Workgroup: Web Authorization Protocol
Internet-Draft:
draft-ietf-oauth-selective-disclosure-jwt-02
Published: 7 December 2022
Intended Status: Standards Track
Expires: 10 June 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 10 June 2023.

Copyright Notice

Table of Contents

1. Introduction

The JSON-based representation of claims in a signed JSON Web Token (JWT) [RFC7519] is secured against modification using JSON Web Signature (JWS) [RFC7515] digital signatures. A consumer of a signed JWT that has checked the signature can safely assume that the contents of the token 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.

One of the common use cases of a signed JWT is representing a user's identity. As long as the signed JWT is one-time use, it typically only contains those claims the user has consented to disclose to a specific Verifier. However, there is an increasing number of use cases where a signed JWT is created once and then used a number of times by the user (the "Holder" of the JWT). In such cases, the signed JWT needs to contain the superset of all claims the user of the signed JWT might want to disclose 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.

In an SD-JWT, claims can be hidden, but cryptographically protected against undetected modification. When issuing the SD-JWT to the Holder, the Issuer also sends the cleartext counterparts of all

hidden claims, the so-called Disclosures, separate from the SD-JWT itself.

The Holder decides which claims to disclose to a Verifier and forwards the respective Disclosures together with the SD-JWT to the Verifier. The Verifier has to verify that all disclosed claim values were part of the original, Issuer-signed SD-JWT. The Verifier will not, however, learn any claim values not disclosed in the Disclosures.

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 an optional mechanism for Holder Binding, or the concept of binding an SD-JWT to key material controlled by the Holder. The strength of the Holder Binding is conditional upon the trust in the protection of the private key of the key pair an SD-JWT is bound to.

This specification aims to be easy to implement and to leverage established and widely used data formats and cryptographic algorithms wherever possible.

1.1. Feature Summary

* This specification defines
  - a format enabling selective disclosure for JWTs,
  - formats for associated data that enables disclosing claims, and
  - formats for the combined transport of SD-JWTs and the associated data.
* The specification supports selectively disclosable claims in flat data structures as well as more complex, nested data structures.
* This specification enables combining selectively disclosable claims with clear-text claims that are always disclosed.
* For selectively disclosable claims, claim names are always blinded.

1.2. 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: Process of a Holder disclosing to a Verifier a subset of claims contained in a claim set issued by an Issuer.

Selectively Disclosable JWT (SD-JWT): An Issuer-created signed JWT (JWS, [RFC7515]) that supports selective disclosure as defined in this document and can contain both regular claims and digests of selectively-disclosable claims.

Disclosure: A combination of a salt, a cleartext claim name, and a cleartext claim value, all of which are used to calculate a digest for the respective claim.

Cryptographic Holder Binding: Ability of the Holder to prove legitimate possession of an SD-JWT by proving control over the same private key during the issuance and presentation. An SD-JWT with Holder Binding 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.

Holder: An entity that received SD-JWTs from the Issuer and has control over them.

Verifier: An entity that requests, checks and extracts the claims from an SD-JWT and respective Disclosures.

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

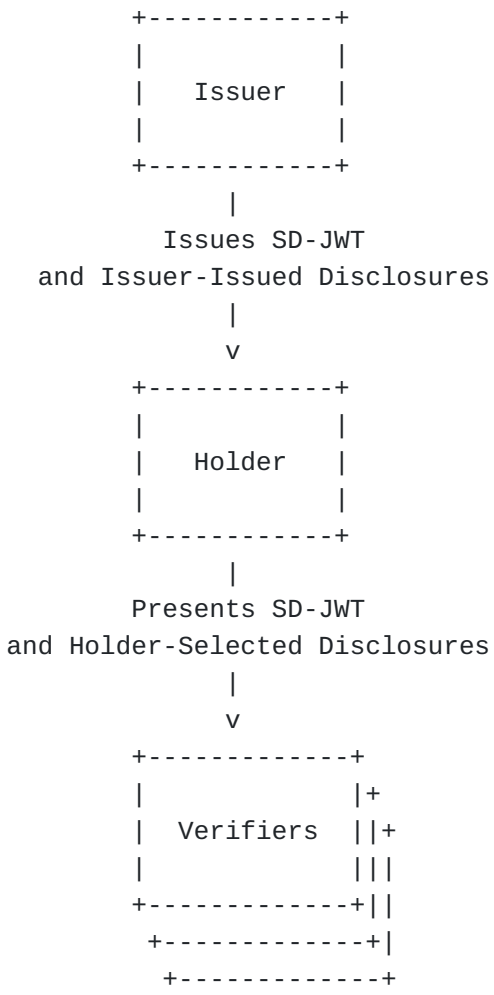

Figure 1: SD-JWT Issuance and Presentation Flow

4. Concepts

In the following, the contents of SD-JWTs and Disclosures 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 digests over the claims (per claim: a random salt, the claim name and the claim value). It MAY further contain clear-text claims that are always disclosed to the Verifier. It MUST be digitally signed using the Issuer's private key.

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

SD-CLAIMS is an array of digest values that ensure the integrity of and map to the respective Disclosures. Digest values are calculated

over the Disclosures, each of which contains the claim name (CLAIM-NAME), the claim value (CLAIM-VALUE), and a random salt (SALT). Digests are calculated using a hash function:

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

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

The Issuer further creates a set of Disclosures for all claims in the SD-JWT. The Disclosures are sent to the Holder together with the SD-JWT:

```
DISCLOSURES = (
    (SALT, CLAIM-NAME, CLAIM-VALUE)
)*
```

The SD-JWT and the Disclosures are sent to the Holder by the Issuer:

```
COMBINED-ISSUANCE = SD-JWT | DISCLOSURES
```

4.2. Creating Holder-Selected Disclosures

To disclose to a Verifier a subset of the SD-JWT claim values, a Holder selects a subset of the Disclosures and sends it to the Verifier along with the SD-JWT.

```
HOLDER-SELECTED-DISCLOSURES = (
    (SALT, CLAIM-NAME, CLAIM-VALUE)
)*

COMBINED-PRESENTATION = SD-JWT | HOLDER-SELECTED-DISCLOSURES
```

4.3. Optional Holder Binding

Some use-cases may require Holder Binding.

Cryptographic Holder Binding is an optional feature, but when it is desired, SD-JWT must contain information about key material controlled by the Holder:

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

The Holder can then create a signed document HOLDER-BINDING-JWT using its private key. This document contains some data provided by

the Verifier (out of scope of this document) to ensure the freshness of the signature, for example, a nonce and an indicator of the intended audience for the document.

```
HOLDER-BINDING-JWT-DOC = (NONCE, AUDIENCE)
HOLDER-BINDING-JWT = HOLDER-BINDING-JWT-DOC |
    SIG(HOLDER-BINDING-JWT-DOC, HOLDER-PRIV-KEY)
```

The Holder Binding JWT is sent to the Verifier along with the SD-JWT and the Holder-Selected Disclosures.

```
COMBINED-PRESENTATION = SD-JWT | HOLDER-SELECTED-DISCLOSURES | HOLDER-BI
```

Note that there may be other ways to send the Holder Binding JWT to the Verifier or to prove Holder Binding. In these cases, inclusion of the Holder Binding JWT in the COMBINED-PRESENTATION is not required.

4.4. Verifying Holder-Selected Disclosures

At a high level, the Verifier

* receives the COMBINED-PRESENTATION from the Holder and verifies the signature of the SD-JWT using the Issuer's public key,
* verifies the Holder Binding JWT, if Holder Binding is required by the Verifier's policy, using the public key included in the SD-JWT,
* calculates the digests over the Holder-Selected Disclosures and verifies that each digest is contained in the SD-JWT.

The detailed algorithm is described in Section 6.2.

5. Data Formats

This section defines data formats for SD-JWTs, Disclosures, Holder Binding JWTs and formats for combining these elements for transport.

5.1. The Challenge of Canonicalization

When receiving an SD-JWT with associated Disclosures, a Verifier must be able to re-compute digests of the disclosed claim values and, given the same input values, obtain the same digest values as signed by the Issuer.

Usually, JSON-based formats transport claim values as simple properties of a JSON object such as this:

```
...
  "family_name": "Möbius",
  "address": {
    "street_address": "Schulstr. 12",
    "locality": "Schulpforta"
  }
...
```

However, a problem arises when computation over the data need to be performed and verified, like signing or computing digests. Common signature schemes require the same byte string as input to the signature verification as was used for creating the signature. In the digest approach outlined above, the same problem exists: for the Issuer and the Verifier to arrive at the same digest, the same byte string must be hashed.

JSON [RFC7159], however, does not prescribe a unique encoding for data, but allows for variations in the encoded string. The data above, for example, can be encoded as

```
...
"family_name": "M\u00f6bius",
"address": {
  "street_address": "Schulstr. 12",
  "locality": "Schulpforta"
}
...
```

or as

```
...
"family_name": "Möbius",
"address": {"locality":"Schulpforta", "street_address":"Schulstr. 12"}
...
```

The two representations "M\u00f6bius" and "Möbius" are very different on the byte-level, but yield equivalent objects. Same for the representations of address, varying in white space and order of elements in the object.

The variations in white space, ordering of object properties, and encoding of Unicode characters are all allowed by the JSON specification, including further variations, e.g., concerning floating-point numbers, as described in [RFC8785]. Variations can be introduced whenever JSON data is serialized or deserialized and unless dealt with, will lead to different digests and the inability to verify signatures.

There are generally two approaches to deal with this problem:

1. Canonicalization: The data is transferred in JSON format, potentially introducing variations in its representation, but is transformed into a canonical form before computing a digest. Both the Issuer and the Verifier must use the same canonicalization algorithm to arrive at the same byte string for computing a digest.

2. Source string hardening: Instead of transferring data in a format that may introduce variations, a representation of the data is serialized. This representation is then used as the hashing input at the Verifier, but also transferred to the Verifier and used for the same digest calculcation there. This means that the Verifier can easily compute and check the digest of the byte string before finally deserializing and accessing the data.

Mixed approaches are conceivable, i.e., transferring both the original JSON data plus a string suitable for computing a digest, but such approaches can easily lead to undetected inconsistencies resulting in time-of-check-time-of-use type security vulnerabilities.

In this specification, the source string hardening approach is used, as it allows for simple and reliable interoperability without the requirement for a canonicalization library. To harden the source string, any serialization format that supports the necessary data types could be used in theory, like protobuf, msgpack, or pickle. In this specification, JSON is used and plain text values of each Disclosure are encoded using base64url-encoding for transport. This approach means that SD-JWTs can be implemented purely based on widely available JWT, JSON, and Base64 encoding and decoding libraries.

A Verifier can then easily check the digest over the source string before extracting the original JSON data. Variations in the encoding of the source string are implicitly tolerated by the Verifier, as the digest is computed over a predefined byte string and not over a JSON object.

It is important to note that the Disclosures are neither intended nor suitable for direct consumption by an application that needs to access the disclosed claim values after the verification by the Verifier. The Disclosures are only intended to be used by a Verifier to check the digests over the source strings and to extract the original JSON data. The original JSON data is then used by the application. See Section 6.2 for details.

5.2. 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_alg claim described in the following, MAY contain one or more selectively disclosable claims, 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.

5.2.1. Selectively Disclosable Claims

For each claim that is to be selectively disclosed, the Issuer creates a Disclosure, hashes it, and includes the hash instead of the original claim in the SD-JWT, as described next. The Disclosures are then sent to the Holder.

5.2.1.1. Creating Disclosures

The Issuer MUST create a Disclosure for each selectively disclosable claim as follows:

* Create an array of three elements in this order:

    1. A salt value. See Section 8.4 and Section 8.5 for security considerations. The salt value MUST be unique for each claim that is to be selectively disclosed. It is RECOMMENDED to base64url-encode the salt value, producing a string. Any other type that is allowed in JSON MAY be used, e.g., a number.

    2. The claim name, or key, as it would be used in a regular JWT body. This MUST be a string.

    3. The claim's value, as it would be used in a regular JWT body. The value MAY be of any type that is allowed in JSON, including numbers, strings, booleans, arrays, and objects.

* JSON-encode the array such that an UTF-8 string is produced.

* base64url-encode the byte representation of the UTF-8 string, producing a US-ASCII [RFC0020] string. This string is the Disclosure.

The order is decided based on the readability considerations: salts would have a constant length within the SD-JWT, claim names would be around the same length all the time, and claim values would vary in size, potentially being large objects.

The following example illustrates the steps described above.

The array is created as follows:

```
["_26bc4LT-ac6q2KI6cBW5es", "family_name", "Möbius"]
```

The resulting Disclosure would be:
WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsICJmYW1pbHlfbmFtZSIsICJNw7ZiaXVzIl0

Note that the JSON encoding of the object is not canonicalized, so variations in white space, encoding of Unicode characters, and ordering of object properties are allowed. For example, the following strings are all valid and encode the same claim value:

- A different way to encode the umlaut (two dots ¨ placed over the letter):
  WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsICJmYW1pbHlfbmFtZSIsICJNXHUwMGY2Yml1cyJd
- No white space:
  WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImZhbWlseV9uYW1lIiwiTcO2Yml1cyJd
- Newline characters between elements:
  WwoiXzI2YmM0TFQtYWM2cTJLSTZjQlc1ZXMiLAoiZmFtaWx5X25hbWUiLAoiTcO2Yml1cyIKXQ

5.2.1.2. Hashing Disclosures

For embedding the Disclosures in the SD-JWT, the Disclosures are hashed using the hash algorithm specified in the _sd_alg claim described below. The resulting digest is then included in the SD-JWT instead of the original claim value, as described next.

The digest MUST be taken over the US-ASCII bytes of the base64url-encoded Disclosure. This follows the convention in JWS [RFC7515] and JWE [RFC7516]. The bytes of the digest MUST then be base64url-encoded.

It is important to note that:

- The input to the hash function is the base64url-encoded Disclosure, not the bytes encoded by the base64url string.
- The bytes of the output of the hash function are base64url-encoded, not the bytes making up the (often used) hex representation of the bytes of the digest.

For example, the SHA-256 digest of the Disclosure WyI2cU1RdlJMNWhhaiIsICJmYW1pbHlfbmFtZSIsICJNw7ZiaXVzIl0 would be uutlBuYeMDyjLLTpf6Jxi7yNkEF35jdyWMn9U7b_RYY.

5.2.1.3. Decoy Digests

An Issuer MAY add additional digests to the SD-JWT that are not associated with any claim. The purpose of such "decoy" digests is to make it more difficult for an attacker to see the original number of claims contained in the SD-JWT. It is RECOMMENDED to create the decoy digests by hashing over a cryptographically secure random number. The bytes of the digest MUST then be base64url-encoded as above. The same digest function as for the Disclosures MUST be used.

For decoy digests, no Disclosure is sent to the Holder, i.e., the Holder will see digests that do not correspond to any Disclosure. See Section 9.2 for additional privacy considerations.

To ensure readability and replicability, the examples in this specification do not contain decoy digests unless explicitly stated.

5.2.1.4. Creating an SD-JWT

An SD-JWT is a JWT that MUST be signed using the Issuer's private key.

An SD-JWT MAY contain both selectively disclosable claims and non-selectively disclosable claims, i.e., claims that are always contained in the SD-JWT in plaintext and are always visible to a Verifier.

It is the Issuer who decides which claims are selectively disclosable and which are not. However, claims controlling the validity of the SD-JWT, such as iss, exp, or nbf are usually included in plaintext. End-User claims MAY be included as plaintext as well, e.g., if hiding the particular claims from the Verifier does not make sense in the intended use case.

Claims that are not selectively disclosable are included in the SD-JWT in plaintext just as they would be in any other JWT.

Selectively disclosable claims are omitted from the SD-JWT. Instead, the digests of the respective Disclosures and potentially decoy digests are contained as an array in a new JWT claim, _sd.

The _sd claim MUST be an array of strings, each string being a digest of a Disclosure or a decoy digest as described above.

The array MAY be empty in case the Issuer decided not to selectively disclose any of the claims at that level. However, it is RECOMMENDED to omit _sd claim in this case to save space.

The Issuer MUST hide the original order of the claims in the array. To ensure this, it is RECOMMENDED to shuffle the array of hashes,

e.g., by sorting it alphanumerically or randomly. The precise method does not matter as long as it does not depend on the original order of elements.

Issuers MUST NOT issue SD-JWTs where

- the key _sd is already used for the purpose other than to contain the array of digests, or
- the claim value contained in a Disclosure contains (at the top level or nested deeper) an object with an _sd key, or
- the same Disclosure value appears more than once (in the same array or in different arrays).

5.2.1.5. Nested Data in SD-JWTs

Just like any JWT, an SD-JWT MAY contain key value pairs where the value is an object. For any object in an SD-JWT, the Issuer MAY decide to either make the entire object selectively disclosable or to make its properties selectively disclosable individually. In the latter case, the Issuer MAY even choose to make some of the object's properties selectively disclosable and others not.

In any case, the _sd claim MUST be included in the SD-JWT at the same level as the original claim and therefore MAY appear multiple times in an SD-JWT.

The following examples show some of the options an Issuer has when producing an SD-JWT with the following End-User data.

```
{
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "address": {
    "street_address": "Schulstr. 12",
    "locality": "Schulpforta",
    "region": "Sachsen-Anhalt",
    "country": "DE"
  }
}
```

Important: Throughout the examples in this document, line breaks had to be added to JSON strings and base64-encoded strings (as shown in the next example) to adhere to the 72 character limit for lines in RFCs and for readability. JSON does not allow line breaks in strings.

5.2.1.5.1. 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": [
    "vNhbLbXx6SFqIJZn5lNXyrASjpaXGUT2uNGuy_gSMbw"
  ],
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1516247022,
  "_sd_alg": "sha-256"
}
```

The Issuer would create the following Disclosure:

Disclosure for address:

```
WyJEa1daaVFLdW1SYXIzbHBmNHFFYlNBIiwgImFkZHJlc3MiLCB7InN0cmVldF9hZGRy
ZXNzIjogIlNjaHVsc3RyLiAxMiIsICJsb2NhbGl0eSI6ICJTY2h1bHBmb3J0YSIsICJy
ZWdpb24iOiAiU2FjaHNlbi1BbmhhbHQiLCAiY291bnRyeSI6ICJERSJ9XQ
```

Contents:

```
["DkWZiQKumRar3lpf4qEbSA", "address", {"street_address": "Schulstr.
12", "locality": "Schulpforta", "region": "Sachsen-Anhalt",
"country": "DE"}]
```

SHA-256 Hash: vNhbLbXx6SFqIJZn5lNXyrASjpaXGUT2uNGuy_gSMbw

5.2.1.5.2. Option 2: Structured SD-JWT

The Issuer may instead decide to make the address claim contents selectively disclosable individually:

```
{
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "address": {
    "_sd": [
      "7pHe1uQ5uSClgAxXdG0E6dKnBgXcxEO1zvoQO9E5Lr4",
      "9-VdSnvRTZNDo-4Bxcp3X-V9VtLOCRUkR6oLWZQl81I",
      "nTzPZ3Q68z1Ko_9ao9LK0mSYXY5gY6UG6KEkQ_BdqU0",
      "pEtkKwoFK_JHN7yNby0Lc_Jc10BAxCm5yXJjDbVehvU"
    ]
  },
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1516247022,
  "_sd_alg": "sha-256"
}
```

In this case, the Issuer would use the following data in the Disclosures for the address sub-claims:

Disclosure for street_address:

```
WyI0d3dqUzlyMm4tblBxdzNpTHR0TkFBIiwgInN0cmVldF9hZGRyZXNzIiwgIlNjaHVs
c3RyLiAxMiJd
```

Contents:

```
["4wwjS9r2n-nPqw3iLttNAA", "street_address", "Schulstr. 12"]
```

SHA-256 Hash: pEtkKwoFK_JHN7yNby0Lc_Jc10BAxCm5yXJjDbVehvU

Disclosure for locality:

```
WyJXcEtIQmVTa3A5U2MyNVV4a1F1RmNRIiwgImxvY2FsaXR5IiwgIlNjaHVscGZvcnRh
Il0
```

Contents:

```
["WpKHBeSkp9Sc25UxkQuFcQ", "locality", "Schulpforta"]
```

SHA-256 Hash: nTzPZ3Q68z1Ko_9ao9LK0mSYXY5gY6UG6KEkQ_BdqU0

Disclosure for region:

```
WyIzSl9xWGctdUwxYzdtN1FoT0hUNTJnIiwgInJlZ2lvbiIsICJTYWNoc2VuLUFuaGFs
dCJd
```

Contents:

```
["3J_qXg-uL1c7m7QhOHT52g", "region", "Sachsen-Anhalt"]
```

SHA-256 Hash: 9-VdSnvRTZNDo-4Bxcp3X-V9VtLOCRUkR6oLWZQl81I

Disclosure for country:

```
WyIwN2U3bWY2YWpTUDJjZkQ3NmJCZE93IiwgImNvdW50cnkiLCAiREUiXQ
```

Contents:

```
["07e7mf6ajSP2cfD76bBdOw", "country", "DE"]
```

SHA-256 Hash: 7pHe1uQ5uSClgAxXdG0E6dKnBgXcxEO1zvoQO9E5Lr4

5.2.1.5.3. Option 3: Structured SD-JWT, only some properties selectively disclosable

The Issuer may also make one sub-claim of address non-selectively disclosable and hide only the other sub-claims:

```
{
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "address": {
    "_sd": [
      "ToD9fSNGo_SOCnTM0r0AaGjdHEIh4doXinCkKjR2fk4",
      "bIPnfvtg9QQSGd7W9srnYOVTK-sNUFz9kr1Js7XaU4E",
      "xWq47lkkG-K5CYfcWtHqwi9CbL9LCP3q8v5YsSlmoUQ"
    ],
    "country": "DE"
  },
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1516247022,
  "_sd_alg": "sha-256"
}
```

In this case, the Issuer would issue the following Disclosures:

Disclosure for street_address:

```
WyJkTWZMOXBrOVFXUUNPajJRVDA0c21BIiwgInN0cmVldF9hZGRyZXNzIiwgIlNjaHVs
c3RyLiAxMiJd
```

Contents:

```
["dMfL9pk9QWQCOj2QT04smA", "street_address", "Schulstr. 12"]
```

SHA-256 Hash: ToD9fSNGo_SOCnTM0r0AaGjdHEIh4doXinCkKjR2fk4

Disclosure for locality:

```
WyJsNklkRC1FeDZ5eHFGck9DUjFNbktBIiwgImxvY2FsaXR5IiwgIlNjaHVscGZvcnRh
Il0
```

Contents:

```
["l6IdD-Ex6yxqFrOCR1MnKA", "locality", "Schulpforta"]
```

SHA-256 Hash: bIPnfvtg9QQSGd7W9srnYOVTK-sNUFz9kr1Js7XaU4E

Disclosure for region:

```
WyI3WFB4R21ldC1vaWFoOEhDdmU3bTJBIiwgInJlZ2lvbiIsICJTYWNoc2VuLUFuaGFs
dCJd
```

Contents:

```
["7XPxGmet-oiah8HCve7m2A", "region", "Sachsen-Anhalt"]
```

SHA-256 Hash: xWq47lkkG-K5CYfcWtHqwi9CbL9LCP3q8v5YsSlmoUQ

5.2.2. Hash Function Claim

The claim _sd_alg indicates the hash algorithm used by the Issuer to generate the digests over the salts and the claim values.

The hash algorithm identifier MUST be a hash algorithm value from the "Hash Name String" column in the IANA "Named Information Hash Algorithm" registry [IANA.Hash.Algorithms].

To promote interoperability, implementations MUST support the SHA-256 hash algorithm.

See Section 8 for requirements regarding entropy of the salt, minimum length of the salt, and choice of a hash algorithm.

5.2.3. 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: Examples in this document use cnf Claim defined in [RFC7800] to include raw public key by value in SD-JWT.

5.3. Example 1: SD-JWT

This example uses the following object as the set of claims that the Issuer is issuing:

```
{
  "sub": "john_doe_42",
  "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 in this case, i.e., all of the claims in the address claim can only be disclosed in full.

```
{
  "_sd": [
    "NYCoSRKEYwXdpe5yduJXCxxhynEU8z-b4TyNiap77UY",
    "SY8n2BbkX9lrY3exHlSwPRFXoD09GF8a9CPO-G8j208",
    "TPsGNPYA46wmBxfv2znOJhfdoN5Y1GkezbpaGZCT1ac",
    "ZkSJxxeGluIdYBb7CqkZbJVm0w2V5UrReNTzAQCYBjw",
    "l9qIJ9JTQwLG7OLEICTFBVxmArw8Pjy65dD6mtQVG5c",
    "o1SAsJ33YMioO9pX5VeAM1lxuHF6hZW2kGdkKKBnVlo",
    "qqvcqnczAMgYx7EykI6wwtspyvyvK790ge7MBbQ-Nus"
  ],
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1516247022,
  "_sd_alg": "sha-256",
  "cnf": {
    "jwk": {
      "kty": "RSA",
      "n": "pm4bOHBg-oYhAyPWzR56AWX3rUIXp11_ICDkGgS6W3ZWLts-hzwI3x65
        659kg4hVo9dbGoCJE3ZGF_eaetE30UhBUEgpGwrDrQiJ9zqprmcFfr3qvvkG
        jtth8Zgl1eM2bJcOwE7PCBHWTKWYs152R7g6Jg2OVph-a8rq-q79MhKG5QoW
        _mTz10QT_6H4c7PjWG1fjh8hpWNnbP_pv6d1zSwZfc5fl6yVRL0DV0V3lGHK
        e2Wqf_eNGjBrBLVklDTk8-stX_MWLcR-EGmXAOv0UBWitS_dXJKJu-vXJyw1
        4nHSGuxTIK2hx1pttMft9CsvqimXKeDTU14qQL1eE7ihcw",
      "e": "AQAB"
    }
  }
}
```


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

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJfc2QiOiBbIk5ZQ29TUktFWXdYZHBlNXlkdUpYQ
3h4aHluRVU4ei1iNFR5TmlhcDc3VVkiLCAiU1k4bjJCYmtYOWxyWTNleEhsU3dQUkZYb
0QwOUdGOGE5Q1BPLUc4ajIwOCIsICJUUHNHTlBZQTQ2d21CeGZ2MnpuT0poZmRvTjVZM
UdrZXpicGFHWkNUMWFjIiwgIlprU0p4eGVHbHVJZFlCYjdDcWtaYkpWbTB3MlY1VXJSZ
U5UekFRQ1lCanciLCAibDlxSUo5SlRRd0xHN09MRUlDVEZCVnhtQXJ3OFBqeTY1ZEQ2b
XRRVkc1YyIsICJvMVNBc0ozM1lNaW9POXBYNVZlQU0xbHh1SEY2aFpXMmtHZGtLS0JuV
mxvIiwgInFxdmNxbmN6QU1nWXg3RXlrSTZ3d3RzcHl2eXZLNzkwZ2U3TUJiUS1OdXMiX
SwgImlzcyI6ICJodHRwczovL2V4YW1wbGUuY29tL2lzc3VlciIsICJpYXQiOiAxNTE2M
jM5MDIyLCAiZXhwIjogMTUxNjI0NzAyMiwgIl9zZF9hbGciOiAic2hhLTI1NiIsICJjb
mYiOiB7Imp3ayI6IHsia3R5IjogIlJTQSIsICJuIjogInBtNGJPSEJnLW9ZaEF5UFd6U
jU2QVdYM3JVSVhwMTFfSUNEa0dnUzZXM1pXTHRzLWh6d0kzeDY1NjU5a2c0aFZvOWRiR
29DSkUzWkdGX2VhZXRFMzBVaEJVRWdwR3dyRHJRaUo5enFwcm1jRmZyM3F2dmtHanR0a
DhaZ2wxZU0yYkpjT3dFN1BDQkhXVEtXWXMxNTJSN2c2SmcyT1ZwaC1hOHJxLXE3OU1oS
0c1UW9XX21UejEwUVRfNkg0YzdQaldHMWZqaDhocFdObmJQX3B2NmQxelN3WmZjNWZsN
nlWUkwwRFYwVjNsR0hLZTJXcWZfZU5HakJyQkxWa2xEVGs4LXN0WF9NV0xjUi1FR21YQ
U92MFVCV2l0U19kWEpLSnUtdlhKeXcxNG5IU0d1eFRJSzJoeDFwdHRNZnQ5Q3N2cWltW
EtlRFRVMTRxUUwxZUU3aWhjdyIsICJlIjogIkFRQUIifX19.xqgKrDO6dK_oBL3fiqdc
q_elaIGxM6Z-RyuysglGyddR1O1IiE3mIk8kCpoqcRLR88opkVWN2392K_XYfAuAmeT9
kJVisD8ZcgNcv-MQlWW9s8WaViXxBRe7EZWkWRQcQVR6jf95XZ5H2-_KA54POq3L42xj
k0y5vDr8yc08Reak6vvJVvjXpp-Wk6uxsdEEAKFspt_EYIvISFJhfTuQqyhCjnaW13X3
12MSQBPwjbHn74ylUqVLljDvqcemxeqjh42KWJq4C3RqNJ7anA2i3FU1kB4-KNZWsijY
7-op49iL7BrnIBxdlAMrbHEkoGTbFWdl7Ki17GHtDxxa1jaxQg
```

The Issuer creates the following Disclosures:

Disclosure for sub:

```
WyJkcVR2WE14UzBHYTNEb2FHbmU5eDBRIiwgInN1YiIsICJqb2huX2RvZV80MiJd
```

Contents:

```
["dqTvXMxS0Ga3DoaGne9x0Q", "sub", "john_doe_42"]
```

SHA-256 Hash: ZkSJxxeGluIdYBb7CqkZbJVm0w2V5UrReNTzAQCYBjw

Disclosure for given_name:

```
WyIzanFjYjY3ejl3a3MwOHp3aUs3RXlRIiwgImdpdmVuX25hbWUiLCAiSm9obiJd
```

Contents:

```
["3jqcb67z9wks08zwiK7EyQ", "given_name", "John"]
```

SHA-256 Hash: qqvcqnczAMgYx7EykI6wwtspyvyvK790ge7MBbQ-Nus

Disclosure for family_name:

```
WyJxUVdtakpsMXMxUjRscWhFTkxScnJ3IiwgImZhbWlseV9uYW1lIiwgIkRvZSJd
```

Contents:

["qQWmjJl1s1R4lqhENLRrrw", "family_name", "Doe"]

SHA-256 Hash: l9qIJ9JTQwLG7OLEICTFBVxmArw8Pjy65dD6mtQVG5c

Disclosure for email:

WyJLVXhTNWhFX1hiVmFjckdBYzdFRnd3IiwgImVtYWlsIiwgImpvaG5kb2VAZXhhbXBs
ZS5jb20iXQ

Contents:

["KUxS5hE_XbVacrGAc7EFww", "email", "johndoe@example.com"]

SHA-256 Hash: o1SAsJ33YMioO9pX5VeAM1lxuHF6hZW2kGdkKKBnVlo

Disclosure for phone_number:

WyIzcXZWSjFCQURwSERTUzkzOVEtUml3IiwgInBob25lX251bWJlciIsICIrMS0yMDIt
NTU1LTAxMDEiXQ

Contents:

["3qvVJ1BADpHDSS939Q-Riw", "phone_number", "+1-202-555-0101"]

SHA-256 Hash: SY8n2BbkX9lrY3exHlSwPRFXoD09GF8a9CPO-G8j208

Disclosure for address:

WyIweEd6bjNNaXFzY3RaSV9PcERsQWJRIiwgImFkZHJlc3MiLCB7InN0cmVldF9hZGRy
ZXNzIjogIjEyMyBNYWluIFN0IiwgImxvY2FsaXR5IjogIkFueXRvd24iLCAicmVnaW9u
IjogIkFueXN0YXRlIiwgImNvdW50cnkiOiAiVVMifV0

Contents:

["0xGzn3MiqsctZI_OpDlAbQ", "address", {"street_address": "123 Main
St", "locality": "Anytown", "region": "Anystate", "country": "US"}]

SHA-256 Hash: TPsGNPYA46wmBxfv2znOJhfdoN5Y1GkezbpaGZCT1ac

Disclosure for birthdate:

WyJFUktNMENOZUZKa2FENW1UWFZfWDh3IiwgImJpcnRoZGF0ZSIsICIxOTQwLTAxLTAx
Il0

Contents:

["ERKM0CNeFJkaD5mTXV_X8w", "birthdate", "1940-01-01"]

SHA-256 Hash: NYCoSRKEYwXdpe5yduJXCxxhynEU8z-b4TyNiap77UY

5.4. Combined Format for Issuance

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 digest calculation and the salts. To this end, the Issuer sends the Disclosure objects that were also used for the hash calculation, as described in Section 5.2.1.1, to the Holder.

The data format for sending the SD-JWT and the Disclosures to the Holder is as follows:

```
<SD-JWT>~<Disclosure 1>~<Disclosure 2>~...~<Disclosure N>
```

This is called the Combined Format for Issuance.

The Disclosures and SD-JWT are implicitly linked through the digest values of the Disclosures included in the SD-JWT.

5.4.1. Example

For Example 1, the Combined Format for Issuance looks as follows:

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJfc2QiOiBbIk5ZQ29TUktFWXdYZHBlNXlkdUpYQ
3h4aHluRVU4ei1iNFR5TmlhcDc3VVkiLCAiU1k4bjJCYmtYOWxyWTNleEhsU3dQUkZYb
0QwOUdGOGE5Q1BPLUc4ajIwOCIsICJUUHNHTlBZQTQ2d21CeGZ2MnpuT0poZmRvTjVZM
UdrZXpicGFHWkNUMWFjIiwgIlprU0p4eGVHbHVJZFlCYjdDcWtaYkpWbTB3MlY1VXJSZ
U5UekFRQ1lCanciLCAibDlxSUo5SlRRd0xHN09MRUlDVEZCVnhtQXJ3OFBqeTY1ZEQ2b
XRRVkc1YyIsICJvMVNBc0ozM1lNaW9POXBYNVZlQU0xbHh1SEY2aFpXMmtHZGtLS0JuV
mxvIiwgInFxdmNxbmN6QU1nWXg3RXlrSTZ3d3RzcHl2eXZLNzkwZ2U3TUJiUS1OdXMiX
SwgImlzcyI6ICJodHRwczovL2V4YW1wbGUuY29tL2lzc3VlciIsICJpYXQiOiAxNTE2M
jM5MDIyLCAiZXhwIjogMTUxNjI0NzAyMiwgIl9zZF9hbGciOiAic2hhLTI1NiIsICJjb
mYiOiB7Imp3ayI6IHsia3R5IjogIlJTQSIsICJuIjogInBtNGJPSEJnLW9ZaEF5UFd6U
jU2QVdYM3JVSVhwMTFfSUNEa0dnUzZXM1pXTHRzLWh6d0kzeDY1NjU5a2c0aFZvOWRiR
29DSkUzWkdGX2VhZXRFMzBVaEJVRWdwR3dyRHJRaUo5enFwcm1jRmZyM3F2dmtHanR0a
DhaZ2wxZU0yYkpjT3dFN1BDQkhXVEtXWXMxNTJSN2c2SmcyT1ZwaC1hOHJxLXE3OU1oS
0c1UW9XX21UejEwUVRfNkg0YzdQaldHMWZqaDhocFdObmJQX3B2NmQxelN3WmZjNWZsN
nlWUkwwRFYwVjNsR0hLZTJXcWZfZU5HakJyQkxWa2xEVGs4LXN0WF9NV0xjUi1FR21YQ
U92MFVCV2l0U19kWEpLSnUtdlhKeXcxNG5IU0d1eFRJSzJoeDFwdHRNZnQ5Q3N2cWltW
EtlRFRVMTRxUUwxZUU3aWhjdyIsICJlIjogIkFRQUIifX19.xqgKrDO6dK_oBL3fiqdc
q_elaIGxM6Z-RyuysglGyddR1O1IiE3mIk8kCpoqcRLR88opkVWN2392K_XYfAuAmeT9
kJVisD8ZcgNcv-MQlWW9s8WaViXxBRe7EZWkWRQcQVR6jf95XZ5H2-_KA54POq3L42xj
k0y5vDr8yc08Reak6vvJVvjXpp-Wk6uxsdEEAKFspt_EYIvISFJhfTuQqyhCjnaW13X3
12MSQBPwjbHn74ylUqVLljDvqcemxeqjh42KWJq4C3RqNJ7anA2i3FU1kB4-KNZWsijY
7-op49iL7BrnIBxdlAMrbHEkoGTbFWdl7Ki17GHtDxxa1jaxQg~WyJkcVR2WE14UzBHY
TNEb2FHbmU5eDBRIiwgInN1YiIsICJqb2huX2RvZV80MiJd~WyIzanFjYjY3ejl3a3Mw
OHp3aUs3RXlRIiwgImdpdmVuX25hbWUiLCAiSm9obiJd~WyJxUVdtakpsMXMxUjRscWh
FTkxScnJ3IiwgImZhbWlseV9uYW1lIiwgIkRvZSJd~WyJLVXhTNWhFX1hiVmFjckdBYz
dFRnd3IiwgImVtYWlsIiwgImpvaG5kb2VAZXhhbXBsZS5jb20iXQ~WyIzcXZWSjFCQUR
wSERTUzkzOVEtUml3IiwgInBob25lX251bWJlciIsICIrMS0yMDItNTU1LTAxMDEiXQ~
WyIweEd6bjNNaXFzY3RaSV9PcERsQWJRIiwgImFkZHJlc3MiLCB7InN0cmVldF9hZGRy
ZXNzIjogIjEyMyBNYWluIFN0IiwgImxvY2FsaXR5IjogIkFueXRvd24iLCAicmVnaW9u
IjogIkFueXN0YXRlIiwgImNvdW50cnkiOiAiVVMifV0~WyJFUktNMENOZUZKa2FENW1U
WFZfWDh3IiwgImJpcnRoZGF0ZSIsICIxOTQwLTAxLTAxIl0
```


(Line breaks for presentation only.)

5.5. Combined Format for Presentation

For presentation to a Verifier, the Holder sends the SD-JWT and a selected subset of the Disclosures to the Verifier.

The data format for sending the SD-JWT and the Disclosures to the Verifier is as follows (line break added for readability):

```
<SD-JWT>~<Disclosure 1>~<Disclosure 2>~...~<Disclosure M>~<optional Hold
```

This is called the Combined Format for Presentation.

The Holder MAY send any subset of the Disclosures to the Verifier, i.e., none, multiple, or all Disclosures.

A Holder MUST NOT send a Disclosure that was not included in the SD-JWT or send a Disclosure more than once.

5.5.1. Enabling Holder Binding

The Holder MAY add an optional JWT to prove Holder Binding to the Verifier. The precise contents of the JWT are out of scope of this specification. Usually, a nonce and aud claim are included to show that the proof is intended for the Verifier and to prevent replay attacks. How the nonce or other claims are obtained by the Holder is out of scope of this specification.

Example Holder Binding JWT payload:

```
{
  "nonce": "XZOUco1u_gEPknxS78sWWg",
  "aud": "https://example.com/verifier",
  "iat": 1670366430
}
```

Which is then signed by the Holder to create a JWT like the following:

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogIkxkeVRYd0F5ZnJpcjRfVjZORzFSYzEwVThK
ZExZVHJFQktKaF9oNWlfclUifQ.eyJub25jZSI6ICJYWk9VY28xdV9nRVBrbnhTNzhzV
1dnIiwgImF1ZCI6ICJodHRwczovL2V4YW1wbGUuY29tL3ZlcmlmaWVyIiwgImlhdCI6I
DE2NzAzNjY0MzB9.aJ1HRh21dXP2_pMCHmkqNKaDN25aKuJGiTu2XKOegQ2xjZ_OzLu5
_6gNy0qMhJIXb28-uQc7pMpWMZqkQLV3d7HGBiqiA71RsYYUCF-4VYrFNFb63HQxOHTc
u-nXALwbJgBSbQXKjdpjOj6NrIwItQNh7_EQ3vrf7E0_exmkNoZxy0QCGqApAwUtiLtw
QDExM9UYgms_ayPDZ3TJVcWHd9UtQTJugF4fTCJnq6TQLKuUioK-kLAbDuRamnOpQ5d6
Rbz3NHWdVp3nHM6bgh_3Y0zBFdg6VsmAXvnoIN9b2jzSemNNAniAsalmd4X5pwzyK1TU
HFJm-ib_kFf2G7Iapw
```

Whether to require Holder Binding is up to the Verifier's policy, based on the set of trust requirements such as trust frameworks it belongs to.

Other ways of proving Holder Binding MAY be used when supported by the Verifier, e.g., when the Combined Format for Presentation is itself embedded in a signed JWT. See Section 7 for details.

If no Holder Binding JWT is included, the Combined Format for Presentation ends with the ~ character after the last Disclosure.

5.5.2. Example

The following is a non-normative example of the contents of a Presentation for Example 1, disclosing the claims given_name, family_name, and address, as it would be sent from the Holder to the

Verifier. The Holder Binding JWT as shown before is included as the last element.

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJfc2QiOiBbIk5ZQ29TUktFWXdYZHBlNXlkdUpYQ
3h4aHluRVU4ei1iNFR5TmlhcDc3VVkiLCAiU1k4bjJCYmtYOWxyWTNleEhsU3dQUkZYb
0QwOUdGOGE5Q1BPLUc4ajIwOCIsICJUUHNHTlBZQTQ2d21CeGZ2MnpuT0poZmRvTjVZM
UdrZXpicGFHWkNUMWFjIiwgIlprU0p4eGVHbHVJZFlCYjdDcWtaYkpWbTB3MlY1VXJSZ
U5UekFRQ1lCanciLCAibDlxSUo5SlRRd0xHN09MRUlDVEZCVnhtQXJ3OFBqeTY1ZEQ2b
XRRVkc1YyIsICJvMVNBc0ozM1lNaW9POXBYNVZlQU0xbHh1SEY2aFpXMmtHZGtLS0JuV
mxvIiwgInFxdmNxbmN6QU1nWXg3RXlrSTZ3d3RzcHl2eXZLNzkwZ2U3TUJiUS1OdXMiX
SwgImlzcyI6ICJodHRwczovL2V4YW1wbGUuY29tL2lzc3VlciIsICJpYXQiOiAxNTE2M
jM5MDIyLCAiZXhwIjogMTUxNjI0NzAyMiwgIl9zZF9hbGciOiAic2hhLTI1NiIsICJjb
mYiOiB7Imp3ayI6IHsia3R5IjogIlJTQSIsICJuIjogInBtNGJPSEJnLW9ZaEF5UFd6U
jU2QVdYM3JVSVhwMTFfSUNEa0dnUzZXM1pXTHRzLWh6d0kzeDY1NjU5a2c0aFZvOWRiR
29DSkUzWkdGX2VhZXRFMzBVaEJVRWdwR3dyRHJRaUo5enFwcm1jRmZyM3F2dmtHanR0a
DhaZ2wxZU0yYkpjT3dFN1BDQkhXVEtXWXMxNTJSN2c2SmcyT1ZwaC1hOHJxLXE3OU1oS
0c1UW9XX21UejEwUVRfNkg0YzdQaldHMWZqaDhocFdObmJQX3B2NmQxelN3WmZjNWZsN
nlWUkwwRFYwVjNsR0hLZTJXcWZfZU5HakJyQkxWa2xEVGs4LXN0WF9NV0xjUi1FR21YQ
U92MFVCV2l0U19kWEpLSnUtdlhKeXcxNG5IU0d1eFRJSzJoeDFwdHRNZnQ5Q3N2cWltW
EtlRFRVMTRxUUwxZUU3aWhjdyIsICJlIjogIkFRQUIifX19.xqgKrDO6dK_oBL3fiqdc
q_elaIGxM6Z-RyuysglGyddR1O1IiE3mIk8kCpoqcRLR88opkVWN2392K_XYfAuAmeT9
kJVisD8ZcgNcv-MQlWW9s8WaViXxBRe7EZWkWRQcQVR6jf95XZ5H2-_KA54POq3L42xj
k0y5vDr8yc08Reak6vvJVvjXpp-Wk6uxsdEEAKFspt_EYIvISFJhfTuQqyhCjnaW13X3
12MSQBPwjbHn74ylUqVLljDvqcemxeqjh42KWJq4C3RqNJ7anA2i3FU1kB4-KNZWsijY
7-op49iL7BrnIBxdlAMrbHEkoGTbFWdl7Ki17GHtDxxa1jaxQg~WyIweEd6bjNNaXFzY
3RaSV9PcERsQWJRIiwgImFkZHJlc3MiLCB7InN0cmVldF9hZGRyZXNzIjogIjEyMyBNY
WluIFN0IiwgImxvY2FsaXR5IjogIkFueXRvd24iLCAicmVnaW9uIjogIkFueXN0YXRlI
iwgImNvdW50cnkiOiAiVVMifV0~WyJxUVdtakpsMXMxUjRscWhFTkxScnJ3IiwgImZhb
WlseV9uYW1lIiwgIkRvZSJd~WyIzanFjYjY3ejl3a3MwOHp3aUs3RXlRIiwgImdpdmVu
X25hbWUiLCAiSm9obiJd~eyJhbGciOiAiUlMyNTYiLCAia2lkIjogIkxkeVRYd0F5ZnJ
pcjRfVjZORzFSYzEwVThKZExZVHJFQktKaF9oNWlfclUifQ.eyJub25jZSI6ICJYWk9V
Y28xdV9nRVBrbnhTNzhzV1dnIiwgImF1ZCI6ICJodHRwczovL2V4YW1wbGUuY29tL3Zl
cmlmaWVyIiwgImlhdCI6IDE2NzAzNjY0MzB9.aJ1HRh21dXP2_pMCHmkqNKaDN25aKuJ
GiTu2XKOegQ2xjZ_OzLu5_6gNy0qMhJIXb28-uQc7pMpWMZqkQLV3d7HGBiqiA71RsYY
UCF-4VYrFNFb63HQxOHTcu-nXALwbJgBSbQXKjdpjOj6NrIwItQNh7_EQ3vrf7E0_exm
kNoZxy0QCGqApAwUtiLtwQDExM9UYgms_ayPDZ3TJVcWHd9UtQTJugF4fTCJnq6TQLKu
UioK-kLAbDuRamnOpQ5d6Rbz3NHWdVp3nHM6bgh_3Y0zBFdg6VsmAXvnoIN9b2jzSemN
NAniAsalmd4X5pwzyK1TUHFJm-ib_kFf2G7Iapw
```

6. Verification and Processing

6.1. Processing by the Holder

The Holder MUST perform the following (or equivalent) steps when receiving a Combined Format for Issuance:

1. Separate the SD-JWT and the Disclosures in the Combined Format for Issuance.

2. Hash all of the Disclosures separately.

3. Find the places in the SD-JWT where the digests of the Disclosures are included. If any of the digests cannot be found in the SD-JWT, the Holder MUST reject the SD-JWT.

4. Decode Disclosures and obtain plaintext of the claim values.

It is up to the Holder how to maintain the mapping between the Disclosures and the plaintext claim values to be able to display them to the End-User when needed.

For presentation to a Verifier, the Holder MUST perform the following (or equivalent) steps:

1. Decide which Disclosures to release to the Verifier, obtaining proper End-User consent if necessary.

2. If Holder Binding is required, create a Holder Binding JWT.

3. Create the Combined Format for Presentation, including the selected Disclosures and, if applicable, the Holder Binding JWT.

4. Send the Presentation to the Verifier.

6.2. Verification by the Verifier

Upon receiving a Presentation, Verifiers MUST ensure that

* the SD-JWT is valid, i.e., it is signed by the Issuer and the signature is valid,

* all Disclosures are correct, i.e., their digests are referenced in the SD-JWT, and

* if Holder Binding is required, the Holder Binding JWT is signed by the Holder and valid.

To this end, Verifiers MUST follow the following steps (or equivalent):

1. Determine if Holder Binding is to be checked according to the Verifier's policy for the use case at hand. This decision MUST NOT be based on whether a Holder Binding JWT is provided by the Holder or not. Refer to Section 8.7 for details.

2. Separate the Presentation into the SD-JWT, the Disclosures (if any), and the Holder Binding JWT (if provided).

3. 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. The none algorithm MUST NOT be accepted.

    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, and not selectively disclosed.

    5. Check that the _sd_alg claim is present and its value is understood and the hash algorithm is deemed secure.

4. Create a copy of the SD-JWT payload, if required for further processing.

5. Process the Disclosures. For each Disclosure provided:

    1. Calculate the digest over the base64url string as described in Section 5.2.1.2.

    2. Find all _sd keys in the SD-JWT payload that contain a digest calculated in the previous step. Note that there might be more than one _sd arrays in on SD-JWT.

        1. If the digest cannot be found in the SD-JWT payload, the Verifier MUST reject the Presentation.

        2. If there is more than one place where the digest is included, the Verifier MUST reject the Presentation.

        3. If there is a key _sd that does not refer to an array, the Verifier MUST reject the Presentation.

        4. Otherwise, insert, at the level of the _sd claim, the claim described by the Disclosure with the claim name and claim value provided in the Disclosure.

            1. If the Disclosure is not a JSON-encoded array of three elements, the Verifier MUST reject the Presentation.

            2. If the claim name already exists at the same level, the Verifier MUST reject the

Presentation. Note that this also means that if a Holder sends the same Disclosure multiple times, the Verifier MUST reject the Presentation.

3. If the claim value contains an object with an _sd key (at the top level or nested deeper), the Verifier MUST reject the Presentation.

3. Remove all _sd claims from the SD-JWT payload.

4. Remove the claim _sd_alg from the SD-JWT payload.

6. If Holder Binding is required:

1. If Holder Binding is provided by means not defined in this specification, verify the Holder Binding according to the method used.

2. Otherwise, verify the Holder Binding JWT as follows:

1. If Holder Binding JWT is not provided, the Verifier MUST reject the Presentation.

2. Determine the public key for the Holder from the SD-JWT.

3. 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. The none algorithm MUST NOT be accepted.

4. Validate the signature over the Holder Binding JWT.

5. Check that the Holder Binding JWT is valid using nbf, iat, and exp claims, if provided in the Holder Binding JWT.

6. Determine that the Holder Binding JWT 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 Holder Binding JWT.

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

Otherwise, the processed SD-JWT payload can be passed to the application to be used for the intended purpose.

7. Enveloping the Combined Format for Issuance and Presentation

In some applications or transport protocols, it is desirable to put an SD-JWT and associated Disclosures into a JWT container. For example, an implementation may envelope all credentials and presentations, independent of their format, in a JWT to enable application-layer encryption during transport.

For such use cases, the SD-JWT and the respective Disclosures SHOULD be transported as a single string using the Combined Formats for Issuance and Presentation, respectively. Holder Binding MAY be achieved by signing the envelope JWT instead of adding a separate Holder Binding JWT as described in Section 5.5.1.

The claim _sd_jwt SHOULD be used when transporting a Combined Format unless the application or protocol defines a different claim name.

The following non-normative example shows a Combined Format for Presentation enveloped in a JWT payload:

```
{
  "iss": "https://holder.example.com",
  "sub": "did:example:123",
  "aud": "https://verifier.example.com",
  "exp": 1590000000,
  "iat": 1580000000,
  "nbf": 1580000000,
  "jti": "urn:uuid:12345678-1234-1234-1234-123456789012",
  "_sd_jwt": "eyJhbGci...emhlaUJhZzBZ~eyJhb...dYALCGg~"
}
```

Here, eyJhbGci...emhlaUJhZzBZ represents the SD-JWT and eyJhb...dYALCGg represents a Disclosure. The Combined Format for Presentation does not contain a Holder Binding JWT as the outer container can be signed instead.

8. Security Considerations

8.1. Mandatory digest computation of the revealed claim values by the Verifier

ToDo: add text explaining mechanisms that should be adopted to ensure that Verifiers validate the claim values received in HS-Disclosures JWT by calculating the digests of those values and comparing them with the digests in the SD-JWT: - create a test suite that forces digest computation by the Verifiers, and includes negative test cases in test vectors - use only implementations/ libraries that are compliant to the test suite - etc.

8.2. 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.

8.3. Manipulation of Disclosures

Holders can manipulate the Disclosures by changing the values of the claims before sending them to the Issuer. The Issuer MUST check the Disclosures to ensure that the values of the claims are correct, i.e., the digests of the Disclosures are actually present in the signed SD-JWT.

A naive Verifier that extracts all claim values from the Disclosures (without checking the hashes) and inserts them into the SD-JWT payload is vulnerable to this attack. However, in a structured SD-JWT, without comparing the digests of the Disclosures, such an implementation could not determine the correct place in a nested object where a claim needs to be inserted. Therefore, the naive implementation would not only be insecure, but also incorrect.

The steps described in Section 6.2 ensure that the Verifier checks the Disclosures correctly.

8.4. Entropy of the salt

The security model that conceals the plaintext claims relies on the fact that the salt cannot be learned or guessed by the attacker. It is vitally important to adhere to this principle. As such, the salt MUST 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. A new salt MUST be chosen for each claim.

8.5. Minimum length of the salt

The RECOMMENDED minimum length of the randomly-generated portion of the salt is 128 bits.

The Issuer MUST ensure that a new salt value is chosen for each claim, including when the same claim name occurs at different places in the structure of the SD-JWT. This can be seen in Example 3 in the

Appendix, where multiple claims with the name type appear, but each of them has a different salt.

8.6. Choice of a Hash Algorithm

For the security of this scheme, the hash algorithm is required to be preimage and collision resistant, i.e., it is infeasible to calculate the salt and claim value that result in a particular digest, and it is infeasible to find a different salt and claim value pair that result in a matching digest, respectively.

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

8.7. Holder Binding

Verifiers MUST decide whether Holder Binding is required for a particular use case or not before verifying a credential. This decision can be informed by various factors including, but not limited to the following: business requirements, the use case, the type of binding between a Holder and its credential that is required for a use case, the sensitivity of the use case, the expected properties of a credential, the type and contents of other credentials expected to be presented at the same time, etc.

This can be showcased based on two scenarios for a mobile driver's license use case for SD-JWT:

Scenario A: For the verification of the driver's license when stopped by a police officer for exceeding a speed limit, Holder Binding may be necessary to ensure that the person driving the car and presenting the license is the actual Holder of the license. The Verifier (e.g., the software used by the police officer) will ensure that a Holder Binding JWT is present and signed with the Holder's private key.

Scenario B: A rental car agency may want to ensure, for insurance purposes, that all drivers named on the rental contract own a government-issued driver's license. The signer of the rental contract can present the mobile driver's license of all named drivers. In this case, the rental car agency does not need to check Holder Binding as the goal is not to verify the identity of the person presenting the license, but to verify that a license exists and is valid.

It is important that a Verifier does not make its security policy decisions based on data that can be influenced by an attacker or that can be misinterpreted. For this reason, when deciding whether

Holder binding is required or not, Verifiers MUST NOT take into account

- whether an Holder Binding JWT is present or not, as an attacker can remove the Holder Binding JWT from any Presentation and present it to the Verifier, or

- whether a key reference is present in the SD-JWT or not, as the Issuer might have added the key to the SD-JWT in a format/claim that is not recognized by the Verifier.

If a Verifier has decided that Holder Binding is required for a particular use case and the Holder Binding is not present, does not fulfill the requirements (e.g., on the signing algorithm), or no recognized key reference is present in the SD-JWT, the Verifier will reject the presentation, as described in Section 6.2.

8.8. Blinding Claim Names

SD-JWT ensures that names of claims that are selectively disclosable are always blinded. This prevents an attacker from learning the names of the disclosable claims. However, the names of the claims that are not disclosable are not blinded. This includes the keys of objects that themselves are not blinded, but contain disclosable claims. This limitation needs to be taken into account by Issuers when creating the structure of the SD-JWT.

9. Privacy Considerations

9.1. Confidentiality during Transport

If the SD-JWT and associated Disclosures are transmitted over an insecure channel during issuance or presentation, an adversary may be able to intercept and read the End-User's personal data or correlate the information with previous uses of the same SD-JWT.

Usually, transport protocols for issuance and presentation of credentials are designed to protect the confidentiality of the transmitted data, for example, by requiring the use of TLS.

This specification therefore considers the confidentiality of the data to be provided by the transport protocol and does not specify any encryption mechanism.

Implementers MUST ensure that the transport protocol provides confidentiality, if the privacy of End-User data or correlation attacks are a concern. Implementers MAY define an envelope format (such as described in Section 7 or nesting the SD-JWT Combined Format as the plaintext payload of a JWE) to encrypt the SD-JWT and associated Disclosures when transmitted over an insecure channel.

9.2. Decoy Digests

The use of decoy digests is RECOMMENDED when the number of claims (or the existence of particular claims) can be a side-channel disclosing information about otherwise undisclosed claims. In particular, if a claim in an SD-JWT is present only if a certain condition is met (e.g., a membership number is only contained if the End-User is a member of a group), the Issuer SHOULD add decoy digests when the condition is not met.

Decoy digests increase the size of the SD-JWT. The number of decoy digests (or whether to use them at all) is a trade-off between the size of the SD-JWT and the privacy of the End-User's data.

9.3. Unlinkability

Colluding Issuer/Verifier or Verifier/Verifier pairs could link issuance/presentation or two presentation sessions to the same user on the basis of unique values encoded in the SD-JWT (Issuer signature, salts, digests, etc.).

To prevent these types of linkability, various methods, including but not limited to the following ones can be used:

* Use advanced cryptographic schemes, outside the scope of this specification.

* Issue a batch of SD-JWTs to the Holder to enable the Holder to use a unique SD-JWT per Verifier. This only helps with Verifier/Verifier unlinkability.

10. Acknowledgements

We would like to thank Alen Horvat, Arjan Geluk, Christian Paquin, David Bakker, David Waite, Fabian Hauck, Giuseppe De Marco, Kushal Das, Mike Jones, Nat Sakimura, Orie Steele, Pieter Kasselman, Ryosuke Abe, Shawn Butterfield, and Torsten Lodderstedt Vittorio Bertocci for their contributions (some of which substantial) to this draft and to the initial set of implementations.

The work on this draft was started at OAuth Security Workshop 2022 in Trondheim, Norway.

11. IANA Considerations

TBD

12. Normative References

[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>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

[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>.

[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>.

13. Informative References

[IANA.Hash.Algorithms] IANA, "Named Information Hash Algorithm", <https://www.iana.org/assignments/named-information/named-information.xhtml>.

[OIDC.IDA] Lodderstedt, T., Fett, D., Haine, M., Pulido, A., Lehmann, K., and K. Koiwai, "OpenID Connect for Identity Assurance 1.0", <https://openid.net/specs/openid-connect-4-identity-assurance-1_0-13.html>.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[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>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/info/rfc7800>.

[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>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/info/rfc8785>.

[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

All of the following examples are non-normative.

A.1. Example 2a: Handling Structured Claims

This example uses the following object as the set of claims that the Issuer is issuing:

```
{
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "given_name": "  ",
  "family_name": "  ",
  "email": "\"unusual email address\"@nihon.com",
  "phone_number": "+81-80-1234-5678",
  "address": {
    "street_address": "         – ",
    "locality": "   ",
    "region": "  ",
    "country": "JP"
  },
  "birthdate": "1940-01-01"
}
```

Note that in contrast to Example 1, here the Issuer decided to create a structured object for the address claim, allowing for separate disclosure of the individual members of the claim.

```
{
  "_sd": [
    "0lkScKjiJMHedJfq7sJB7HQ3qomGfrELbo5gZhvKRWo",
    "1h29ggQi1xou-_NjVyyoChK2aswUto2UjCfF-1LahA8",
    "3CoLULmDxxUw_LdyYETjuduXuEpGuBy4rXHotuD40X4",
    "AIDyoxx1ipy45-GFpKgvc-HIbcVrlLlrYLXmx3ev2e4",
    "OlthfRoFdS-J6S82OWlrO4pWhoeRMrH_KGP_h6EapYQ",
    "r4dbpGeeia02SyGL5gBdHY1w8IhjT3xx05RsfyyHUk4"
  ],
  "address": {
    "_sd": [
      "6OKNwnGGKWXCI9uijNAsv64u22eLS4rMdLNlg1fqJp4",
      "HEVMgD-NKK5NvXPbARorVdODITmkUyuMpCs_m7HXnYc",
      "Uq02nV73K0bdRK23rzabmnDa4O8YNQZvt9x8Lykok_Y",
      "oxFRilno26UYe7kqMM4bdq8IvNMtIi6F8ptt-uiPLbM"
    ]
  },
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1516247022,
  "_sd_alg": "sha-256"
}
```

The Disclosures for this SD-JWT are as follows:

Disclosure for sub:

```
WyJWb1p5VUUtNlFvQ3hVLThHeTl5RTBnIiwgInN1YiIsICI2YzVjMGE0OS1iNTg5LTQz
MWQtYmFlNy0yMTkxMjJhOWVjMmMiXQ
```

Contents:

```
["VoZyUE-6QoCxU-8Gy9yE0g", "sub",
"6c5c0a49-b589-431d-bae7-219122a9ec2c"]
```

SHA-256 Hash: AIDyoxx1ipy45-GFpKgvc-HIbcVrlLlrYLXmx3ev2e4

Disclosure for given_name:

```
WyJicjgxenVSc0NUcXJuWEp4MHVqMkRRIiwgImdpdmVuX25hbWUiLCAiXHU1OTJhXHU5
MGNlIl0
```

Contents:

```
["br81zuRsCTqrnXJx0uj2DQ", "given_name", "\u592a\u90ce"]
```

SHA-256 Hash: 1h29ggQi1xou-_NjVyyoChK2aswUto2UjCfF-1LahA8

Disclosure for family_name:

WyJSdHczZUFFUE5wWjIwTkhZSzNNRWNnIiwgImZhbWlseV9uYW1lIiwgIlx1NWM3MVx1NzUzMCJd

Contents:

["Rtw3eAEPNpZ20NHYK3MEcg", "family_name", "\u5c71\u7530"]

SHA-256 Hash: 0lkScKjiJMHedJfq7sJB7HQ3qomGfrELbo5gZhvKRWo

Disclosure for email:

WyJJTms2bkx4WGFybDF4NmVabHdBOTV3IiwgImVtYWlsIiwgIlwidW51c3VhbCBlbWFpbCBhZGRyZXNzXCJAbmlob24uY29tIl0

Contents:

["INk6nLxXarl1x6eZlwA95w", "email", "\"unusual email address\"@nihon.com"]

SHA-256 Hash: OlthfRoFdS-J6S82OWlrO4pWhoeRMrH_KGP_h6EapYQ

Disclosure for phone_number:

WyJuUG9LdV9oakkyeTMtbTVnbmVkYzFBIiwgInBob25lX251bWJlciIsICIrODEtODAtMTIzNC01Njc4Il0

Contents:

["nPoKu_hjI2y3-m5gnedc1A", "phone_number", "+81-80-1234-5678"]

SHA-256 Hash: r4dbpGeeia02SyGL5gBdHY1w8IhjT3xx05RsfyyHUk4

Disclosure for street_address:

WyJiLWtOOG9nVjg4N0FHY2RzRnQyT1RBIiwgInN0cmVldF9hZGRyZXNzIiwgIlx1Njc3MVx1NGVhY1x1OTBmZFx1NmUyZlx1NTMzYVx1ODI5ZFx1NTE2Y1x1NTcxMlx1ZmYxNFx1NGUwMVx1NzZlZVx1ZmYxMlx1MjIxMlx1ZmYxOCJd

Contents:

["b-kN8ogV887AGcdsFt2OTA", "street_address", "\u6771\u4eac\u90fd\u6e2f\u533a\u829d\u516c\u5712\uff14\u4e01\u76ee\uff12\u2212\uff18"]

SHA-256 Hash: HEVMgD-NKK5NvXPbARorVdODITmkUyuMpCs_m7HXnYc

Disclosure for locality:

WyJnclBTdm9iLVVfbHVOSVJUUmJvb2dBIiwgImxvY2FsaXR5IiwgIlx1Njc3MVx1NGVhY1x1OTBmZCJd

Contents:

["grPSvob-U_luNIRTRboogA", "locality", "\u6771\u4eac\u90fd"]

SHA-256 Hash: 6OKNwnGGKWXCI9uijNAsv64u22eLS4rMdLNlg1fqJp4

Disclosure for region:

WyJNOVY2N3V0UC1hTF9lR1B0UU5hM0RRIiwgInJlZ2lvbiIsICJcdTZlMmZcdTUzM2Ei
XQ

Contents:

["M9V67utP-aL_eGPtQNa3DQ", "region", "\u6e2f\u533a"]

SHA-256 Hash: Uq02nV73K0bdRK23rzabmnDa4O8YNQZvt9x8Lykok_Y

Disclosure for country:

WyJzNFhNSmxXQ2Eza3hDWk4wSVVrbnlBIiwgImNvdW50cnkiLCAiSlAiXQ

Contents:

["s4XMJlWCa3kxCZN0IUknyA", "country", "JP"]

SHA-256 Hash: oxFRilno26UYe7kqMM4bdq8IvNMtIi6F8ptt-uiPLbM

Disclosure for birthdate:

WyI1Z2NXRmxWSEM1VVEwbktrallybDlnIiwgImJpcnRoZGF0ZSIsICIxOTQwLTAxLTAx
Il0

Contents:

["5gcWFlVHC5UQ0nKkjYrl9g", "birthdate", "1940-01-01"]

SHA-256 Hash: 3CoLULmDxxUw_LdyYETjuduXuEpGuBy4rXHotuD40X4

A Presentation for the SD-JWT that discloses only region and country of the address property and without a Holder Binding JWT could look as follows:

eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJfc2QiOiBbIjBsa1NjS2ppSk1IZWRKZnE3c0pCN
0hRM3FvbUdmckVMYm81Z1podktSV28iLCAiMWgyOWdnUWkxeG91LV9OalZ5eW9DaEsyY
XN3VXRvMlVqQ2ZGLTFMYWhBOCIsICIzQ29MVUxtRHh4VXdfTGR5WUVUanVkdVh1RXBHd
UJ5NHJYSG90dUQ0MFg0IiwgIkFJRHlveHgxaXB5NDUtR0ZwS2d2Yy1ISWJjVnJsTGxyW
UxYbXgzZXYyZTQiLCAiT2x0aGZSb0ZkUy1KNlM4Mk9XbHJPNHBXaG9lUk1ySF9LR1Bfa
DZFYXBZUSIsICJyNGRicEdlZWlhMDJTeUdMNWdCZEhZMXc4SWhqVDN4eDA1UnNmeXlIV
Ws0Il0sICJhZGRyZXNzIjogeyJfc2QiOiBbIjZPS053bkdHS1dYQ0k5dWlqTkFzdjY0d
TIyZUxTNHJNZExObGcxZnFKcDQiLCAiSEVWTWdELU5LSzVOdlhQYkFSb3JWZE9ESVRta
1V5dU1wQ3NfbTdIWG5ZYyIsICJVcTAyblY3M0swYmRSSzIzcnphYm1uRGE0TzhZTlFad
nQ5eDhMeWtva19ZIiwgIm94RlJpbG5vMjZVWWU3a3FNTTRiZHE4SXZOTXRJaTZGOHB0d
C11aVBMYk0iXX0sICJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXIiLCAia
WF0IjogMTUxNjIzOTAyMiwgImV4cCI6IDE1MTYyNDcwMjIsICJfc2RfYWxnIjogInNoY
S0yNTYifQ.MDBuqNanBUtQtvI0W18zYI7nXqys-EgkEeuMjDFehYkzWifFYI8qbIb-vD
keV6TvMUc3tiV5FWMgoF_1UNVG6msjL1MLC0ztXcIqtEJOMpAx0w13Jaxv4QOTy6NXkI
-DnKxlNxFp6F7Fd3lxaYzVzGFHfTo9AzTfy5ZrNgVmfbgCU40xqhR0Xjjgm1CsFn0dl8
ib2S-CgxcdYUkpKWp2G9eqXkUcWQ55bgUQQdgPTCpGoalads8bcdki5e1vD2eDsfquDy
_JL48rwIJWBELtihJxeT7plnARgKyJk889i5VBXb4Qyr5QZErcgoZVIi6xygzuVeYoKI
l8STyH0f85wQ~WyJSdHczZUFFUE5wWjIwTkhZSzNNRWNnIiwgImZhbWlseV9uYW1lIiw
gIlx1NWM3MVx1NzUzMCJd~WyJicjgxenVSc0NUcXJuWEp4MHVqMkRRIiwgImdpdmVuX2
5hbWUiLCAiXHU1OTJhXHU5MGNlIl0~WyI1Z2NXRmxWSEM1VVEwbktrallybDlnIiwgIm
JpcnRoZGF0ZSIsICIxOTQwLTAxLTAxIl0~WyJJTms2bkx4WGFybDF4NmVabHdBOTV3Ii
wgImVtYWlsIiwgIlwidW51c3VhbCBlbWFpbCBhZGRyZXNzXCJAbmlob24uY29tIl0~Wy
JNOVY2N3V0UC1hTF9lR1B0UU5hM0RRIiwgInJlZ2lvbiIsICJcdTZlMmZcdTUzM2EiXQ
~WyJzNFhNSmxXQ2Eza3hDWk4wSVVrbnlBIiwgImNvdW50cnkiLCAiSlAiXQ~

A.2. Example 2b: Adding Decoys

This example is based on the same set of user data as Example 2a, but here, the Issuer decided to add decoy digests to the SD-JWT.

The SD-JWT payload is as follows:

```
{
  "_sd": [
    "2GFPzomyv0LvsNqVnq9EWkojSdmn8xYhEW6PUF166aE",
    "723eL1tWNqk0k60oEc2jbjgi9Lg09hp_qr5BL2omA8w",
    "ABOPJ9UwDrrYOj3XF9FldPk4qniliHktfC0faS2nEDk",
    "Lccl25w97vsyfqBTCOMXAtvCt_8f4BlqDUBhrvdS2VI",
    "QSBjkY_KqFtGhL7--yjUU0BTbOQKXXEKTzmoVAzCUhY",
    "iVw--s-TwoFGMsv1dhoq1sBsjrBKyKtJoQ5DZu3Oq1I",
    "nGzC_ytvNpNRhxNqUJ8yGd3DtGv-3u6goHM7JqsRzCw",
    "skyXccgGkjo2BK4p19vDmbl2JsnKBdbiVBhqV5b_RXk",
    "woMsAJhxWETDjavMSyNVyu62nyy9PzFfZw9lz5yfC4U"
  ],
  "address": {
    "_sd": [
      "1nY2u90v7ur8X8mniVZ2tSRNRh6DPnLBYtDEtRS0zM8",
      "F_xLpiE-o4hv3QPy7WpIyQJYEhYVvlG0b5PguzbG64A",
      "Os3GWFP8wcrl01HgFG3EQgX4lLSeGSWcLliHH2Nr4wQ",
      "nQj9x2LdAW4uGv9zl8T3ElZmbf_XScaN8maE-K6kGO0",
      "opzyobnErA3GpoURHpkHGUpmnVGT74GxP9PrdySlxAA",
      "q2Mq7M5Lb6u7qbteQF0BzN6gqs_CbP9bx22xoLjbx2w",
      "xuJuhq1aJqiYgdIxMRmMJNTBkzivgvV9Zk4jmmJYSkk"
    ]
  },
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1516247022,
  "_sd_alg": "sha-256"
}
```


Since the Disclosures or Presentation are not affected by the decoy digests (other than a slightly larger SD-JWT), they are omitted here.

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

In this example, an SD-JWT with a complex object is demonstrated. Here, the data structures defined in OIDC4IDA [OIDC.IDA] are used.

The Issuer is using the following user data:

```
{
  "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": "Müller",
      "nationalities": [
        "DE"
      ],
      "birthdate": "1956-01-28",
      "place_of_birth": {
        "country": "IS",
        "locality": "Þykkvabæjarklaustur"
      },
      "address": {
        "locality": "Maxstadt",
        "postal_code": "12344",
        "country": "DE",
        "street_address": "Weidenstraße 22"
      }
    }
  },
  "birth_middle_name": "Timotheus",
  "salutation": "Dr.",
  "msisdn": "49123456789"
}
```

The Issuer in this example sends the two claims birthdate and place_of_birth in the claims element in plain text. The following shows the resulting SD-JWT payload:

```
{
  "_sd": [
    "0fGhQf6nEvYH9sTcCtkxCfAVKp9qAYDVmFsxADWJsio",
    "H0ZuxXsa39gwxHQ0cK_HQAIKjW4t11YwM0Gy_1gUluU",
    "gXzq9tVKaddfQvt94riVze6GlZiW35ItiMngC5p_hXE"
  ],
  "verified_claims": {
    "verification": {
      "_sd": [
        "4nQI4-y9wtJvyCs2WttyLmITY02K5WBMKKT1qKeEhHE",
        "Zvx3GXqsjB1cR5rmxrlv3xQuTykWDZ7lKlmaxtBVYo0",
        "cCzobgUal0u_Dn0PengU8YcZfv_R3xh0MW1K01Okg1o"
      ],
      "evidence": [
        {
          "_sd": [
            "18o6apk_mELe_RaWjI1Om877h9_Jky5dpgzuBBNl-dQ",
            "Ed7fTsdbgv2EZ5uGsuUAk5YBaMtCfvTFkmZZszVRJpk",
            "rvJdOi2J6gFeNIvCdgIprrjmi1rKTD_Ru96di5FS7i4"
          ],
          "document": {
            "_sd": [
              "BiKOIkmtJfr7IpWsn8_X4hc6FygqaeQPujoVuXAHnWQ",
              "F1vtjNyR7vaY8bO6CHdFr9tAje0ltSLe7bIFYlpNW7s",
              "HXyhyliGWAP8uvBf3JhdGSh9Pbg3KNu969581e8MZPo",
              "sVzUIXAzzMw0EfIwZyL23xwee3874u3a7iXcWyZhwp4"
            ],
            "issuer": {
              "_sd": [
                "C1saJVc21n1Y1O2s4ya0UYJabTAoGyg__Q39OfyjqSM",
                "Li9BJSo7AFcKVTlrLz6ip6t82nqIt2qrXc3wp6rColM"
              ]
            }
          }
        }
      ]
    },
    "claims": {
      "_sd": [
        "1qb26tNg6OZuZyVDYwK4--mQxXbZqwcQbhUxGHrXeLM",
        "AHX0EgNpd_wak07lK8HX2izDNntsUZojuzyEWd2GJdk",
        "FwzTz0THaEOzexgEzLRXu-zsTPND7by3aBF57AwKCZI",
        "xKbTMlOSjFjkJw8kAMZAyhEfqbaq2b-klaPK8srpgvs"
      ],
      "birthdate": "1956-01-28",
      "place_of_birth": {
        "country": "IS",
        "locality": "Þykkvabæjarklaustur"
      }
```

```
    }
  },
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1516247022,
  "_sd_alg": "sha-256"
}
```

With the following Disclosures:

Disclosure for trust_framework:

WyI2eUNER1RSblBzUHowS3RxSVh3cjhRIiwgInRydXN0X2ZyYW1ld29yayIsICJkZV9h
bWwiXQ

Contents:

["6yCDGTRnPsPz0KtqIXwr8Q", "trust_framework", "de_aml"]

SHA-256 Hash: Zvx3GXqsjB1cR5rmxrlv3xQuTykWDZ7lKlmaxtBVYo0

Disclosure for time:

WyI2TlpFYW9FUmVmdzJ2YTl3VWZDSlJBIiwgInRpbWUiLCAiMjAxMi0wNC0yM1QxODoy
NVoiXQ

Contents:

["6NZEaoERefw2va9wUfCJRA", "time", "2012-04-23T18:25Z"]

SHA-256 Hash: cCzobgUal0u_Dn0PengU8YcZfv_R3xh0MW1K01Okg1o

Disclosure for verification_process:

WyJTWVhaS291anJYdXZPZ3hjbnVZS1J3IiwgInZlcmlmaWNhdGlvbl9wcm9jZXNzIiwg
ImYyNGM2Zi02ZDNmLTRlYzUtOTczZS1iMGQ4NTA2ZjNiYzciXQ

Contents:

["SYXZKoujrXuvOgxcnuYKRw", "verification_process",
"f24c6f-6d3f-4ec5-973e-b0d8506f3bc7"]

SHA-256 Hash: 4nQI4-y9wtJvyCs2WttyLmITY02K5WBMKKT1qKeEhHE

Disclosure for type:

WyJyNE1TdlZEY1hQMFlFSmNsd1BJTmRnIiwgInR5cGUiLCAiZG9jdW1lbnQiXQ

Contents:

["r4MSvVDcXP0YEJclwPINdg", "type", "document"]

SHA-256 Hash: rvJdOi2J6gFeNIvCdgIprrjmi1rKTD_Ru96di5FS7i4

Disclosure for method:

WyJVYmI4MGZRYndOOWx0S0xYc0RvbkdnIiwgIm1ldGhvZCIsICJwaXBwIl0

Contents:

["Ubb80fQbwN9ltKLXsDonGg", "method", "pipp"]

SHA-256 Hash: 18o6apk_mELe_RaWjI1Om877h9_Jky5dpgzuBBNl-dQ

Disclosure for time:

WyJaRW1vWTc0ZlNvUXdxOFlWMkxGMmt3IiwgInRpbWUiLCAiMjAxMi0wNC0yMlQxMTozMFoiXQ

Contents:

["ZEmoY74fSoQwq8YV2LF2kw", "time", "2012-04-22T11:30Z"]

SHA-256 Hash: Ed7fTsdbgv2EZ5uGsuUAk5YBaMtCfvTFkmZZszVRJpk

Disclosure for type:

WyJtanBjRGkweHFZbTF4aFJyQ3lCeGZnIiwgInR5cGUiLCAiaWRjYXJkIl0

Contents:

["mjpcDi0xqYm1xhRrCyBxfg", "type", "idcard"]

SHA-256 Hash: F1vtjNyR7vaY8bO6CHdFr9tAje0ltSLe7bIFYlpNW7s

Disclosure for name:

WyJEY0IzOXRwcHBqbGhjVFctLUhKYVJ3IiwgIm5hbWUiLCAiU3RhZHQgQXVnc2J1cmciXQ

Contents:

["DcB39tpppjlhcTW--HJaRw", "name", "Stadt Augsburg"]

SHA-256 Hash: Li9BJSo7AFcKVTlrLz6ip6t82nqIt2qrXc3wp6rColM

Disclosure for country:

WyJGbWU0VVdmeDZBV0g1NS1BOHJXRF93IiwgImNvdW50cnkiLCAiREUiXQ

Contents:

["Fme4UWfx6AWH55-A8rWD_w", "country", "DE"]

SHA-256 Hash: C1saJVc21n1Y1O2s4ya0UYJabTAoGyg__Q39OfyjqSM

Disclosure for number:

WyIzZ3NJRUs0NWM4bkZaT09YbWJELVRBIiwgIm51bWJlciIsICI1MzU1NDU1NCJd

Contents:

["3gsIEK45c8nFZOOXmbD-TA", "number", "53554554"]

SHA-256 Hash: BiKOIkmtJfr7IpWsn8_X4hc6FygqaeQPujoVuXAHnWQ

Disclosure for date_of_issuance:

WyJrNlZGS0I3V29xeWZXT1M0RVZqU3d3IiwgImRhdGVfb2ZfaXNzdWFuY2UiLCAiMjAx
MC0wMy0yMyJd

Contents:

["k6VFKB7WoqyfWOS4EVjSww", "date_of_issuance", "2010-03-23"]

SHA-256 Hash: HXyhyliGWAP8uvBf3JhdGSh9Pbg3KNu969581e8MZPo

Disclosure for date_of_expiry:

WyJqTEtGZlFYQk5FdHRFRzR1UGV6cnlnIiwgImRhdGVfb2ZfZXhwaXJ5IiwgIjIwMjAt
MDMtMjIiXQ

Contents:

["jLKFfQXBNEttEG4uPezryg", "date_of_expiry", "2020-03-22"]

SHA-256 Hash: sVzUIXAzzMw0EfIwZyL23xwee3874u3a7iXcWyZhwp4

Disclosure for given_name:

WyJ2WmN5ZDRDTDhFeDFBUGlSb0tVVFB3IiwgImdpdmVuX25hbWUiLCAiTWF4Il0

Contents:

["vZcyd4CL8Ex1APiRoKUTPw", "given_name", "Max"]

SHA-256 Hash: xKbTMlOSjFjkJw8kAMZAyhEfqbaq2b-klaPK8srpgvs

Disclosure for family_name:

WyJod0lFTklDMWwtQlRuZnNfMnZQV2ZBIiwgImZhbWlseV9uYW1lIiwgIk1cdTAwZmNs
bGVyIl0

Contents:

["hwIENIC1l-BTnfs_2vPWfA", "family_name", "M\u00fcller"]

SHA-256 Hash: 1qb26tNg6OZuZyVDYwK4--mQxXbZqwcQbhUxGHrXeLM

Disclosure for nationalities:

WyJHRFhMLTBfWWlvYUxNU1l0RzdCdkxnIiwgIm5hdGlvbmFsaXRpZXMiLCBbIkRFIl1d

Contents:

["GDXL-0_YioaLMSYtG7BvLg", "nationalities", ["DE"]]

SHA-256 Hash: FwzTz0THaEOzexgEzLRXu-zsTPND7by3aBF57AwKCZI

Disclosure for address:

WyJVUjZsSC1jbUtIenh5VGlhazRmRVVRIiwgImFkZHJlc3MiLCB7ImxvY2FsaXR5Ijog
Ik1heHN0YWR0IiwgInBvc3RhbF9jb2RlIjogIjEyMzQ0IiwgImNvdW50cnkiOiAiREUi
LCAic3RyZWV0X2FkZHJlc3MiOiAiV2VpZGVuc3RyYVx1MDBkZmUgMjIifV0

Contents:

["UR6lH-cmKHzxyTiak4fEUQ", "address", {"locality": "Maxstadt",
"postal_code": "12344", "country": "DE", "street_address":
"Weidenstra\u00dfe 22"}]

SHA-256 Hash: AHX0EgNpd_wak07lK8HX2izDNntsUZojuzyEWd2GJdk

Disclosure for birth_middle_name:

WyJuNV9BU3lpa2FSNHBZMzRrVG9YTlBRIiwgImJpcnRoX21pZGRsZV9uYW1lIiwgIlRp
bW90aGV1cyJd

Contents:

["n5_ASyikaR4pY34kToXNPQ", "birth_middle_name", "Timotheus"]

SHA-256 Hash: gXzq9tVKaddfQvt94riVze6GlZiW35ItiMngC5p_hXE

Disclosure for salutation:

WyJFSGh2d1dnWGg2VUZKcVRmRjVRaE93IiwgInNhbHV0YXRpb24iLCAiRHIuIl0

Contents:

["EHhvwWgXh6UFJqTfF5QhOw", "salutation", "Dr."]

SHA-256 Hash: H0ZuxXsa39gwxHQ0cK_HQAIKjW4t11YwM0Gy_1gUluU

Disclosure for msisdn:

WyJTMzF2RjdWZHNpUDhRTXB0b21PZW1BIiwgIm1zaXNkbiIsICI0OTEyMzQ1Njc4OSJd

Contents:

["S31vF7VdsiP8QMptomOemA", "msisdn", "49123456789"]

SHA-256 Hash: 0fGhQf6nEvYH9sTcCtkxCfAVKp9qAYDVmFsxADWJsio

The Verifier would receive the Issuer-signed SD-JWT together with a selection of the Disclosures. The Presentation in this example would look as follows:

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJfc2QiOiBbIjBmR2hRZjZuRXZZSDlzVGNDdGt4Q
2ZBVktwOXFBWURWbUZzeEFEV0pzaW8iLCAiSDBadXhYc2EzOWd3eEhRMGNLX0hRQUlLa
lc0dDExWXdNMEd5XzFnVWx1VSIsICJnWHpxOXRWS2FkZGZRdnQ5NHJpVnplNkdsWmlXM
zVJdGlNbmdDNXBfaFhFIl0sICJ2ZXJpZmllZF9jbGFpbXMiOiB7InZlcmlmaWNhdGlvb
iI6IHsiX3NkIjogWyI0blFJNC15OXd0SnZ5Q3MyV3R0eUxtSVRZMDJLNVdCTUtLVDFxS
2VFaEhFIiwgIlp2eDNHWHFzakIxY1I1cm14cmx2M3hRdVR5a1dEWjdsS2xtYXh0QlZZb
zAiLCAiY0N6b2JnVWFsMHVfRG4wUGVuZ1U4WWNaZnZfUjN4aDBNVzFLMDFPa2cxbyJdL
CAiZXZpZGVuY2UiOiBbeyJfc2QiOiBbIjE4bzZhcGtfbUVMZV9SYVdqSTFPbTg3N2g5X
0preTVkcGd6dUJCTmwtZFEiLCAiRWQ3ZlRzZGJndjJFWjV1R3N1VUFrNVlCYU10Q2Z2V
EZrbVpac3pWUkpwayIsICJydkpkT2kySjZnRmVOSXZDZGdJcHJyam1pMXJLVERfUnU5N
mRpNUZTN2k0Il0sICJkb2N1bWVudCI6IHsiX3NkIjogWyJCaUtPSWttdEpmcjdJcFdzb
jhfWDRoYzZGeWdxYWVRUHVqb1Z1WEFIbldRIiwgIkYxdnRqTnlSN3ZhWThiTzZDSGRGc
jl0QWplMGx0U0xlN2JJRllscE5XN3MiLCAiSFh5aHlsaUdXQVA4dXZCZjNKaGRHU2g5U
GJnM0tOdTk2OTU4MWU4TVpQbyIsICJzVnpVSVhBenpNdzBFZkl3WnlMMjN4d2VlMzg3N
HUzYTdpWGNXeVpod3A0Il0sICJpc3N1ZXIiOiB7Il9zZCI6IFsiQzFzYUpWYzIxbjFZM
U8yczR5YTBVWUphYlRBb0d5Z19fUTM5T2Z5anFTTSIsICJMaTlCSlNvN0FGY0tWVGxyT
Ho2aXA2dDgybnFJdDJxclhjM3dwNnJDb2xNIl19fX1dfSwgImNsYWltcyI6IHsiX3NkI
jogWyIxcWIyNnROZzZPWnVaeVZEWXdLNC0tbVF4WGJacXdjUWJoVXhHSHJYZUxNIiwgI
kFIWDBFZ05wZF93YWswN2xLOEhYMml6RE5udHNVWm9qdXp5RVdkMkdKZGsiLCAiRnd6V
HowVEhhRU96ZXhnRXpMUlh1LXpzVFBORDdieTNhQkY1N0F3S0NaSSIsICJ4S2JUTWxPU
2pGamtKdzhrQU1aQXloRWZxYmFxMmIta2xhUEs4c3JwZ3ZzIl0sICJiaXJ0aGRhdGUiO
iAiMTk1Ni0wMS0yOCIsICJwbGFjZV9vZl9iaXJ0aCI6IHsiY291bnRyeSI6ICJJUyIsI
CJsb2NhbGl0eSI6ICJcdTAwZGV5a2t2YWJcdTAwZTZqYXJrbGF1c3R1ciJ9fX0sICJpc
3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXIiLCAiaWF0IjogMTUxNjIzOTAyM
iwgImV4cCI6IDE1MTYyNDcwMjIsICJfc2RfYWxnIjogInNoYS0yNTYifQ.a7Chaa1KrV
l6IhnmWDnRxEKdjess03TWHjzgp1aUoj2JXD97DwePyKg1K-DBgJehOtU6k7tdcrwIMq
zLnQWDMukCNyT-hBkxZ8jArtB4S5hyDLde1-B5toMVBGF-nv2Oq0TqBV9FlTg884zAPS
WcEsv0OYDDnj5b0pLD5cHCkc3IREkGsCB3Cc07G6beDakS2_QPLZ763Uz3bYX0XFPMDV
7z3xC6vqzUXyKL76_ENQyzyArC8a5e3OegQQ5RY7mdm9zJUUzi-rn_QnTambpiyh9ux-
41ABwf8vHgCe953EBqD9KJNh7ZvQcOstUFEzqV6loSU3gquy2UhZUfXRDSCw~WyI2eUN
ER1RSblBzUHowS3RxSVh3cjhRIiwgInRydXN0X2ZyYW1ld29yayIsICJkZV9hbWwiXQ~
WyI2TlpFYW9FUmVmdzJ2YTl3VWZDSlJBIiwgInRpbWUiLCAiMjAxMi0wNC0yM1QxODoy
NVoiXQ~WyJyNE1TdlZEY1hQMFlFSmNsd1BJTmRnIiwgInR5cGUiLCAiZG9jdW1lbnQiX
Q~WyJod0lFTklDMWwtQlRuZnNfMnZQV2ZBIiwgImZhbWlseV9uYW1lIiwgIk1cdTAwZm
NsbGVyIl0~WyJVUjZsSC1jbUtIenh5VGlhazRmRVVRIiwgImFkZHJlc3MiLCB7ImxvY2
FsaXR5IjogIk1heHN0YWR0IiwgInBvc3RhbF9jb2RlIjogIjEyMzQ0IiwgImNvdW50cn
kiOiAiREUiLCAic3RyZWV0X2FkZHJlc3MiOiAiV2VpZGVuc3RyYVx1MDBkZmUgMjIifV
0~WyJ2WmN5ZDRDTDhFeDFBUGlSb0tVVFB3IiwgImdpdmVuX25hbWUiLCAiTWF4Il0~
```

After the verification of the data, the Verifier will pass the following result on to the application for further processing:

```
{
  "verified_claims": {
    "verification": {
      "evidence": [
        {
          "document": {
            "issuer": {}
          },
          "type": "document"
        }
      ],
      "trust_framework": "de_aml",
      "time": "2012-04-23T18:25Z"
    },
    "claims": {
      "birthdate": "1956-01-28",
      "place_of_birth": {
        "country": "IS",
        "locality": "Þykkvabæjarklaustur"
      },
      "family_name": "Müller",
      "address": {
        "locality": "Maxstadt",
        "postal_code": "12344",
        "country": "DE",
        "street_address": "Weidenstraße 22"
      },
      "given_name": "Max"
    }
  },
  "iss": "https://example.com/issuer",
  "iat": 1516239022,
  "exp": 1516247022,
  "_sd_alg": "sha-256"
}
```

A.4. Example 4 - W3C Verifiable Credentials Data Model (work in progress)

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

SD-JWT is equivalent to an Issuer-signed W3C Verifiable Credential (W3C VC). Disclosures are sent alongside a VC.

A Presentation with a Holder Binding JWT is equivalent to a Holder-signed W3C Verifiable Presentation (W3C VP).

Holder Binding is applied and the Holder Binding JWT is signed using a raw public key passed in a cnf Claim in a W3C VC (SD-JWT).

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].

The following data will be used in this example:

```
{
  "iss": "https://example.com",
  "jti": "http://example.com/credentials/3732",
  "nbf": 1541493724,
  "iat": 1541493724,
  "cnf": {
    "jwk": {
      "kty": "RSA",
      "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbf
        AAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMst
        n64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_F
        DW2QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n9
        1CbOpbISD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHa
        Q-G_xBniIqbw0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
      "e": "AQAB"
    }
  },
  "type": "IdentityCredential",
  "credentialSubject": {
    "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",
    "is_over_18": true,
    "is_over_21": true,
    "is_over_65": true
  }
}
```

The encoded SD-JWT looks as follows:

Header:

```
{
  "typ": "sd-jwt-vc",
  "alg": "RS256",
  "kid": "cAEIUqJ0cmLzD1kzGzheiBag0YRAzVdlfxN280NgHaA"
}
```

Body:

```
{
  "iss": "https://example.com/issuer",
  "jti": "http://example.com/credentials/3732",
  "nbf": 1541493724,
  "iat": 1516239022,
  "cnf": {
    "jwk": {
      "kty": "RSA",
      "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbf
        AAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMst
        n64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_F
        DW2QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n9
        1CbOpbISD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHa
        Q-G_xBniIqbw0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
      "e": "AQAB"
    }
  },
  "type": "IdentityCredential",
  "credentialSubject": {
    "_sd": [
      "8Bcr2ZGImJ9FlsBNmGiLgj7XmHm-g8QOuYVSXOWoM58",
      "8wawaYT3XNlnl0zUdP8Xri1252yHK6pjgr8JInBzMQg",
      "NVTo4oQvI0kuqdMYeb6M_sp_ByUY7d4zcLfHB483aVY",
      "UQ6IHUAtHqW8Xcik7GqVikWtoviseS4BKjvCmmon6xA",
      "Um7LRM6Ol51wJZLobMeREW6HsMyO3DCCHnWtaikD-40",
      "jIjo9ElQ4Ng2mMzs0SoWq2molpyU80OE_LENjk2Ha_8",
      "w9rrWnA1RQ5dUOBaJDKGfAgtiW12obi1joeF_oLTaZ0",
      "zOJJSAmcB0OqPmFkOZebY70hHI3GpJC1x4beJRtIPbM"
    ],
    "address": {
      "_sd": [
        "0mcZNz0nIGNYBI4WgKJFx9mjaaB10E59Bc_pefmm544",
        "4zIfrBGk3TStRDz_wlxi4VgYDza81mBs_zeJ84czsS4",
        "Hl_wpSSSagpXtSk_rGoE-xkYGVnIHBO-ZzhUWchy8bo",
        "ySoQUT3JlKF5NdTPMWzmlDamB4TY2EMmGCgrzzJzih0"
      ]
    }
  },
  "exp": 1516247022,
  "_sd_alg": "sha-256"
}
```

Disclosures:

Disclosure for given_name:

```
WyI5S05NMUxWcU1PVXR6Rk9iSFV4Q2J3IiwgImdpdmVuX25hbWUiLCAiSm9obiJd
```

Contents:

```
["9KNM1LVqMOUtzFObHUxCbw", "given_name", "John"]
```

SHA-256 Hash: jIjo9ElQ4Ng2mMzs0SoWq2molpyU80OE_LENjk2Ha_8

Disclosure for family_name:

```
WyJJeC1jVTQzcXpBUFhvU2xZclF3RnRnIiwgImZhbWlseV9uYW1lIiwgIkRvZSJd
```

Contents:

```
["Ix-cU43qzAPXoSlYrQwFtg", "family_name", "Doe"]
```

SHA-256 Hash: w9rrWnA1RQ5dUOBaJDKGfAgtiW12obi1joeF_oLTaZ0

Disclosure for email:

```
WyJrbnFPOXJYYldqc2llOEtTMkJJSkFRIiwgImVtYWlsIiwgImpvaG5kb2VAZXhhbXBs
ZS5jb20iXQ
```

Contents:

```
["knqO9rXbWjsie8KS2BIJAQ", "email", "johndoe@example.com"]
```

SHA-256 Hash: NVTo4oQvI0kuqdMYeb6M_sp_ByUY7d4zcLfHB483aVY

Disclosure for phone_number:

```
WyJSbVdDWEhxaE41TDg2SllqOUdYN0h3IiwgInBob25lX251bWJlciIsICIrMS0yMDIt
NTU1LTAxMDEiXQ
```

Contents:

```
["RmWCXHqhN5L86JYj9GX7Hw", "phone_number", "+1-202-555-0101"]
```

SHA-256 Hash: zOJJSAmcB0OqPmFkOZebY70hHI3GpJC1x4beJRtIPbM

Disclosure for street_address:

```
WyJjTE84ZnZpNlVHYUstbF91QmFJYTVRIiwgInN0cmVldF9hZGRyZXNzIiwgIjEyMyBN
YWluIFN0Il0
```

Contents:

["cLO8fvi6UGaK-l_uBaIa5Q", "street_address", "123 Main St"]

SHA-256 Hash: Hl_wpSSSagpXtSk_rGoE-xkYGVnIHBO-ZzhUWchy8bo

Disclosure for locality:

WyJOSzBZQU1MWUwtLXRVdi1EdGNoaU1RIiwgImxvY2FsaXR5IiwgIkFueXRvd24iXQ

Contents:

["NK0YAMLYL--tUv-DtchiMQ", "locality", "Anytown"]

SHA-256 Hash: 4zIfrBGk3TStRDz_wlxi4VgYDza81mBs_zeJ84czsS4

Disclosure for region:

WyI3c3FoSU1aMlkzOHM4aWxaY012c2RRIiwgInJlZ2lvbiIsICJBbnlzdGF0ZSJd

Contents:

["7sqhIMZ2Y38s8ilZcMvsdQ", "region", "Anystate"]

SHA-256 Hash: ySoQUT3JlKF5NdTPMWzmlDamB4TY2EMmGCgrzzJzih0

Disclosure for country:

WyJISXlPZXkzNXEzX0ZTbDRVV0g2Mm5BIiwgImNvdW50cnkiLCAiVVMiXQ

Contents:

["HIyOey35q3_FSl4UWH62nA", "country", "US"]

SHA-256 Hash: 0mcZNz0nIGNYBI4WgKJFx9mjaaB10E59Bc_pefmm544

Disclosure for birthdate:

WyJpUU5VWU9tblo1N1Z5VHBRaGZnV2lnIiwgImJpcnRoZGF0ZSIsICIxOTQwLTAxLTAx
Il0

Contents:

["iQNUYOmnZ57VyTpQhfgWig", "birthdate", "1940-01-01"]

SHA-256 Hash: Um7LRM6Ol51wJZLobMeREW6HsMyO3DCCHnWtaikD-40

Disclosure for is_over_18:

WyJ0cmNEb1BJUDVzeW5wWFhZU0dFc0FRIiwgImlzX292ZXJfMTgiLCB0cnVlXQ

Contents:

["trcDoPIP5synpXXYSGEsAQ", "is_over_18", true]

SHA-256 Hash: 8wawaYT3XNlnl0zUdP8Xri1252yHK6pjgr8JInBzMQg

Disclosure for is_over_21:

WyJRMnhzRWpnUnBhSE5rdUdEM2tUNUpnIiwgImlzX292ZXJfMjEiLCB0cnVlXQ

Contents:

["Q2xsEjgRpaHNkuGD3kT5Jg", "is_over_21", true]

SHA-256 Hash: UQ6IHUAtHqW8Xcik7GqVikWtoviseS4BKjvCmmon6xA

Disclosure for is_over_65:

WyI0RDd2N1JWTTh2THUwN0tRVUJqOF9RIiwgImlzX292ZXJfNjUiLCB0cnVlXQ

Contents:

["4D7v7RVM8vLu07KQUBj8_Q", "is_over_65", true]

SHA-256 Hash: 8Bcr2ZGImJ9FlsBNmGiLgj7XmHm-g8QOuYVSXOWoM58

Appendix B. Document History

[[To be removed from the final specification]]

-03

- Disclosures are now delivered not as a JWT but as separate base64url-encoded JSON objects.
- In the SD-JWT, digests are collected under a _sd claim per level.
- Terms "II-Disclosures" and "HS-Disclosures" are replaced with "Disclosures".
- Holder Binding is now separate from delivering the Disclosures and implemented, if required, with a separate JWT.
- Examples updated and modified to properly explain the specifics of the new SD-JWT format.
- Examples are now pulled in from the examples directory, not inlined.
- Updated and automated the W3C VC example.
- Added examples with multibyte characters to show that the specification and demo code work well with UTF-8.

*reverted back to hash alg from digest derivation alg (renamed to _sd_alg)

-02

*reformatted

-01

*introduced blinded claim names

*explained why JSON-encoding of values is needed

*explained merging algorithm ("processing model")

*generalized hash alg to digest derivation alg which also enables HMAC to calculate digests

*_sd_hash_alg renamed to sd_digest_derivation_alg

*Salt/Value Container (SVC) renamed to Issuer-Issued Disclosures (II-Disclosures)

*SD-JWT-Release (SD-JWT-R) renamed to Holder-Selected Disclosures (HS-Disclosures)

*sd_disclosure in II-Disclosures renamed to sd_ii_disclosures

*sd_disclosure in HS-Disclosures renamed to sd_hs_disclosures

*clarified relationship between sd_hs_disclosure and SD-JWT

*clarified combined formats for issuance and presentation

*clarified security requirements for blinded claim names

*improved description of Holder Binding security considerations - especially around the usage of "alg=none".

*updated examples

*text clarifications

*fixed cnf structure in examples

*added feature summary

-00

*Upload as draft-ietf-oauth-selective-disclosure-jwt-00

[[pre Working Group Adoption:]]

-02

*Added acknowledgements

*Improved Security Considerations

*Stressed entropy requirements for salts

*Python reference implementation clean-up and refactoring

*hash_alg renamed to _sd_hash_alg

-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

Brian Campbell
Ping Identity

Email: bcampbell@pingidentity.com