD-JWT is valid using nbf, iat, and exp claims, if provided in the SD-JWT.

    5. Check that the claim sd_digests is present in the SD-JWT.

    6. Check that the hash_alg claim is present and its value is understand and the hash algorithm deemed secure.

5. Validate the SD-JWT Release:

    1. If holder binding is required, validate the signature over the SD-JWT using the same steps as for the SD-JWT plus the following steps:

        1. Determine that the public key for the private key that used to sign the SD-JWT-R is bound to the SD-JWT, i.e., the SD-JWT either contains a reference to the public key or contains the public key itself.

        2. Determine that the SD-JWT-R is bound to the current transaction and was created for this verifier (replay protection). This is usually achieved by a nonce and aud field within the SD-JWT Release.

    2. For each claim in the SD-JWT Release:

        1. Ensure that the claim is present as well in sd_release in the SD-JWT. If sd_release is structured, the claim MUST be present at the same place within the structure.

        2. Compute the base64url-encoded hash of a claim revealed from the Holder using the claim value and

the salt included in the SD-JWT-R and the hash_alg in SD-JWT.

3. Compare the hah computed in the previous step with the hash of the same claim in SD-JWT. Accept the claim only when the two hashes match.

4. Ensure that the claim value in the SD-JWT-R is a JSON-encoded array of exactly two values.

5. Store the second of the two values.

3. Once all necessary claims have been verified, their values can be validated and used according to the requirements of the application. It MUST be ensured that all claims required for the application have been released.

If any step fails, the input is not valid and processing MUST be aborted.

7. Security Considerations

7.1. Mandatory signing of the SD-JWT

The SD-JWT MUST be signed by the issuer to protect integrity of the issued claims. An attacker can modify or add claims if an SD-JWT is not signed (e.g., change the "email" attribute to take over the victim's account or add an attribute indicating a fake academic qualification).

The verifier MUST always check the SD-JWT signature to ensure that the SD-JWT has not been tampered with since its issuance. If the signature on the SD-JWT cannot be verified, the SD-JWT MUST be rejected.

7.2. Entropy of the salt

The security model relies on the fact that the salt is not learned or guessed by the attacker. It is vitally important to adhere to this principle. As such, the salt has to be created in such a manner that it is cryptographically random, long enough and has high entropy that it is not practical for the attacker to guess.

7.3. Minimum length of the salt

The length of the randomly-generated portion of the salt MUST be at least 128 bits.

7.4. Choice of a hash function

For the security of this scheme, the hash function is required to have the following property. Given a claim value, a salt, and the resulting hash, it is hard to find a second salt value so that HASH(salt | claim_value) equals the hash.

Furthermore the hash algorithms MD2, MD4, MD5, RIPEMD-160, and SHA-1 revealed fundamental weaknesses and they MUST NOT be used.

7.5. Holder Binding

TBD

8. Privacy Considerations

8.1. Claim Names

Claim names are not hashed in the SD-JWT and are used as keys in a key-value pair, where the value is the hash. This is because SD-JWT already reveals information about the issuer and the schema, and revealing the claim names does not provide any additional information.

8.2. Unlinkability

It is also important to note that this format enables selective disclosure of claims, but in itself it does not achieve unlinkability of the subject of an SD-SWT.

9. Acknowledgements

We would like to thank ...

10. IANA Considerations

TBD

11. Normative References

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12. Informative References

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/info/rfc8725>.

[VC_DATA] Sporny, M., Noble, G., Longley, D., Burnett, D. C., Zundel, B., and D. Chadwick, "Verifiable Credentials Data Model 1.0", 19 November 2019, <https://www.w3.org/TR/vc_data>.

Appendix A. Additional Examples

A.1. Example 2 - Structured SD-JWT

This non-normative example is based on the same claim values as Example 1, but this time the issuer decided to create a structured object for the hashes. This allows for the release of individual members of the address claim separately.

```
{
  "iss": "https://example.com/issuer",
  "sub_jwk": {
    "kty": "RSA",
    "n": "pcHdUSmbR3A8_eJcxaOWtk8wmrsxP7Fpl1DYVeNJRRYBS2kHLewBLAG4CpZDAB-AuuIkaGRyJdcISfN0Uj
  },
  "hash_alg": "sha-256",
  "iat": 1516239022,
  "exp": 1516247022,
  "sd_digests": {
    "sub": "LbnhkOr5oS7KjeUrxezAu8TG0CpWz0jSixy6tffuo04",
    "given_name": "fUMdn88aaoyKTHrvZd6AuLmPraGhPJ0zF5r_JhxCVZs",
    "family_name": "9h5vgv6TpFV6GmnPtugiMLl5tHetHeb5X_2cKHjN7cw",
    "email": "fPZ92dtYMCN2Nb-2ac_zSH19p4yakUXrZl_-wSgaazA",
    "phone_number": "QdSffzNzzd0n60MsSmuiKj6Y6Enk2b-BS-KtEePde5M",
    "address": {
      "street_address": "4FpVpd563Owh9G3HkGNTN9FiSHT0e6y9-Abk_IuG86M",
      "locality": "Kr0BpdZz6yU8HMhjyYHh1EEgJxeUyLIpJEi47iXhp8Y",
      "region": "QXxWKvcV4Bc9t3M7MF43W5vdCnWtA9hsYX8ycWLu1LQ",
      "country": "3itkoMzrDrinn7T0MUbAmrMm1ya1LzbBgif_50WoFOs"
    },
    "birthdate": "fvLCnDm3r4VSYcBF3pIlXP4ulEoHuHOfG_YmFZEuxpQ"
  },
  "hash_alg": "sha-256"
}
```


The SVC for this SD-JWT is as follows:


```
{
  "sd_release": {
    "sub": "[\"eluV5Og3gSNII8EYnsxA_A\", \"6c5c0a49-b589-431d-bae7-219122a9ec2c\"]",
    "given_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"John\"]",
    "family_name": "[\"eI8ZWm9QnKPpNPeNenHdhQ\", \"Doe\"]",
    "email": "[\"Qg_O64zqAxe412a108iroA\", \"johndoe@example.com\"]",
    "phone_number": "[\"AJx-095VPrpTtN4QMOqROA\", \"+1-202-555-0101\"]",
    "address": {
      "street_address": "[\"Pc33JM2LchcU_lHggv_ufQ\", \"123 Main St\"]",
      "locality": "[\"G02NSrQfjFXQ7Io09syajA\", \"Anytown\"]",
      "region": "[\"lklxF5jMYlGTPUovMNIvCA\", \"Anystate\"]",
      "country": "[\"nPuoQnkRFq3BIeAm7AnXFA\", \"US\"]"
    },
    "birthdate": "[\"5bPs1IquZNa0hkaFzzzZNw\", \"1940-01-01\"]"
  }
}
```

An SD-JWT-R for the SD-JWT above that discloses only region and country of the address property:

```
{
  "nonce": "2GLC42sKQveCfGfryNRN9w",
  "aud": "https://example.com/verifier",
  "sd_release": {
    "given_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"John\"]",
    "family_name": "[\"eI8ZWm9QnKPpNPeNenHdhQ\", \"Doe\"]",
    "birthdate": "[\"5bPs1IquZNa0hkaFzzzZNw\", \"1940-01-01\"]",
    "address": {
      "region": "[\"lklxF5jMYlGTPUovMNIvCA\", \"Anystate\"]",
      "country": "[\"nPuoQnkRFq3BIeAm7AnXFA\", \"US\"]"
    }
  }
}
```

A.2. Example 3 - Complex Structured SD-JWT

In this example, a complex object such as those used for OIDC4IDA (todo reference) is used.

In this example, the Issuer is using a following object as a set of claims to issue to the Holder:

```
{
  "verified_claims": {
    "verification": {
      "trust_framework": "de_aml",
      "time": "2012-04-23T18:25Z",
      "verification_process": "f24c6f-6d3f-4ec5-973e-b0d8506f3bc7",
      "evidence": [
        {
          "type": "document",
          "method": "pipp",
          "time": "2012-04-22T11:30Z",
          "document": {
            "type": "idcard",
            "issuer": {
              "name": "Stadt Augsburg",
              "country": "DE"
            },
            "number": "53554554",
            "date_of_issuance": "2010-03-23",
            "date_of_expiry": "2020-03-22"
          }
        }
      ]
    },
    "claims": {
      "given_name": "Max",
      "family_name": "Meier",
      "birthdate": "1956-01-28",
      "place_of_birth": {
        "country": "DE",
        "locality": "Musterstadt"
      },
      "nationalities": [
        "DE"
      ],
      "address": {
        "locality": "Maxstadt",
        "postal_code": "12344",
        "country": "DE",
        "street_address": "An der Weide 22"
      }
    }
  },
  "birth_middle_name": "Timotheus",
  "salutation": "Dr.",
  "msisdn": "49123456789"
}
```

The following shows the resulting SD-JWT payload:

```
{
  "iss": "https://example.com/issuer",
  "sub_jwk": {
    "kty": "RSA",
    "n": "6bh1cbaN-fyTUTfawCabGpTSeHOWmsHuB-VZ0aoAKBAfH6MOeLolLYLMcgP1VCNa1CenudLRzm8ULvinMi
    "e": "AQAB"
  },
  "hash_alg": "sha-256",
  "iat": 1516239022,
  "exp": 1516247022,
  "sd_digests": {
    "verified_claims": {
      "verification": {
        "trust_framework": "UI-SRNlQFy-YEFE46yyHKqc64jmM65q8ma9cq2V_erY",
        "time": "jI-FYlteydXzsjRIrXBZs9foBSNF1Od1Q-4XnuqpgjI",
        "verification_process": "F979I7b5ZhADtyYMlYxctdc9-IalD_Td0HpfcFBzVXs",
        "evidence": [
          {
            "type": "i2w3mrKAQV2nhTa5c2koZ-aQTBDoSaVfvYk7aLQianc",
            "method": "fEQ0tVPD67GfO30h_SRs8ZPbnZ_vwEt5S8lUOR77va0",
            "time": "9jueDP5r0gTB64DqdCZbek3yaS5AJJnW8FEkWtPTaOk",
            "document": {
              "type": "K-rZQk89w89YBhjUNUho07suLxhG8Sl2JTPAcoAJB34",
              "issuer": {
                "name": "BkCULCU-txVGvzNqnWe5DxefFvJE8LMib8GV3I3WO90",
                "country": "DSyF5TtmYgLk92u4GkDQzSdFbvIbw5rkFjzSsJJsyw4"
              },
              "number": "epH3OuU51TBelOE4PX6ueHwr1ZtoUjzG-7pZjIAsXg8",
              "date_of_issuance": "cVvqTueVq6OWz-dJj2cdo19A0Ajj859eGDzDfwPYyN4",
              "date_of_expiry": "nxJBNdtwvb2TKKJNGvF6_1ywEdKrotj66C88WPomLfo"
            }
          }
        ]
      },
      "claims": {
        "given_name": "y9uFPHAVqNAZ7PJyk1-1yQJZZWZzKGP5FLt9txKM84M",
        "family_name": "XyUikY8V8MWeBfXUOp8gI7F7-yC28Jr5IyDgvBxXzd4",
        "birthdate": "7GlieMLJhM78C_uQQp9wUXSZLeqBN1YGQT87BIubyKU",
        "place_of_birth": {
          "country": "RN3xcnLYX_GDhVwfPvtisuLPfi0d74zqihFbQrd_UG0",
          "locality": "iNkpWqJ9kIZQq95dzSyEZjbPJs6Fqu7GFBKouEC3OxE"
        },
        "nationalities": "-tinYGK0GXnkfARxiNIWq0VnzNRl-Kv3KY3m5g5Femg",
        "address": "63EzPV0yvTpeOgV34yCwweCvO-2wxts2Wqbja_SuwPQ"
      }
    },
    "birth_middle_name": "vM68I6XnrVlyt1LxK9xxgFycsjtw2vLdGpNgk3E8QQ4",
    "salutation": "iThfCu2ulLoe5i6gCEq--Y6R-gxHHtIukXb9qnfjH5k",
    "msisdn": "xUpU-azBYdXeJidc8Yw5MXtfPz4_4kArJhflXcxzkzs"
```

```
  },
  "hash_alg": "sha-256"
}
```

The SD-JWT is then signed by the issuer to create a document like the following:

eyJhbGciOiAiUlMyNTYifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXI
iLCAic3ViX2p3ayI6IHsia3R5IjogIlJTQSIsICJuIjogIjZiaDFjYmFOLWZ5VFVUZmF3Q
2FiR3BUU2VIT1dtc0h1Qi1WWjBhb0FLQkFmSDZNT2VMb2xMWUxNY2dQMVZDTmExQ2VudWR
MUnptOFVMdmluTWljbWZDT3VtdnlocTh3c2dIMWpJSkRHX1RWcnVwUzZpWnZ0aE90WFBwa
1NYeERpaUx4Z21uTFI1QWxwV0JLanJ6WG1YMWpRMlYxZ1FsQzJTMmVON3RfQ1ItSmZqM3l
iNHJUVzIwVXl2aHBPcHk2NDlDYVlzQW8zVWxxMm9KcG5HNnJJblhfWHpZSDg2cGxKbVlEV
XEwN1NvR0ljZFlZSzZJeXVkWkJoMmJWeFo5YUJ1dmtZUXkxMUFGdlhrMkJQMFJiVFAxYU9
3d3pTLUxXYWhRSmRzSzVPaVBMUnJmVFozOTloMkkyNFZXM3FYVDJadXNxOXRVTkFsT1d2a
DFLN3VtbnBNdyIsICJlIjogIkFRQUIifSwgImlhdCI6IDE1MTYyMzkwMjIsICJleHAiOiA
xNTE2MjQ3MDIyLCAiX3NkIjogeyJ2ZXJpZmllZF9jbGFpbXMiOiB7InZlcmlmaWNhdGlvb
iI6IHsidHJ1c3RfZnJhbWV3b3JrIjogIlVJLVNSTmxRRnktWUVGRTQ2eXlIS3FjNjRqbU0
2NXE4bWE5Y3EyVl9lclkiLCAidGltZSI6ICJqSS1GWWx0ZXlkWHpzalJJclhCWnM5Zm9CU
05GMU9kMVEtNFhudXFwZ2pJIiwgInZlcmlmaWNhdGlvbl9wcm9jZXNzIjogIkY5NzlJN2I
1WmhBRHR5WU1sWXhjdGRjOS1JYWxEX1RkMEhwZmNGQnpWWHMiLCAiZXZpZGVuY2UiOiBBe
yJ0eXBlIjogImkydzNtcktBUVYybmhUYTVjMmtvWi1hUVRCRG9TYVZmdllrN2FMUWlhbmM
iLCAibWV0aG9kIjogImZFUTB0VlBENjdHZk8zMGhfU1JzOFpQYm5aX3Z3RXQ1UzhsVU9SN
zd2YTAiLCAidGltZSI6ICI5anVlRFA1cjBnVEI2NERxZENaYmVrM3lhUzVBSkpuVzhGRWt
XdFBUYU9rIiwgImRvY3VtZW50IjogeyJ0eXBlIjogIkstclpRazg5dzg5WUJoalVOVWhvM
DdzdUx4aEc4U2wySlRQQWNvQUpCMzQiLCAiaXNzdWVyIjogeyJuYW1lIjogIkJrQ1VMQ1U
tdHhWR3Z6TnFuV2U1RHhlZkZ2SkU4TE1pYjhHVjNJM1dPOTAiLCAiY291bnRyeSI6ICJEU
3lGNVR0bVlnTGs5MnU0R2tEUXpTZEZidklidzVya0ZqelNzSkpzeXc0In0sICJudW1iZXI
iOiAiZXBIM091VTUxVEJlbE9FNFBYNnVlSHdyMVp0b1VqekctN3BaaklBc1hnOCIsICJkY
XRlX29mX2lzc3VhbmNlIjogImNWdnFUdWVWcTZPV3otZEpqMmNkbzE5QTBBamo4NTllR0R
6RGZ3UFl5TjQiLCAiZGF0ZV9vZl9leHBpcnkiOiAibnhKQk5kdHd2YjJUS0tKTkd2RjZfM
Xl3RWRLcm90ajY2Qzg4V1BvbUxmbyJ9fV19LCAiY2xhaW1zIjogeyJnaXZlbl9uYW1lIjo
gInk5dUZQSEFWcU5BWjdQSnlrMS0xeVFKWlpXWnpLR1A1Rkx0OXR4S004NE0iLCAiZmFta
Wx5X25hbWUiOiAiWHlVaWtZOFY4TVdlQmZYVU9wOGdJN0Y3LXlDMjhKcjVJeURndkJ4WHp
kNCIsICJiaXJ0aGRhdGUiOiAiN0dsaWVNTEpoTTc4Q191UVFwOXdVWFNaTGVxQk4xWUdRV
Dg3Qkl1YnlLVSIsICJwbGFjZV9vZl9iaXJ0aCI6IHsiY291bnRyeSI6ICJSTjN4Y25MWVh
fR0RoVndmUHZ0aXN1TFBmaTBkNzR6cWloRmJRcmRfVUcwIiwgImxvY2FsaXR5IjogImlOa
3BXcUo5a0laUXE5NWR6U3lFWmpiUEpzNkZxdTdHRkJLb3VFQzNPeEUifSwgIm5hdGlvbmF
saXRpZXMiOiAiLXRpbllHSzBHWG5rZkFSeGlOSVdxMFZuek5SbC1LdjNLWTNtNWc1RmVtZ
yIsICJhZGRyZXNzIjogIjYzRXpQVjB5dlRwZU9nVjM0eUN3d2VDdk8tMnd4dHMyV3FiamF
fU3V3UFEifX0sICJiaXJ0aF9taWRkbGVfbmFtZSI6ICJ2TTY4STZYbnJWbHl0MUx4Szl4e
GdGeWNzanR3MnZMZEdwTmdrM0U4UVE0IiwgInNhbHV0YXRpb24iOiAiaVRoZkN1MnVsTG9
lNWk2Z0NFcS0tWTZSLWd4SEh0SXVrWGI5cW5makg1ayIsICJtc2lzZG4iOiAieFVwVS1he
kJZZFhlSmlkYzhZdzVNWHRmUHo0XzRrQXJKaGZsWGN4emt6cyJ9LCAiaGFzaF9hbGciOiA
ic2hhLTI1NiJ9.Z1w9pHuA1sZoCUonBBOfA93f9AAriL0NM3g2-0x6hT6Syo3t52BOONED
lNuvJhGMEnemR8Up5nAYzmZfVCQWCy8In729hH_A-Pn1YnZ4m3mYhYw65U0Z9-MlF5guBD
GM6X7gcGf-3O2wnIRquFJ4au3pES6oZbXYpIHFojjbW8YUy_ZFDNUTVOk4uzTrIcJFocsY
zWr0QQe4WB5EIv8k5LToW_gCopEVQbQ3db9g-Cxtfi9GaCGFcHeANXwa8Z_mg4weYXc3Fz
z4DFKFpU2WhjXZ6g0wLS0AHHqE3d8FF7iZyFq9sXI9_N07YnU7D1wkWzpkCwEp-bop0SK8
DKGUXA.ewogICAgIl9zZCI6IHsKICAgICAgICAidmVyaWZpZWRfY2xhaW1zIjogewogICA
gICAgICAgICAidmVyaWZpY2F0aW9uIjogewogICAgICAgICAgICAgInRydXN0X2ZyY
W1ld29yayI6ICJbXCJlbHVWNU9nM2dTTklOEVZbnN4QV9BXCIsIFwiZGVfYW1sXCJdIiw
KICAgICAgICAgICAgICAgICJ0aW1lIjogIltcIjZJajd0TS1hNWlWUEdib1M1dG12VkFcI
iwgXCIyMDEyLTA0LTIzVDE4OjI1WlwiXSIsCiAgICAgICAgICAgICAgICAidmVyaWZpY2F
0aW9uX3Byb2Nlc3MiOiAiW1wiZUk4WldtOVFuS1BwTlBlTmVuSGRoUVwiLCBcImYyNGM2Z
i02ZDNmLTRlYzUtOTczZS1iMGQ4NTA2ZjNiYzdcIl0iLAogICAgICAgICAgICAgICAgImV

2aWRlbmNlIjogWwogICAgICAgICAgICAgICAgICAgIHsKICAgICAgICAgICAgICAgICAgI
CAgICAgInR5cGUiOiAiW1wiUWdfTzY0enFBeGU0MTJhMTA4aXJvQVwiLCBcImRvY3VtZW5
0XCJdIiwKICAgICAgICAgICAgICAgICAgICAgIm1ldGhvZCI6ICJbXCJBSngtMDk1V
lBycFR0TjRRTU9xUk9BXCIsIFwicGlwcFwiXSIsCiAgICAgICAgICAgICAgICAgICAgICA
gICJ0aW1lIjogIltcIlBjMzNKTTJMY2hjVV9sSGdndl91ZlFcIiwgXCIyMDEyLTA0LTIyV
DExOjMwWlwiXSIsCiAgICAgICAgICAgICAgICAgICAgICJkb2N1bWVudCI6IHsKICA
gICAgICAgICAgICAgICAgICAgICAgICJ0eXBlIjogIltcIkcwMk5TclFmakZYUTdJb
zA5c3lhakFcIiwgXCJpZGNhcmRcIl0iLAogICAgICAgICAgICAgICAgICAgICAgICAgICA
gImlzc3VlciI6IHsKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAibmFtZSI6I
CJbXCJsa2x4RjVqTVlsR1RQVW92TU5JdkNBXCIsIFwiU3RhZHQgQXVnc2J1cmdcIl0iLAo
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICJjb3VudHJ5IjogIltcIm5QdW9Rb
mtSRnEzQkllQW03QW5YRkFcIiwgXCJERVwiXSIKICAgICAgICAgICAgICAgICAgICAgICA
gICAgIH0sCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAibnVtYmVyIjogIltcIjViU
HMxSXF1Wk5hMGhrYUZ6enpaTndcIiwgXCI1MzU1NDU1NFwiXSIsCiAgICAgICAgICAgICA
gICAgICAgICAgICAiZGF0ZV9vZl9pc3N1YW5jZSI6ICJbXCI1YTJXMF9OcmxFWnpmc
W1rXzdQcS13XCIsIFwiMjAxMC0wMy0yM1wiXSIsCiAgICAgICAgICAgICAgICAgICAgICA
gICAgICAiZGF0ZV9vZl9leHBpcnkiOiAiW1wieTFzVlU1d2RmSmFoVmRnd1BnUzdSUVwiL
CBcIjIwMjAtMDMtMjJcIl0iCiAgICAgICAgICAgICAgICAgICAgICAgIH0KICAgICAgICA
gICAgICAgICAgICB9CiAgICAgICAgICAgICAgICBdCiAgICAgICAgICAgIH0sCiAgICAgI
CAgICAgICJjbGFpbXMiOiB7CiAgICAgICAgICAgICAgICAiZ2l2ZW5fbmFtZSI6ICJbXCJ
IYlE0WDhzclZXM1FEeG5JSmRxeU9BXCIsIFwiTWF4XCJdIiwKICAgICAgICAgICAgICAgI
CJmYW1pbHlfbmFtZSI6ICJbXCJDOUdTb3VqdmlKcXVFZ1lmb2pDYjFBXCIsIFwiTWVpZXJ
cIl0iLAogICAgICAgICAgICAgICAgImJpcnRoZGF0ZSI6ICJbXCJreDVrRjE3Vi14MEptd
1V4OXZndnR3XCIsIFwiMTk1Ni0wMS0yOFwiXSIsCiAgICAgICAgICAgICAgICAicGxhY2V
fb2ZfYmlydGgiOiB7CiAgICAgICAgICAgICAgICAgICAgImNvdW50cnkiOiAiW1wiSDNvM
XVzd1A3NjBGaTJ5ZUdkVkNFUVwiLCBcIkRFXCJdIiwKICAgICAgICAgICAgICAgICAgICA
ibG9jYWxpdHkiOiAiW1wiT0JLbFRWbHZMZy1BZHdxWUdiUDhaQVwiLCBcIk11c3RlcnN0Y
WR0XCJdIgogICAgICAgICAgICAgICAgfSwKICAgICAgICAgICAgICAgICJuYXRpb25hbGl
0aWVzIjogIltcIk0wSmI1N3Q0MXVicmtTdXlyRFQzeEFcIiwgW1wiREVcIl1dIiwKICAgI
CAgICAgICAgICAgICJhZGRyZXNzIjogIltcIkRzbXRLTmdwVjRkQUhwanJjYW9zQXdcIiw
ge1wibG9jYWxpdHlcIjogXCJNYXhzdGFkdFwiLCBcInBvc3RhbF9jb2RlXCI6IFwiMTIzN
DRcIiwgXCJjb3VudHJ5XCI6IFwiREVcIiwgXCJzdHJlZXRfYWRkcmVzc1wiOiBcIkFuIGR
lciBXZWlkZSAyMlwifV0iCiAgICAgICAgICAgIH0KICAgICAgICB9LAogICAgICAgICJia
XJ0aF9taWRkbGVfbmFtZSI6ICJbXCJlSzVvNXBIZmd1cFBwbHRqMXFoQUp3XCIsIFwiVGl
tb3RoZXVzXCJdIiwKICAgICAgICAic2FsdXRhdGlvbiI6ICJbXCJqN0FEZGIwVVZiMExpM
GNpUGNQMGV3XCIsIFwiRHIuXCJdIiwKICAgICAgICAibXNpc2RuIjogIltcIldweEpyRnV
YOHVTaTJwNGh0MDlqdndcIiwgXCI0OTEyMzQ1Njc4OVwiXSIKICAgIH0KfQ

(Line breaks for presentation only.)

A SD-JWT-R for some of the claims:

```
{
  "nonce": "2GLC42sKQveCfGfryNRN9w",
  "aud": "https://example.com/verifier",
  "sd_release": {
    "verified_claims": {
      "verification": {
        "trust_framework": "[\"eluV5Og3gSNII8EYnsxA_A\", \"de_aml\"]",
        "time": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"2012-04-23T18:25Z\"]",
        "evidence": [
          {
            "type": "[\"Qg_O64zqAxe412a108iroA\", \"document\"]"
          }
        ]
      },
      "claims": {
        "given_name": "[\"HbQ4X8srVW3QDxnIJdqyOA\", \"Max\"]",
        "family_name": "[\"C9GSoujviJquEgYfojCb1A\", \"Meier\"]",
        "birthdate": "[\"kx5kF17V-x0JmwUx9vgvtw\", \"1956-01-28\"]",
        "place_of_birth": {
          "country": "[\"H3o1uswP760Fi2yeGdVCEQ\", \"DE\"]"
        }
      }
    }
  }
}
```

A.3. Example 4 - W3C Verifiable Credentials Data Model

This example illustrates how the artifacts defined in this specification can be represented using W3C Verifiable Credentials Data Model as defined in [VC_DATA].

Below is a non-normative example of an SD-JWT represented as a verifiable credential encoded as JSON and signed as JWS compliant to [VC_DATA].

SVC sent alongside this SD-JWT as a JWT-VC is same as in Example 1.

```
{
  "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"
    ]
  },
  "sd_digests": {
    "given_name": "fUMdn88aaoyKTHrvZd6AuLmPraGhPJ0zF5r_JhxCVZs",
    "family_name": "9h5vgv6TpFV6GmnPtugiMLl5tHetHeb5X_2cKHjN7cw",
    "birthdate": "fvLCnDm3r4VSYcBF3pIlXP4ulEoHuHOfG_YmFZEuxpQ"
  }
}
```

Below is a non-normative example of an SD-JWT-R represented as a verifiable presentation encoded as JSON and signed as a JWS compliant to [VC_DATA].

```
{
  "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\"]",
    "birthdate": "[\"5bPs1IquZNa0hkaFzzzZNw\", \"1940-01-01\"]"
  }
}
```

Appendix B. Document History

[[To be removed from the final specification]]

-01

*Editorial fixes

*Added hash_alg claim

*Renamed _sd to sd_digests and sd_release

*Added descriptions on holder binding - more work to do

*Clarify that signing the SD-JWT is mandatory

-00

*Renamed to SD-JWT (focus on JWT instead of JWS since signature is optional)

*Make holder binding optional

*Rename proof to release, since when there is no signature, the term "proof" can be misleading

*Improved the structure of the description

*Described verification steps

*All examples generated from python demo implementation

*Examples for structured objects

Authors' Addresses

Daniel Fett
yes.com

Email: mail@danielfett.de
URI: https://danielfett.de/

Kristina Yasuda
Microsoft

Email: Kristina.Yasuda@microsoft.com