Workgroup: Web Authorization Protocol
Internet-Draft:
draft-ietf-oauth-selective-disclosure-jwt-01
Published: 24 October 2022
Intended Status: Standards Track
Expires: 27 April 2023
Authors: D. Fett   K. Yasuda
yes.com   Microsoft

Selective Disclosure for JWTs (SD-JWT)

Abstract

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

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

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, claim values are hidden, but cryptographically protected against undetected modification. When issuing the SD-JWT to the Holder, the Issuer also sends a JSON object that contains a mapping between hidden claim values and their cleartext counterparts, the so-called Disclosures. This JSON object is therefore called the Issuer-Issued Disclosures (II-Disclosures) object.

The Holder decides which claims to disclose to a Verifier. This specification defines a format for conveying the selected subset of the II-Disclosures to the Verifier. This subset is called the Holder-Selected Disclosures (HS-Disclosures) and is transported in a JWT, the HS-Disclosures JWT, for presentation alongside the SD-JWT. The Verifier can (and 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 HS-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.

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.

-Optionally, this specification allows to also hide ("blind") the claim names, not only the claim values.

-When claim names are blinded, this specification enables combining claims with blinded and unblinded names in the same SD-JWT.

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 cleartext claim value, a cleartext claim name, a salt, and optionally a blinded claim name value that is used to calculate a digest for a certain claim.

Issuer-Issued Disclosures Object (II-Disclosures Object) A JSON object created by the Issuer that contains Disclosures for all selectively-disclosable claims in an SD-JWT.

Holder-Selected Disclosures JWT (HS-Disclosures JWT) A JWT created by the Holder that contains the Disclosures from an Issuer-Issued Disclosures Object that the Holder is disclosing to the Verifier. In addition to the Disclosures, it can contain other properties and may be signed by the Holder.

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.

Claim Name Blinding Feature that enables to blind not only claim values, but also claim names of the claims that are included in

SD-JWT but are not disclosed to the Verifier in the HS-Disclosures JWT.

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 HS-Disclosures JWT.

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

```
           +------------+
           |            |
           |   Issuer   |
           |            |
           +------------+
                 |
             Issues SD-JWT
  and Issuer-Issued Disclosures Object
                 |
                 v
           +------------+
           |            |
           |   Holder   |
           |            |
           +------------+
                 |
           Presents SD-JWT
  and Holder-Selected Disclosures JWT
                 |
                 v
           +-------------+
           |             |+
           |  Verifiers  ||+
           |             |||
           +-------------+||
            +-------------+|
             +-------------+
```

Figure 1: SD-JWT Issuance and Presentation Flow

4. Concepts

In the following, the contents of SD-JWTs and HS-Disclosures JWTs 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 claim values with random 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 is an object with claim names (CLAIM-NAME) mapped to the digests over the claim values (CLAIM-VALUE) with random salts (SALT). Digests are calculated using a digest derivation function such as a hash function, HMAC, or other (DIGEST-DERIVATION()):

```
SD-CLAIMS = (
    CLAIM-NAME: DIGEST-DERIVATION(SALT, CLAIM-VALUE)
)*
```

When an HMAC or another type of derivation function is used for digest calculation, a secret cryptographic key or other cryptographic secret is used instead of a salt value. However, the term "salt" is used throughout this document for brevity.

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 a Holder-Selected Disclosures JWT

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

```
HOLDER-SELECTED-DISCLOSURES-DOC = (METADATA, SD-DISCLOSURES)
HOLDER-SELECTED-DISCLOSURES-JWT = HOLDER-SELECTED-DISCLOSURES-DOC
```

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

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

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

HOLDER-SELECTED-DISCLOSURES-JWT 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.

With Holder Binding, the HOLDER-SELECTED-DISCLOSURES-JWT is signed by the Holder using its private key. It therefore looks as follows:

```
HOLDER-SELECTED-DISCLOSURES = HOLDER-SELECTED-DISCLOSURES-DOC |
    SIG(HOLDER-SELECTED-DISCLOSURES-DOC, HOLDER-PRIV-KEY)
```

4.3.1. Optional Claim Name Blinding

If Claim Name Blinding is used, SD-CLAIMS is created as follows:

```
SD-CLAIMS = (
    CLAIM-NAME-PLACEHOLDER: DIGEST-DERIVATION(SALT,
        CLAIM-VALUE, CLAIM-NAME)
)*
```

CLAIM-NAME-PLACEHOLDER is a placeholder used instead of the original claim name, chosen such that it does not leak information about the claim name (e.g., randomly).

The contents of SD-DISCLOSURES are modified as follows:

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

Note that blinded and unblinded claim names can be mixed in SD-CLAIMS and accordingly in SD-DISCLOSURES.

4.4. Verifying a Holder-Selected Disclosures JWT

A Verifier checks that

* for each claim in HOLDER-SELECTED-DISCLOSURES, the digest over the disclosed values matches the digest under the given claim name in SD-JWT,

* if Holder Binding is used, the HOLDER-SELECTED-DISCLOSURES was signed by the private key belonging to HOLDER-PUBLIC-KEY.

The detailed algorithm is described in Section 6.2.

5. Data Formats

This section defines data formats for SD-JWT (containing digests of the salted claim values), Issuer-Issued Disclosures (containing the mapping of the plain-text claim values and the salt values), and HS-Disclosures (containing a subset of the same mapping).

5.1. The Challenge of Canonicalization

When receiving an SD-JWT with associated HS-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 derivation 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, 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 "M&#246;bius" 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 encoding: 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 digest 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 check the digest over 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 encoding approach is used, as it allows for simple and reliable interoperability without the requirement for a canonicalization library. To encode 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, as it is human-readable and used in JWTs as well. This approach means that SD-JWTs can be implemented purely based on widely available JWT and JSON encoding and decoding libraries.

To produce a source string to compute a digest, the data is put into a JSON object together with the salt value, like so (non-normative example, see Section 5.2.1 for details):

```
{"s": "6qMQvRL5haj", "v": "Möbius"}
```

Or, for the address example above:

```
{"s": "al1N3Zom221", "v":
  {"locality": "Schulpforta", "street_address": "Schulstr. 12"}}
```

(Line break and indentation of the second line for presentation only!)

This object is then JSON-encoded and used as the source string. The JSON-encoded value is transferred in the HS-Disclosures instead of the original JSON data:

```
"family_name": "{\"s\": \"6qMQvRL5haj\", \"v\": \"M\\u00f6bius\"}"
```

Or, for the address example:

```
"address": "{\"s\": \"al1N3Zom221\", \"v\":
  {\"locality\": \"Schulpforta\",
  \"street_address\": \"Schulstr. 12\"}}"
```

(Line break and indentation of the second and third line for presentation only!)

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.

Since the encoding is based on JSON, all value types that are allowed in JSON are also allowed in the v property in the source string. This includes numbers, strings, booleans, arrays, and objects.

It is important to note that the HS-Disclosures object containing the source string is neither intended nor suitable for direct consumption by an application that needs to access the disclosed claim values. The HS-Disclosures object is 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 6.3 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_digests and sd_digest_derivation_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.

5.2.1. sd_digests Claim (Digests of Selectively Disclosable Claims)

The property sd_digests MUST be used by the Issuer to include digests of the salted claim values for any claim that is intended to be selectively disclosable.

The Issuer MUST choose a new, cryptographically random salt value for each claim value. The salt value MUST then be encoded as a string. It is RECOMMENDED to base64url-encode the salt value.

The Issuer MUST generate the digests over a JSON literal according to [RFC8259] that is formed by JSON-encoding an object with the following contents:

* REQUIRED with the key s: the salt value,

* REQUIRED with the key v: the claim value (either a string or a more complex object, e.g., for the [OIDC] address claim),

* OPTIONAL, with the key n: the claim name (if Claim Name Blinding is to be used for this claim).

The following is an example for a JSON literal without Claim Name Blinding:

```
{"s": "6qMQvRL5haj", "v": "Peter"}
```

The following is an example for a JSON literal with Claim Name Blinding:

```
{"s": "6qMQvRL5haj", "v": "Peter", "n": "given_name"}
```

The sd_digests claim contains an object where claim names are mapped to the respective digests. If a claim name is to be blinded, the digests MUST contain the n key as described above and the claim name in sd_digests MUST be replaced by a placeholder name that does not leak information about the claim's original name. The same placeholder name will be used in the II-Disclosures (sd_ii_disclosures) and HS-Disclosures (sd_hs_disclosures) described below.

To this end, the Issuer MUST choose a random placeholder name for each claim that is to be blinded. It is RECOMMENDED to do so by base64url-encoding a cryptographically secure nonce. See Section 7.7 for further requirements.

5.2.1.1. Flat and Structured sd_digests objects

The sd_digests object can be a 'flat' object, directly containing all claim names and digests without any deeper structure. The sd_digests object can also be a 'structured' object, where some claims and their respective digests 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 2a 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 OpenID Connect for Identity Assurance [OIDC.IDA].

5.2.2. Digest Derivation Function Claim

The claim sd_digest_derivation_alg indicates the digest derivation algorithm used by the Issuer to generate the digests over the salts and the claim values.

The digest derivation algorithm identifier MUST be one of the following:

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

* an HMAC algorithm value from the "Algorithmn Name" column in the IANA "JSON Web Signature and Encryption Algorithms" registry [IANA.JWS.Algorithms]

* a value defined in another specification and/or profile of this specification

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

See Section 7 for requirements regarding entropy of the salt, minimum length of the salt, and choice of a digest derivation 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 and Example 2a 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",
  "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"
    }
  },
  "iat": 1516239022,
  "exp": 1516247022,
  "sd_digest_derivation_alg": "sha-256",
  "sd_digests": {
    "sub": "2EDXXZ1JcE6aTcM70fZopFneYAS9-hY3lalaoLuWD1s",
    "given_name": "pC56LWpTgec18Ll1kps3koXapnw6SOiI0d1ba34t-mY",
    "family_name": "EySQc316Ln3ZGJXwioELWSyylm_6OXV6rcL6LyPb7oI",
    "email": "qHv6gGaq4oFmIXyKh9ZlFjQ5rOClS-dXHiPMZyl2FaU",
    "phone_number": "jhr_PsauT4xsYZS_OxBW8y_1MLULOovKseRvF9CE0TM",
    "address": "eQXgmowqkT_ORkedoqeW0wBUy4vzkWG1VhvOjh3tl_o",
    "birthdate": "qgDxFuNpf83MkKe4GCaiLuL_XZdzO4pYD7lQKbv4zos"
  }
}
```

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.

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

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc
3N1ZXIiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJSU0EiLCAibiI6ICJwbTRiT0hCZ
y1vWWhBeVBXelI1NkFXWDNyVUlYcDExX0lDRGtHZ1M2VzNaV0x0cy1oendJM3g2NTY1O
WtnNGhWbzlkYkdvQ0pFM1pHRl9lYWV0RTMwVWhCVUVncEd3ckRyUWlKOXpxcHJtY0Zmc
jNxdnZrR2p0dGg4WmdsMWVNMmJKY093RTdQQ0JIV1RLV1lzMTUyUjdnNkpnMk9WcGgtY
ThycS1xNzlNaEtHNVFvV19tVHoxMFFUXzZINGM3UGpXRzFmamg4aHBXTm5iUF9wdjZkM
XpTd1pmYzVmbDZ5VlJMMERWMFYzbEdIS2UyV3FmX2VOR2pCckJMVmtsRFRrOC1zdFhfT
VdMY1ItRUdtWEFPdjBVQldpdFNfZFhKS0p1LXZYSnl3MTRuSFNHdXhUSUsyaHgxcHR0T
WZ0OUNzdnFpbVhLZURUVTE0cVFMMWVFN2loY3ciLCAiZSI6ICJBUUFCIn19LCAiaWF0I
jogMTUxNjIzOTAyMiwgImV4cCI6IDE1MTYyNDcwMjIsICJzZF9kaWdlc3RfZGVyaXZhd
Glvbl9hbGciOiAic2hhLTI1NiIsICJzZF9kaWdlc3RzIjogeyJzdWIiOiAiMkVEWFhaM
UpjRTZhVGNNNzBmWm9wRm5lWUFTOS1oWTNsYWxhb0x1V0QxcyIsICJnaXZlbl9uYW1lI
jogInBDNTZMV3BUZ2VjMThMbDFrcHMza29YYXBudzZTT2lJMGQxYmEzNHQtbVkiLCAiZ
mFtaWx5X25hbWUiOiAiRXlTUWMzMTZMbjNaR0pYd2lvRUxXU3l5bG1fNk9YVjZyY0w2T
HlQYjdvSSIsICJlbWFpbCI6ICJxSHY2Z0dhcTRvRm1JWHlLaDlabEZqUTVyT0NsUy1kW
EhpUE1aeWwyRmFVIiwgInBob25lX251bWJlciI6ICJqaHJfUHNhdVQ0eHNZWlNfT3hCV
zh5XzFNTFVMT292S3NlUnZGOUNFMFRNIiwgImFkZHJlc3MiOiAiZVFYZ21vd3FrVF9PU
mtlZG9xZVcwd0JVeTR2emtXRzFWaHZPamgzdGxfbyIsICJiaXJ0aGRhdGUiOiAicWdEe
EZ1TnBmODNNa0tlNEdDYWlMdUxfWFpkek80cFlEN2xRS2J2NHpvcyJ9fQ.0w8PQ_tg2K
6Q82XhXn3-Nmi7uGeXkOFFMSfp_8iMKRRlfg-HXXdoZWv8UECv1B2PIJITjH2RAz_egY
j-dLkPopnJ-0vIDKjKhvMCIIo0FEnTV3qQct-8s6NifR2exU1TuyF66Z9Jekk1V3M4Bn
KxCc6-mEf7_d1K-EfQ34dI-6XJFh05s1_sE7ePFvLRGtj4tHHQlwWGm7wQJqPRYtA_F0
N10jIlyFbw4B6T59TpI8ZjHgucCxF9p1IUb-RYb6P1dYF4sVdQT258jAJVCAPz62JoRn
-cPPwV-QbpAKD7npkk7pTxkYg0T9_iyvMcq_RdXGqqANkJn8qxEffwp_OsgA
```

5.4. Format of an Issuer-Issued Disclosures Object

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. 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.

An Issuer-Issued Disclosures Object (II-Disclosures Object) is a JSON object containing at least the top-level property sd_ii_disclosures. Its structure mirrors the one of sd_digests in the SD-JWT, but the values are the inputs to the digest calculations the Issuer used (the Disclosures), as strings.

The II-Disclosures Object MAY contain further properties, for example, to transport the Holder private key.

5.5. Example: Issuer-Issued Disclosures Object for the Flat SD-JWT in Example 1

The II-Disclosures Object for Example 1 is as follows:

```
{
  "sd_ii_disclosures": {
    "sub": "{\"s\": \"YZSmzeu7lFHUbZ8Z1QqH9Q\", \"v\":
      \"6c5c0a49-b589-431d-bae7-219122a9ec2c\"}",
    "given_name": "{\"s\": \"kHHp91-tAZt8m9E4Jl4XbQ\", \"v\":
      \"John\"}",
    "family_name": "{\"s\": \"PjIqpGWl4eB4QroDhqQw0w\", \"v\":
      \"Doe\"}",
    "email": "{\"s\": \"QRamZSB5Ky0MeJyz4EAleA\", \"v\":
      \"johndoe@example.com\"}",
    "phone_number": "{\"s\": \"xniP4JZtNWIH-Lk_Dt-o-A\", \"v\":
      \"+1-202-555-0101\"}",
    "address": "{\"s\": \"KtfsxxTm2mw0YLUcKZU8tA\", \"v\":
      {\"street_address\": \"123 Main St\", \"locality\":
      \"Anytown\", \"region\": \"Anystate\", \"country\": \"US\"}}",
    "birthdate": "{\"s\": \"Ozd4wBLBwqGzJhJvTmQwdQ\", \"v\":
      \"1940-01-01\"}"
  }
}
```

Important: As described in Section 5.1, digests are calculated over the JSON literal formed by serializing an object containing the salt, the claim value, and optionally the claim name. This ensures that the Issuer and Verifier use the same input to their digest derivation algorithms and avoids issues with canonicalization of JSON values that would lead to different digests. The II-Disclosures Object therefore maps claim names to JSON-encoded arrays.

5.6. Combined Format for Issuance

For transporting the II-Disclosures Object together with the SD-JWT from the Issuer to the Holder, the II-Disclosures Object is base64url-encoded and appended to the SD-JWT using a period character . as the separator. This means that the resulting string consists of four dot-separated parts as follows:

```
<SD-JWT Header>
.
<SD-JWT Payload>
.
<SD-JWT Signature>
.
<II-Disclosures>
```

(Line breaks for presentation only.)

This is called the Combined Format for Issuance.

The II-Disclosures Object and SD-JWT are implicitly linked through the digest values of the claims in the II-Disclosures Object that is included in the SD-JWT. To ensure that the correct II-Disclosures Object and SD-JWT pairings are being used, the Holder SHOULD verify the binding between II-Disclosures Object and SD-JWT as defined in Section 6.1.

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

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZU
kF6VmRsZnhOMjgwTmdIYUEifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N
1ZXIiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJSU0EiLCAibiI6ICJwbTRiT0hCZy1v
WWhBeVBXelI1NkFXWDNyVUlYcDExX0lDRGtHZ1M2VzNaV0x0cy1oendJM3g2NTY1OWtnN
GhWbzlkYkdvQ0pFM1pHRl9lYWV0RTMwVWhCVUVncEd3ckRyUWlKOXpxcHJtY0ZmcjNxdn
ZrR2p0dGg4WmdsMWVNMmJKY093RTdQQ0JIV1RLV1lzMTUyUjdnNkpnMk9WcGgtYThycS1
xNzlNaEtHNVFvV19tVHoxMFFUXzZINGM3UGpXRzFmamg4aHBXTm5iUF9wdjZkMXpTd1pm
YzVmbDZ5VlJMMERWMFYzbEdIS2UyV3FmX2VOR2pCckJMVmtsRFRrOC1zdFhfTVdMY1ItR
UdtWEFPdjBVQldpdFNfZFhKS0p1LXZYSnl3MTRuSFNHdXhUSUsyaHgxcHR0TWZ0OUNzdn
FpbVhLZURUVTE0cVFMMWVFN2loY3ciLCAiZSI6ICJBUUFCIn19LCAiaWF0IjogMTUxNjI
zOTAyMiwgImV4cCI6IDE1MTYyNDcwMjIsICJzZF9oYXNoX2FsZyI6ICJzaGEtMjU2Iiwg
InNkX2RpZ2VzdHMiOiB7InN1YiI6ICJPTWR3a2sySFB1aUluUHlwV1VXTXhvdDFZMnRTd
EdzTHVJY0RNaktkWE1VIiwgImdpdmVuX25hbWUiOiAiQWZLS0g0YTBJWmtpOE1GRHl0aE
ZhRlNfWHF6bi13UnZBTWZpeV9WallwRSIsICJmYW1pbHlfbmFtZSI6ICJlVW1YbXJ5MzJ
KaUtfNzZ4TWFzYWdrQVFRc21TVmRXNTdBamsxOHJpU0YwIiwgImVtYWlsIjogIi1SY3I0
ZkR5andsTV9pdGNNeG9RWkNFMVFBRXd5TEpjaWJFcEgxMTRLaUUiLCAicGhvbmVfbnVtY
mVyIjogIkp2Mm53MEMxd1A1QVN1dFlOQXhyV0VuYURSSXBpRjBlVFVBa1VPcDhGNlkiLC
AiYWRkcmVzcyI6ICJacmpLcy1SbUVBVmVBWVN6U3c2R1BGck1wY2djdENmYUo2dDlxUWh
iZko0IiwgImJpcnRoZGF0ZSI6ICJxWFBSUlBkcE5hZWJQOGp0YkVwTy1za0Y0bjd2N0FT
VGg4b0xnMG1rQWRRIn19.QgoJn9wkjFvM9bAr0hTDHLspuqdA21WzfBRVHkASa2ck4PFD
3TC9MiZSi3AiRytRbYT4ZzvkH3BSbm6vy68y62gj0A6OYvZ1Z60Wxho14bxZQveJZgw3u
_lMvYj6GKiUtskypFEHU-Kd-LoDVqEpf6lPQHdpsac__yQ_JL24oCEBlVQRXB-T-6ZNZf
ID6JafSkNNCYQbI8nXbzIEp1LBFm0fE8eUd4G4yPYOj1SeuR6Gy92T0vAoL5QtpIAHo49
oAmiSIj6DQNl2cNYs74jhrBIcNZyt4l8H1lV20wS5OS3T0vXaYD13fgm0p4iWD9cVg3HK
ShUVulEyrSbq94jIKg.eyJzZF9yZWxlYXNlIjogeyJzdWIiOiAie1wic1wiOiBcIjJHTE
M0MnNLUXZlQ2ZHZnJ5TlJOOXdcIiwgXCJ2XCI6IFwiNmM1YzBhNDktYjU4OS00MzFkLWJ
hZTctMjE5MTIyYTllYzJjXCJ9IiwgImdpdmVuX25hbWUiOiAie1wic1wiOiBcIjZJajd0
TS1hNWlWUEdib1M1dG12VkFcIiwgXCJ2XCI6IFwiSm9oblwifSIsICJmYW1pbHlfbmFtZ
SI6ICJ7XCJzXCI6IFwiUWdfTzY0enFBeGU0MTJhMTA4aXJvQVwiLCBcInZcIjogXCJEb2
VcIn0iLCAiZW1haWwiOiAie1wic1wiOiBcIlBjMzNKTTJMY2hjVV9sSGdndl91ZlFcIiw
gXCJ2XCI6IFwiam9obmRvZUBleGFtcGxlLmNvbVwifSIsICJwaG9uZV9udW1iZXIiOiAi
e1wic1wiOiBcImxrbHhGNWpNWWxHVFBVb3ZNTkl2Q0FcIiwgXCJ2XCI6IFwiKzEtMjAyL
TU1NS0wMTAxXCJ9IiwgImFkZHJlc3MiOiAie1wic1wiOiBcIjViUHMxSXF1Wk5hMGhrYU
Z6enpaTndcIiwgXCJ2XCI6IHtcInN0cmVldF9hZGRyZXNzXCI6IFwiMTIzIE1haW4gU3R
cIiwgXCJsb2NhbGl0eVwiOiBcIkFueXRvd25cIiwgXCJyZWdpb25cIjogXCJBbnlzdGF0
ZVwiLCBcImNvdW50cnlcIjogXCJVU1wifX0iLCAiYmlydGhkYXRlIjogIntcInNcIjogX
CJ5MXNWVTV3ZGZKYWhWZGd3UGdTN1JRXCIsIFwidlwiOiBcIjE5NDAtMDEtMDFcIn0ifX
0
```

(Line breaks for presentation only.)

5.7. Format of a Holder-Selected Disclosures JWT

The HS-Disclosures JWT contains the Disclosures of the claims the Holder has consented to disclose to the Verifier. This enables the Verifier to verify the claims received from the Holder by computing the digests of the claim values, salts, and potentially cleartext claim names revealed in the HS-Disclosures JWT using the digest derivation algorithm specified in SD-JWT and comparing them to the digests included in SD-JWT.

The Disclosures are contained in the sd_hs_disclosures object. The structure of the sd_hs_disclosures object in the HS-Disclosures JWT is the same as the structure of the sd_ii_disclosures object in the II-Disclosures Object, but any claims the Holder wishes not to disclose are omitted.

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

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

If Holder Binding is desired, the HS-Disclosures JWT 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.

Whether to check the signature of the HS-Disclosures JWT is up to the Verifier's policy, based on the set of trust requirements such as trust frameworks it belongs to. As described in Section 6.2, the Verifier MUST NOT accept HS-Disclosures JWTs using "none" algorithm, when the Verifier's policy requires a signed HS-Disclosures JWT. See also Section 7.6.

5.8. Example: Holder-Selected Disclosures JWT for Example 1

The following is a non-normative example of the contents of a HS-Disclosures JWT for Example 1:

```
{
  "nonce": "XZOUco1u_gEPknxS78sWWg",
  "aud": "https://example.com/verifier",
  "sd_hs_disclosures": {
    "given_name": "{\"s\": \"kHHp91-tAZt8m9E4Jl4XbQ\", \"v\":
      \"John\"}",
    "family_name": "{\"s\": \"PjIqpGWl4eB4QroDhqQw0w\", \"v\":
      \"Doe\"}",
    "address": "{\"s\": \"KtfsxxTm2mw0YLUcKZU8tA\", \"v\":
      {\"street_address\": \"123 Main St\", \"locality\":
      \"Anytown\", \"region\": \"Anystate\", \"country\": \"US\"}}"
  }
}
```

For each claim, a JSON literal that decodes to an object with the and the claim value (plus optionally the claim name) is contained in the sd_hs_disclosures object.

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

Below is a non-normative example of a representation of the HS-Disclosures JWT using JWS Compact Serialization:

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogIkxkeVRYd0F5ZnJpcjRfVjZORzFSYzEwVThKZ
ExZVHJFQktKaF9oNWlfclUifQ.eyJub25jZSI6ICJYWk9VY28xdV9nRVBrbnhTNzhzV1d
nIiwgImF1ZCI6ICJodHRwczovL2V4YW1wbGUuY29tL3ZlcmlmaWVyIiwgInNkX3JlbGVh
c2UiOiB7ImdpdmVuX25hbWUiOiAie1wic1wiOiBcIjZJajd0TS1hNWlWUEdib1M1dG12V
kFcIiwgXCJ2XCI6IFwiSm9oblwifSIsICJmYW1pbHlfbmFtZSI6ICJ7XCJzXCI6IFwiUW
dfTzY0enFBeGU0MTJhMTA4aXJvQVwiLCBcInZcIjogXCJEb2VcIn0iLCAiYWRkcmVzcyI
6ICJ7XCJzXCI6IFwiNWJQczFJcXVaTmEwaGthRnp6elpOd1wiLCBcInZcIjoge1wic3Ry
ZWV0X2FkZHJlc3NcIjogXCIxMjMgTWFpbiBTdFwiLCBcImxvY2FsaXR5XCI6IFwiQW55d
G93blwiLCBcInJlZ2lvblwiOiBcIkFueXN0YXRlXCIsIFwiY291bnRyeVwiOiBcIlVTXC
J9fSJ9fQ.fw4xRl7m1mDPCZvCTn3GOr2PgBZ--fTKfy7s-GuEifNvzW5KsJaBBFvzdZzt
m25XGhk29uw-XwEw00r0hyxXLBvWfA0XbDK3JBmdpOSW1bEyNBdSHPJoeq9Xyts2JN40v
JzU2UxNaLKDaEheWf3F_E52yhHxvMLNdvZJ9FksJdSMK6ZCyGfRJadPN2GhNltqph52sW
iFKUyUk_4RtwXmT_lF49tWOMZqtG-akN9wrBoMsleM0soA0BXIK10rG5cKZoSNr-u2luz
bdZx3CFdAenaqScIkluPPcrXBZGYyX2zYUbGQs2RRXnBmox_yl6CvLbb0qTTYhDnDEo_M
H-ZtWw
```

5.9. Combined Format for Presentation

The SD-JWT and the HS-Disclosures JWT can be combined into one document using period character . as a separator. This means that the resulting string consists of six dot-separated parts as described below.

The last part (HSD Signature) may be empty when Holder Binding is not used and HS-Disclosures JWT is not signed.

```
<SD-JWT Header>
.
<SD-JWT Payload>
.
<SD-JWT Signature>
.
<HSD Header>
.
<HSD Payload>
.
<HSD Signature?>
```

(Line breaks for presentation only.)

This is called the Combined Format for Presentation.

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

```
eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZU
kF6VmRsZnhOMjgwTmdIYUEifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N
1ZXIiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJSU0EiLCAibiI6ICJwbTRiT0hCZy1v
WWhBeVBXelI1NkFXWDNyVUlYcDExX0lDRGtHZ1M2VzNaV0x0cy1oendJM3g2NTY1OWtnN
GhWbzlkYkdvQ0pFM1pHRl9lYWV0RTMwVWhCVUVncEd3ckRyUWlKOXpxcHJtY0ZmcjNxdn
ZrR2p0dGg4WmdsMWVNMmJKY093RTdQQ0JIV1RLV1lzMTUyUjdnNkpnMk9WcGgtYThycS1
xNzlNaEtHNVFvV19tVHoxMFFUXzZINGM3UGpXRzFmamg4aHBXTm5iUF9wdjZkMXpTd1pm
YzVmbDZ5VlJMMERWMFYzbEdIS2UyV3FmX2VOR2pCckJMVmtsRFRrOC1zdFhfTVdMY1ItR
UdtWEFPdjBVQldpdFNfZFhKS0p1LXZYSnl3MTRuSFNHdXhUSUsyaHgxcHR0TWZ0OUNzdn
FpbVhLZURUVTE0cVFMMWVFN2loY3ciLCAiZSI6ICJBUUFCIn19LCAiaWF0IjogMTUxNjI
zOTAyMiwgImV4cCI6IDE1MTYyNDcwMjIsICJzZF9oYXNoX2FsZyI6ICJzaGEtMjU2Iiwg
InNkX2RpZ2VzdHMiOiB7InN1YiI6ICJPTWR3a2sySFB1aUluUHlwV1VXTXhvdDFZMnRTd
EdzTHVJY0RNaktkWE1VIiwgImdpdmVuX25hbWUiOiAiQWZLS0g0YTBJWmtpOE1GRHl0aE
ZhRlNfWHF6bi13UnZBTWZpeV9WallwRSIsICJmYW1pbHlfbmFtZSI6ICJlVW1YbXJ5MzJ
KaUtfNzZ4TWFzYWdrQVFRc21TVmRXNTdBamsxOHJpU0YwIiwgImVtYWlsIjogIi1SY3I0
ZkR5andsTV9pdGNNeG9RWkNFMVFBRXd5TEpjaWJFcEgxMTRLaUUiLCAicGhvbmVfbnVtY
mVyIjogIkp2Mm53MEMxd1A1QVN1dFlOQXhyV0VuYURSSXBpRjBlVFVBa1VPcDhGNlkiLC
AiYWRkcmVzcyI6ICJacmpLcy1SbUVBVmVBWVN6U3c2R1BGck1wY2djdENmYUo2dDlxUWh
iZko0IiwgImJpcnRoZGF0ZSI6ICJxWFBSUlBkcE5hZWJQOGp0YkVwTy1za0Y0bjd2N0FT
VGg4b0xnMG1rQWRRIn19.QgoJn9wkjFvM9bAr0hTDHLspuqdA21WzfBRVHkASa2ck4PFD
3TC9MiZSi3AiRytRbYT4ZzvkH3BSbm6vy68y62gj0A6OYvZ1Z60Wxho14bxZQveJZgw3u
_lMvYj6GKiUtskypFEHU-Kd-LoDVqEpf6lPQHdpsac__yQ_JL24oCEBlVQRXB-T-6ZNZf
ID6JafSkNNCYQbI8nXbzIEp1LBFm0fE8eUd4G4yPYOj1SeuR6Gy92T0vAoL5QtpIAHo49
oAmiSIj6DQNl2cNYs74jhrBIcNZyt4l8H1lV20wS5OS3T0vXaYD13fgm0p4iWD9cVg3HK
ShUVulEyrSbq94jIKg.eyJhbGciOiAiUlMyNTYiLCAia2lkIjogIkxkeVRYd0F5ZnJpcj
RfVjZORzFSYzEwVThKZExZVHJFQktKaF9oNWlfclUifQ.eyJub25jZSI6ICJYWk9VY28x
dV9nRVBrbnhTNzhzV1dnIiwgImF1ZCI6ICJodHRwczovL2V4YW1wbGUuY29tL3Zlcmlma
WVyIiwgInNkX3JlbGVhc2UiOiB7ImdpdmVuX25hbWUiOiAie1wic1wiOiBcIjZJajd0TS
1hNWlWUEdib1M1dG12VkFcIiwgXCJ2XCI6IFwiSm9oblwifSIsICJmYW1pbHlfbmFtZSI
6ICJ7XCJzXCI6IFwiUWdfTzY0enFBeGU0MTJhMTA4aXJvQVwiLCBcInZcIjogXCJEb2Vc
In0iLCAiYWRkcmVzcyI6ICJ7XCJzXCI6IFwiNWJQczFJcXVaTmEwaGthRnp6elpOd1wiL
CBcInZcIjoge1wic3RyZWV0X2FkZHJlc3NcIjogXCIxMjMgTWFpbiBTdFwiLCBcImxvY2
FsaXR5XCI6IFwiQW55dG93blwiLCBcInJlZ2lvblwiOiBcIkFueXN0YXRlXCIsIFwiY29
1bnRyeVwiOiBcIlVTXCJ9fSJ9fQ.fw4xRl7m1mDPCZvCTn3GOr2PgBZ--fTKfy7s-GuEi
fNvzW5KsJaBBFvzdZztm25XGhk29uw-XwEw00r0hyxXLBvWfA0XbDK3JBmdpOSW1bEyNB
dSHPJoeq9Xyts2JN40vJzU2UxNaLKDaEheWf3F_E52yhHxvMLNdvZJ9FksJdSMK6ZCyGf
RJadPN2GhNltqph52sWiFKUyUk_4RtwXmT_lF49tWOMZqtG-akN9wrBoMsleM0soA0BXI
K10rG5cKZoSNr-u2luzbdZx3CFdAenaqScIkluPPcrXBZGYyX2zYUbGQs2RRXnBmox_yl
6CvLbb0qTTYhDnDEo_MH-ZtWw
```

6. Verification and Processing

6.1. Verification by the Holder when Receiving SD-JWT and Issuer-Issued Disclosures Object

The Holder SHOULD verify the binding between SD-JWT and II-Disclosures Object by performing the following steps: 1. Check that all the claims in the II-Disclosures Object are present in the SD-JWT and that there are no claims in the SD-JWT that are not in the

II-Disclosures Object 2. Check that the digests of the claims in the II-Disclosures Object match those in the SD-JWT

6.2. Verification by the Verifier when Receiving SD-JWT and Holder-Selected Disclosures JWT

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

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 according to the Verifier's policy for the use case at hand. This decision MUST NOT be based on whether the HS-Disclosures JWT is signed or not. Refer to Section 7.6 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 HS-Disclosures JWT.

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. none 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.

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

    6. Check that the sd_digest_derivation_alg claim is present and its value is understood and the digest derivation algorithm is deemed secure.

5. Validate the HS-Disclosures JWT:

    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 HS-Disclosures JWT 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 HS-Disclosures 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 HS-Disclosures JWT.

    2. For each claim in sd_hs_disclosures in the HS-Disclosures JWT:

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

        4. Compute the base64url-encoded digest of the JSON literal disclosed by the Holder using the sd_digest_derivation_alg in SD-JWT.

        5. Compare the digests computed in the previous step with the one of the same claim in the SD-JWT. Accept the claim only when the two digests match.

        6. Ensure that the claim value in the HS-Disclosures JWT is a JSON-encoded object containing at least the keys s and v, and optionally n.

        7. Store the value of the key v as the claim value. If n is contained in the object, use the value of the key n as the claim name.

    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 disclosed.

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

6.3. Processing Model

Neither an SD-JWT nor an HS-Disclosures JWT is suitable for direct use by an application. Besides the REQUIRED verification steps listed above, it is further RECOMMENDED that an application-consumable format is generated from the data released in the HS-Disclosures. The RECOMMENDED way is to merge the released claims and any plaintext claims in the SD-JWT recursively:

* Objects from the released claims must be merged into existing objects from the SD-JWT.

* If a key is present in both objects:

    - If the value in the released claims is an object and the value in the SD-JWT claims is an object, the two objects MUST be merged recursively.

    - Else, the value in the released claims MUST be used.

The keys sd_digests and sd_digest_derivation_alg SHOULD be removed prior to further processing.

The processing is shown in Examples 2b and 3 in the Appendix.

7. Security Considerations

7.1. 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.

7.2. 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.

7.3. 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 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.

7.4. Minimum length of the salt

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

Note that minimum 128 bits would be necessary when SHA-256, HMAC-SHA256, or a function of similar strength is used, but a smaller salt size might achieve similar level of security if a stronger iterative derivation function is used.

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.

7.5. Choice of a digest derivation algorithm

For the security of this scheme, the digest derivation 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.

7.6. 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 the HS-Disclosures JWT is signed by 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 HS-Disclosure JWT is signed or not, as an attacker can remove the signature from any HS-Disclosure JWT 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 HS-Disclosure is unsigned or no recognized key reference is present in the SD-JWT, the Verifier will reject the presentation, as described in Section 6.2.

7.7. Blinding Claim Names

Issuers that chose to blind claim names MUST ensure not to inadvertently leak information about the blinded claim names to Verifiers.

It is RECOMMENDED to use cryptographically random numbers with at least 128 bits of entropy as placeholder claim names.

The order of elements in JSON-encoded objects is generally not relevant to applications, but it may reveal information about a blinded claim name to the verifier. For example, assume the following two clear-text claim sets created by the same Issuer:

(A)

```
{
  "given_name": "Doe",
  "secret_club_membership_no": 42
}
```

(B)

```
{
  "is_secret_agent": true,
  "given_name": "Doe"
}
```

When naively blinding the claim names, the order of the elements might be preserved in the SD-JWT (depending on implementation details of the programming language):

(A)

```
{
  "given_name": "Doe",
  "3DOgmo7w7MDZNh1Zjvmwpg":
    "OXZKGG7Ltar4vz_L7sAtWIkVXVf5r9xONFKZdyoNlco"
}
```

(B)

```
{
  "CwiB46IUgi4NydIfgGTRwg":
    "4miZg7O_JaidVJyjGiPpc4FXAMN16e1SBZfOMlYg3hQ",
  "given_name": "Doe"
}
```

A verifier, even if it does not learn any blinded claim names, can distinguish what claim name has been hidden just by observing the order of blinded and unblinded claim names. It is therefore RECOMMENDED, if at least one claim name is blinded, to either

* randomize the order of all claims (blinded/unblinded, selectively disclosed/not-selectively disclosed),

* or sort the claims by the property name (i.e., the placeholder claim name for blinded claim names and the plaintext claim name for unblinded claim names). The precise order does not matter. For example, ordering by unicode code points or by lexicographic order is sufficient to hide the original order of claims.

This applies to Issuers (SD-JWT and II-Disclosures document) and Holders (HS-Disclosures JWT).

With the approach chosen in this specification, claim names of objects that are not themselves selectively disclosable are not blinded. This can be seen in Example 6 in the Appendix, where even in the blinded SD-JWT, address and delivery_address are visible. This limitation needs to be taken into account by Issuers when creating the structure of the SD-JWT.

The Issuer MUST ensure that a new random placeholder name 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 6 in the Appendix, where multiple claims with same name appear below address and delivery_address, but each of them has a different blinded claim name. For each credential issued, new random placeholder names MUST be chosen by the Issuer.

8. Privacy Considerations

8.1. Claim Names

By default, claim names are not blinded in an SD-JWT. In this case, even when the claim's value is not known to a Verifier, the claim name can disclose some information to the Verifier. For example, if the SD-JWT contains a claim named super_secret_club_membership_no, the Verifier might assume that the end-user is a member of the Super Secret Club.

Blinding claim names can help to avoid this potential privacy issue. In many cases, however, Verifiers can already deduce this or similar information just from the identification of the Issuer and the schema used for the SD-JWT. Blinding claim names might not provide additional privacy if this is the case.

Furthermore, re-using the same value to blind a claim name may limit the privacy benefits.

8.2. 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.). More advanced cryptographic schemes, outside the scope of this specification, can be used to prevent this type of linkability.

9. Acknowledgements

We would like to thank Alen Horvat, Brian Campbell, Christian Paquin, Fabian Hauck, Giuseppe De Marco, Kushal Das, Mike Jones, Nat Sakimura, Pieter Kasselman, Shawn Butterfield, and Torsten

Lodderstedt 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.

10. IANA Considerations

TBD

11. 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>.

[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>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

12. Informative References

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

[IANA.JWS.Algorithms] IANA, "JSON Web Signature and Encryption Algorithms", <https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-algorithms>.

[OIDC] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1", 8 November 2014, <https://openid.net/specs/openid-connect-core-1_0.html>.

[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>.

[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 - Structured SD-JWT

This non-normative example is based on the same claim values as Example 1, but here the Issuer decided to create a structured object for the digests. This allows for the disclosure of individual members of the address claim separately.

```
{
  "iss": "https://example.com/issuer",
  "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"
    }
  },
  "iat": 1516239022,
  "exp": 1516247022,
  "sd_digest_derivation_alg": "sha-256",
  "sd_digests": {
    "sub": "p7GDm8_lnxCJUsQojBatCJQgPCZOVBGxU-eX_lUIcC4",
    "given_name": "BrmUer7nGIRyk3sbHHcZk43M9Oy_BQar0VE3NMOGk9w",
    "family_name": "8voOnlh20GGzTInd6T9-Vcu2l6Q4_Kc-keedo7_3VY8",
    "email": "b9DpmK8_xwhR4PX_MiIsQc1TyB_1NN40lI5Kj8SSNl4",
    "phone_number": "0LFRbHdtG1eze9ET1rDEtSIrPI0poCM3J0EYBt2iwVg",
    "address": {
      "street_address":
        "qYDFWJxdl_OQDdn_lxX1-E9r5H2juwqonoWM8A76X_w",
      "locality": "3mLauig0JJyjJbdMvf3jLJGSBAIt0tdvq7F_VL1gqXw",
      "region": "qRa_XKvVxCzUK8buAsxg9ylzyQlfvUgSwqATQV74z6c",
      "country": "DjbYtjTT3PAQHtVkcpvrnRboYVUfXMro6Y4oEGdHW_0"
    },
    "birthdate": "rXv8RpBXYOy9WtYf2Bg-KIdO0a3KnYGCAhL53iCsLJA"
  }
}
```

The II-Disclosures Object for this SD-JWT is as follows:

```
{
  "sd_ii_disclosures": {
    "sub": "{\"s\": \"2iFrkb5skOft_gSL6BhdBg\", \"v\":
      \"6c5c0a49-b589-431d-bae7-219122a9ec2c\"}",
    "given_name": "{\"s\": \"AbA1MKJ1Oyqtff2JoFKNXA\", \"v\":
      \"John\"}",
    "family_name": "{\"s\": \"vGk9hg40yrI1qazJn8qaKw\", \"v\":
      \"Doe\"}",
    "email": "{\"s\": \"6Ilb1QXTN4Qdv-1qGcQdbw\", \"v\":
      \"johndoe@example.com\"}",
    "phone_number": "{\"s\": \"-F5a6ZAOKHwUsYPDS383pQ\", \"v\":
      \"+1-202-555-0101\"}",
    "address": {
      "street_address": "{\"s\": \"t6GqrdbiTFbJYh4D38aLjA\", \"v\":
        \"123 Main St\"}",
      "locality": "{\"s\": \"B0G5ap7hsAPIYOJ21rUjgg\", \"v\":
        \"Anytown\"}",
      "region": "{\"s\": \"YTPF0rUHYtvldv1Df63WXQ\", \"v\":
        \"Anystate\"}",
      "country": "{\"s\": \"mVZ4hCTnVdpu_GN-Rb9wNw\", \"v\":
        \"US\"}"
    },
    "birthdate": "{\"s\": \"T6-5A3xYsyy2MnwnUWbW3w\", \"v\":
      \"1940-01-01\"}"
  }
}
```

An HS-Disclosures JWT for the SD-JWT above that discloses only region and country of the address property could look as follows:

```
{
  "nonce": "XZOUco1u_gEPknxS78sWWg",
  "aud": "https://example.com/verifier",
  "sd_hs_disclosures": {
    "given_name": "{\"s\": \"AbA1MKJ1Oyqtff2JoFKNXA\", \"v\":
      \"John\"}",
    "family_name": "{\"s\": \"vGk9hg40yrI1qazJn8qaKw\", \"v\":
      \"Doe\"}",
    "birthdate": "{\"s\": \"T6-5A3xYsyy2MnwnUWbW3w\", \"v\":
      \"1940-01-01\"}",
    "address": {
      "region": "{\"s\": \"YTPF0rUHYtvldv1Df63WXQ\", \"v\":
        \"Anystate\"}",
      "country": "{\"s\": \"mVZ4hCTnVdpu_GN-Rb9wNw\", \"v\":
        \"US\"}"
    }
  }
}
```

A.2. Example 2b - Mixing SD and Non-SD Claims

In this example, a variant of Example 2a, the Issuer decided to apply selective disclosure only to some of the claims. In particular, the country component of the address is contained in the JWT as a regular claim, whereas the rest of the claims can be disclosed selectively. Note that the processing model described in Section 6.3 allows for merging the selectively disclosable claims with the regular claims.

The JSON-payload of the SD-JWT that contains both selectively disclosable claims in the sd_digests object and not-selectively disclosable claims in a top-level JWT claim would look as follows:

```
{
  "iss": "https://example.com/issuer",
  "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"
    }
  },
  "iat": 1516239022,
  "exp": 1516247022,
  "sd_digest_derivation_alg": "sha-256",
  "sd_digests": {
    "sub": "m6f849XozrOu1dDvaoGfzp_FwJ0Jpcm8LBt8BeZdxkc",
    "given_name": "CEBrXkrUZcZ3njZE46q_CEdSASdcEP0qoGrjNcPJx8g",
    "family_name": "j5ZcRWCSTbdtevKIp8L1XMunNHXZHOEDLtkJ3By4rms",
    "email": "AXm5JzGxUAfQaqTAz5hZGrhL7ZEM_J3ljKRK4wSpRvU",
    "phone_number": "Vkehj3w1-X9Ssz96tWl8lvap8EaIy9pi9q4qWzWAWNo",
    "address": {
      "street_address":
        "MVldFr-b-NKmQSLyHbnnq9ciMFGcb4GuhLtKLtmmnwk",
      "locality": "aAusTIjJS8e9QwaGs53OaHqngMZ142uDScfW41hqFm0",
      "region": "o6d8Kv-xOL3fidw5t0QF1StAlw5YLSN3Rco1aHsiWn8"
    },
    "birthdate": "_l2Sr5D08premyjfkmrnxMV6aFnEH8qMXme0BFGFGqk"
  },
  "address": {
    "country": "US"
  }
}
```

The Holder can now, for example, release the rest of the components of the address claim in the HS-Disclosures:

```
{
  "nonce": "XZOUco1u_gEPknxS78sWWg",
  "aud": "https://example.com/verifier",
  "sd_hs_disclosures": {
    "given_name": "{\"s\": \"juf0vRMI_5aHaGZQfl5o5A\", \"v\":
      \"John\"}",
    "family_name": "{\"s\": \"mJXFsX6E6IvqR6vMd_un5A\", \"v\":
      \"Doe\"}",
    "birthdate": "{\"s\": \"vnc31gtRYVh_zW8RrqSbaw\", \"v\":
      \"1940-01-01\"}",
    "address": {
      "region": "{\"s\": \"4mt7paa9SIEuEgWIm-10kg\", \"v\":
        \"Anystate\"}",
      "street_address": "{\"s\": \"4r1Y7ivPIQzkp8rKF_BUTQ\", \"v\":
        \"123 Main St\"}",
      "locality": "{\"s\": \"v5I5nfxYin0IB2mWP1oj6Q\", \"v\":
        \"Anytown\"}"
    }
  }
}
```

The Verifier, after verifying the SD-JWT and applying the HS-Disclosures, would process the result according to Section 6.3 and pass the following data to the application:

```
{
  "given_name": "John",
  "family_name": "Doe",
  "birthdate": "1940-01-01",
  "address": {
    "region": "Anystate",
    "street_address": "123 Main St",
    "locality": "Anytown",
    "country": "US"
  },
  "iss": "https://example.com/issuer",
  "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"
    }
  },
  "iat": 1516239022,
  "exp": 1516247022
}
```

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

In this example, a complex object such as those defined in OIDC4IDA [OIDC.IDA] is used. Here, 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": "Meier",
      "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 Issuer in this example further adds the two claims birthdate and place_of_birth to the claims element in plain text. The following shows the resulting SD-JWT payload:

```
{
  "iss": "https://example.com/issuer",
  "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"
    }
  },
  "iat": 1516239022,
  "exp": 1516247022,
  "sd_digest_derivation_alg": "sha-256",
  "sd_digests": {
    "verified_claims": {
      "verification": {
        "trust_framework":
          "fkIW-4iUZgTeIeDg_Z_6oFHU-wyWwazSpuaiQbc5QKw",
        "time": "VRF-G_LfTzSaYkLelVzry82l1zQxGwk1RfGcnUUWukc",
        "verification_process":
          "9OpDml4eRBM6Usfk3MF2i7kBl1xGGkzPq5Ncs1mvbPo",
        "evidence": [
          {
            "type": "HucanHhQwb-TJNg_rVpaonNSDtzPrCEebb3LfXTuLSM",
            "method": "aU7IO7ooT8vArMkqpOfkIAlKw8BNcfRyw3NXs3ZS128",
            "time": "LHcH98bV3-ZNUa00HNnqOf8W5IdijY1aEnpVzDNVBwA",
            "document": {
              "type": "3ITIlfkbUI0NveviEJBw-_VEaGiPtCDcXy9uD9orWFA",
              "issuer": {
                "name":
                  "AY7wW63Vbcd7RnKDb39sSXpLgyiVNxWgoRnV6xZD5C8",
                "country":
                  "Kd3aUmm6XHjpWp6OYiJeEZUrD5J7nIRU3SlTc-E53gs"
              },
              "number":
                "8gKpksl66fN9F2Zxs1PRPgD8kHi8dGC2JzpqtrPZavs",
              "date_of_issuance":
                "GfIEhOGWwe8J7lx6HSAPpC-Qvx0ihwWkEE0_LZ-r_DI",
              "date_of_expiry":
                "_fdljKRdp5wptGi7DwKNZEsSX6AnniVqmDE0aSznH74"
            }
          }
        ]
      },
      "claims": {
        "given_name": "sx4wGd6-ONAsiq7dN16GHeg4RAyOshRBdoXWE_E751w",
        "family_name":
          "Ldbea0SibAQDiZJlBigptwWXZ9QA8a0dKK7jipSn2K8",
        "nationalities":
          "tr8SXHdYS0rzAio_IhFp2lzlta4kDzKCM7hUxItCU2U",
        "address": {
          "locality": "VFgKHPXnNrZHeoBwcu61b5VCoFVX0rQjtH5aOiiLz0E",
          "postal_code":
            "G8XHi8sCPc45WATery6RSvnEcdypnrjypjBl4LBd5YE",
          "country": "YyG4Nhyfjitpo6-yMDRTARSVAnZNvkYqRY3XepoQ_j8",
          "street_address":
            "NwAKfAtjQcN_XbV3kuHt3gbUMvQ83n02C1EexI9Ro2A"
        }
      }
    },
    "birth_middle_name":
      "M5GhkvNcGjGONRey2pRORuL2yCfYz5jo0XqF6K0tUWk",
    "salutation": "8m0-sBNA8I88_LDc05C7gE31pTm_CXQfewiwlL1Sn1Y",
    "msisdn": "dLQVMDIkEHnmPVvuHNYiv7WwAqGE7mbyJMh5EfbjM1Q"
  },
  "verified_claims": {
    "claims": {
      "birthdate": "1956-01-28",
      "place_of_birth": {
        "country": "DE",
        "locality": "Musterstadt"
      }
    }
  }
}
```

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

eyJhbGciOiAiUlMyNTYiLCAia2lkIjogImNBRUlVcUowY21MekQxa3pHemhlaUJhZzBZ
UkF6VmRsZnhOMjgwTmdIYUEifQ.eyJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc
3N1ZXIiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJSU0EiLCAibiI6ICJwbTRiT0hCZ
y1vWWhBeVBXelI1NkFXWDNyVUlYcDExX0lDRGtHZ1M2VzNaV0x0cy1oendJM3g2NTY1O
WtnNGhWbzlkYkdvQ0pFM1pHRl9lYWV0RTMwVWhCVUVncEd3ckRyUWlKOXpxcHJtY0Zmc
jNxdnZrR2p0dGg4WmdsMWVNMmJKY093RTdQQ0JIV1RLV1lzMTUyUjdnNkpnMk9WcGgtY
ThycS1xNzlNaEtHNVFvV19tVHoxMFFUXzZINGM3UGpXRzFmamg4aHBXTm5iUF9wdjZkM
XpTd1pmYzVmbDZ5VlJMMERWMFYzbEdIS2UyV3FmX2VOR2pCckJMVmtsRFRrOC1zdFhfT
VdMY1ItRUdtWEFPdjBVQldpdFNfZFhKS0p1LXZYSnl3MTRuSFNHdXhUSUsyaHgxcHR0T
WZ0OUNzdnFpbVhLZURUVTE0cVFMMWVFN2loY3ciLCAiZSI6ICJBUUFCIn19LCAiaWF0I
jogMTUxNjIzOTAyMiwgImV4cCI6IDE1MTYyNDcwMjIsICJzZF9kaWdlc3RfZGVyaXZhd
Glvbl9hbGciOiAic2hhLTI1NiIsICJzZF9kaWdlc3RzIjogeyJ2ZXJpZmllZF9jbGFpb
XMiOiB7InZlcmlmaWNhdGlvbiI6IHsidHJ1c3RfZnJhbWV3b3JrIjogImZrSVctNGlVW
mdUZUllRGdfWl82b0ZIVS13eVd3YXpTcHVhaVFiYzVRS3ciLCAidGltZSI6ICJWUkYtR
19MZlR6U2FZa0xlbFZ6cnk4MmwxelF4R3drMVJmR2NuVVVXdWtjIiwgInZlcmlmaWNhd
Glvbl9wcm9jZXNzIjogIjlPcERtbDRlUkJNNlVzZmszTUYyaTdrQmwxeEdHa3pQcTVOY
3MxbXZiUG8iLCAiZXZpZGVuY2UiOiBbeyJ0eXBlIjogIkh1Y2FuSGhRd2ItVEpOZ19yV
nBhb25OU0R0elByQ0VlYmIzTGZYVHVMU00iLCAibWV0aG9kIjogImFVN0lPN29vVDh2Q
XJNa3FwT2ZrSUFsS3c4Qk5jZlJ5dzNOWHMzWlMxMjgiLCAidGltZSI6ICJMSGNIOThiV
jMtWk5VYTAwSE5ucU9mOFc1SWRpalkxYUVucFZ6RE5WQndBIiwgImRvY3VtZW50Ijoge
yJ0eXBlIjogIjNJVElsZmtiVUkwTnZldmlFSkJ3LV9WRWFHaVB0Q0RjWHk5dUQ5b3JXR
kEiLCAiaXNzdWVyIjogeyJuYW1lIjogIkFZN3dXNjNWYmNkN1JuS0RiMzlzU1hwTGd5a
VZOeFdnb1JuVjZ4WkQ1QzgiLCAiY291bnRyeSI6ICJLZDNhVW1tNlhIanBXcDZPWWlKZ
UVaVXJENUo3bklSVTNTbFRjLUU1M2dzIn0sICJudW1iZXIiOiAiOGdLcGtzbDY2Zk45R
jJaeHMxUFJQZ0Q4a0hpOGRHQzJKenBxdHJQWmF2cyIsICJkYXRlX29mX2lzc3VhbmNlI
jogIkdmSUVoT0dXd2U4SjdseDZIU0FQcEMtUXZ4MGlod1drRUUwX0xaLXJfREkiLCAiZ
GF0ZV9vZl9leHBpcnkiOiAiX2ZkbGpLUmRwNXdwdEdpN0R3S05aRXNTWDZBbm5pVnFtR
EUwYVN6bkg3NCJ9fV19LCAiY2xhaW1zIjogeyJnaXZlbl9uYW1lIjogInN4NHdHZDYtT
05Bc2lxN2ROMTZHSGVnNFJBeU9zaFJCZG9YV0VfRTc1MXciLCAiZmFtaWx5X25hbWUiO
iAiTGRiZWEwU2liQVFEaVpKbEJpZ3B0d1dYWjlRQThhMGRLSzdqaXBTbjJLOCIsICJuY
XRpb25hbGl0aWVzIjogInRyOFNYSGRZUzByekFpb19JaEZwMmx6bHRhNGtEektDTTdoV
XhJdENVMlUiLCAiYWRkcmVzcyI6IHsibG9jYWxpdHkiOiAiVkZnS0hQWG5OclpIZW9Cd
2N1NjFiNVZDb0ZWWDByUWp0SDVhT2lpTHowRSIsICJwb3N0YWxfY29kZSI6ICJHOFhIa
ThzQ1BjNDVXQVRlcnk2UlN2bkVjZHlwbnJqeXBqQmw0TEJkNVlFIiwgImNvdW50cnkiO
iAiWXlHNE5oeWZqaXRwbzYteU1EUlRBUlNWQW5aTnZrWXFSWTNYZXBvUV9qOCIsICJzd
HJlZXRfYWRkcmVzcyI6ICJOd0FLZkF0alFjTl9YYlYza3VIdDNnYlVNdlE4M24wMkMxR
WV4STlSbzJBIn19fSwgImJpcnRoX21pZGRsZV9uYW1lIjogIk01R2hrdk5jR2pHT05SZ
XkycFJPUnVMMnlDZll6NWpvMFhxRjZLMHRVV2siLCAic2FsdXRhdGlvbiI6ICI4bTAtc
0JOQThJODhfTERjMDVDN2dFMzFwVG1fQ1hRZmV3aXdsTDFTbjFZIiwgIm1zaXNkbiI6I
CJkTFFWTURJa0VIbm1QVnZ1SE5ZaXY3V3dBcUdFN21ieUpNaDVFZmJqTTFRIn0sICJ2Z
XJpZmllZF9jbGFpbXMiOiB7ImNsYWltcyI6IHsiYmlydGhkYXRlIjogIjE5NTYtMDEtM
jgiLCAicGxhY2Vfb2ZfYmlydGgiOiB7ImNvdW50cnkiOiAiREUiLCAibG9jYWxpdHkiO
iAiTXVzdGVyc3RhZHQifX19fQ.57pncJcJ6cQt2fSARbQLlj6e6nYMpWqHNvI2Ep45Wm
NGuTtI3htmodK8svpgbrT-RaLL25WF7J3CqP1ElzpZSgVFs2VXCxGXgnTG6dQIvk2qPP
fP-45hrZiMWyiwRFBr7Di68J01N90yFGbsMH5hh8kGGqFnCpTSQwvk--6aG_03l0nGmL
DjOFyauCF_Tl-SlOHzNGYoP3MOOX9jU25T8z2e3EmLVLTa5KEmNis0GbpfSHUthbtZCC
Taq-bSYaPDUHi22ZNqeoW1Y4v8nSaNIyrV9IxfPJNb37kYN6NLn5zwI33sxE_nCd8wOx
vuI0rtFtmpS_-DNgwPTnLphzUNKA

An HS-Disclosures JWT for some of the claims may look as follows:

```
{
  "nonce": "XZOUco1u_gEPknxS78sWWg",
  "aud": "https://example.com/verifier",
  "sd_hs_disclosures": {
    "verified_claims": {
      "verification": {
        "trust_framework": "{\"s\": \"SJKr-Pydh8RqHomXCOiVwQ\",
          \"v\": \"de_aml\"}",
        "time": "{\"s\": \"CrxH2Ez8uu2t7tEPQqwZig\", \"v\":
          \"2012-04-23T18:25Z\"}",
        "evidence": [
          {
            "type": "{\"s\": \"sPCCbZtOdjnQjfOiPBxOYA\", \"v\":
              \"document\"}"
          }
        ]
      },
      "claims": {
        "given_name": "{\"s\": \"kqwnbB6oHhaBD3F3t-KUGw\", \"v\":
          \"Max\"}",
        "family_name": "{\"s\": \"_6Do5glcgEQDMVJoPArGSA\", \"v\":
          \"Meier\"}"
      }
    }
  }
}
```

After verifying the SD-JWT and HS-Disclosures, the Verifier merges the selectively disclosed claims into the other data contained in the JWT. The Verifier will then pass the result on to the application for further processing:

```
{
  "verified_claims": {
    "verification": {
      "trust_framework": "de_aml",
      "time": "2012-04-23T18:25Z",
      "evidence": [
        {
          "type": "document"
        }
      ]
    },
    "claims": {
      "given_name": "Max",
      "family_name": "Meier",
      "birthdate": "1956-01-28",
      "place_of_birth": {
        "country": "DE",
        "locality": "Musterstadt"
      }
    }
  },
  "iss": "https://example.com/issuer",
  "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"
    }
  },
  "iat": 1516239022,
  "exp": 1516247022
}
```

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). II-Disclosures Object is sent alongside a VC.

HS-Disclosures JWT is equivalent to a Holder-signed W3C Verifiable Presentation (W3C VP).

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

HS-Disclosures JWT as a W3C VP contains a verifiableCredential claim inside a vp claim that is a string array of an SD-JWT as a W3C VC using JWT compact serialization.

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].

II-Disclosures Object is the same as in Example 1.

```
{
  "sub": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCc
    d-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs",
  "jti": "http://example.edu/credentials/3732",
  "iss": "https://example.com/keys/foo.jwk",
  "nbf": 1541493724,
  "iat": 1541493724,
  "exp": 1573029723,
  "cnf": {
    "jwk": {
      "kty":"RSA",
      "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx
     4cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMs
     tn64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FDW2
     QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbI
     SD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqb
     w0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
      "e":"AQAB"
    }
  },
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1"
    ],
    "type": [
      "VerifiableCredential",
      "UniversityDegreeCredential"
    ],
    "credentialSubject": {
      "first_name": "Jane",
      "last_name": "Doe"
    }
  },
  "sd_digests": {
    "vc": {
      "credentialSubject": {
        "email": "-Rcr4fDyjwlM_itcMxoQZCE1QAEwyLJcibEpH114KiE",
        "phone_number": "Jv2nw0C1wP5ASutYNAxrWEnaDRIpiF0eTUAkUOp8F6Y",
        "address": "ZrjKs-RmEAVeAYSzSw6GPFrMpcgctCfaJ6t9qQhbfJ4",
        "birthdate": "qXPRRPdpNaebP8jtbEpO-skF4n7v7ASTh8oLg0mkAdQ"
      }
    }
  }
}
```

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

```
{
  "alg": "RS256",
  "typ": "JWT",
  "jwk": {
      "kty":"RSA",
      "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx
     4cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMs
     tn64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FDW2
     QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbI
     SD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqb
     w0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
      "e":"AQAB"
    }
}.{
  "iss": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCc
    d-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs",
  "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_hs_disclosures": {
    "vc": {
      "credentialSubject": {
        "email": "{\"s\": \"Pc33JM2LchcU_lHggv_ufQ\", \"v\":
          \"johndoe@example.com\"}",
        "phone_number": "{\"s\": \"lklxF5jMYlGTPUovMNIvCA\", \"v\":
          \"+1-202-555-0101\"}",
        "address": "{\"s\": \"5bPs1IquZNa0hkaFzzzZNw\", \"v\":
          {\"street_address\": \"123 Main St\", \"locality\":
          \"Anytown\", \"region\": \"Anystate\", \"country\":
          \"US\"}}",
        "birthdate": "{\"s\": \"y1sVU5wdfJahVdgwPgS7RQ\", \"v\":
          \"1940-01-01\"}"
      }
    }

  }
}
```

A.5. Blinding Claim Names

The following examples show the use of blinded claim names.

A.5.1. Example 5: Some Blinded Claims

The following shows the user information used in this example, included a claim named secret_club_membership_no:

```
{
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "given_name": "John",
  "family_name": "Doe",
  "email": "johndoe@example.com",
  "phone_number": "+1-202-555-0101",
  "secret_club_membership_no": "23",
  "other_secret_club_membership_no": "42",
  "address": {
    "street_address": "123 Main St",
    "locality": "Anytown",
    "region": "Anystate",
    "country": "US"
  },
  "birthdate": "1940-01-01"
}
```

Hiding just the claim secret_club_membership_no, the SD-JWT payload shown in the following would result. Note that the claims are sorted (here by unicode code point numbers) as described in Section 7.7.

```
{
  "cnf": {
    "jwk": {
      "e": "AQAB",
      "kty": "RSA",
      "n": "pm4bOHBg-oYhAyPWzR56AWX3rUIXp11_ICDkGgS6W3ZWLts-hzwI3x65
        659kg4hVo9dbGoCJE3ZGF_eaetE30UhBUEgpGwrDrQiJ9zqprmcFfr3qvvkG
        jtth8Zgl1eM2bJcOwE7PCBHWTKWYs152R7g6Jg2OVph-a8rq-q79MhKG5QoW
        _mTz10QT_6H4c7PjWG1fjh8hpWNnbP_pv6d1zSwZfc5fl6yVRL0DV0V3lGHK
        e2Wqf_eNGjBrBLVklDTk8-stX_MWLcR-EGmXAOv0UBWitS_dXJKJu-vXJyw1
        4nHSGuxTIK2hx1pttMft9CsvqimXKeDTU14qQL1eE7ihcw"
    }
  },
  "exp": 1516247022,
  "iat": 1516239022,
  "iss": "https://example.com/issuer",
  "sd_digest_derivation_alg": "sha-256",
  "sd_digests": {
    "HS4QoeE9ty-I8BZTEupSzw":
      "emp2qhunGPulOGvtgor5dFwNSasDewLqNdqXCkYl4Nw",
    "address": {
      "country": "Bktf3gG1tXbn0XObrZT53RUr_lxMLZGEguLYwCvsaIg",
      "locality": "NeWRh4B9JLRfEODwno3UOXg9Pg3gtZEo45cK9pr4eZk",
      "region": "qpgFbdX1Az4Hm_E63K3J94oMzazHLCqqFb0Damo2eFE",
      "street_address":
        "6Ex8b2gEeACuMal74_OBH_ROVNM7wvzjSck08EC9eSs"
    },
    "birthdate": "1IjWWzdrXEs7iXUbsahdx_-8CIJsz2bcHHH_ccwgTBg",
    "email": "gszmttjNfSw7_uL31KyJRvWgL1gHM6O3LFAzqxluWDQ",
    "family_name": "Xbz5qK4Fqg-bS_CdwQYd_7qiNS9W810mRn42-FTHMPo",
    "given_name": "asBCBSyK-B45q79qxGMe6j4MijK4lZsHHCD8O_jsDdc",
    "other_secret_club_membership_no":
      "3RP5qguZWamNuvdrFS-sqqYq_MaCIzx6Zn_bOZyE9BY",
    "phone_number": "lB98F2RApo-ifhA3lwJGdqV-PAURkstN-oHmCv4LmxA",
    "sub": "sJ88WF6Q05a2eyPnLJHXzZ8bbiQXWlXl44Nss7Ywk0E"
  }
}
```

In the II-Disclosures Object, it can be seen that the blinded claim's original name is secret_club_membership_no. Note that the claims are sorted alphabetically as described in Section 7.7.

```
{
  "sd_ii_disclosures": {
    "HS4QoeE9ty-I8BZTEupSzw": "{\"s\": \"iq6rolXF0SyWSsdCeaETNg\",
      \"v\": \"23\", \"n\": \"secret_club_membership_no\"}",
    "address": {
      "country": "{\"s\": \"l-6DlGlNloOsAUlBhMOt_Q\", \"v\":
        \"US\"}",
      "locality": "{\"s\": \"c6kc69Gmh04VVNPRlhOV_g\", \"v\":
        \"Anytown\"}",
      "region": "{\"s\": \"qwybxKQUee9A0mMhzGC-Pg\", \"v\":
        \"Anystate\"}",
      "street_address": "{\"s\": \"qNsw9K05ZngcEqXLEGalHA\", \"v\":
        \"123 Main St\"}"
    },
    "birthdate": "{\"s\": \"OErzfd2Gy6jw1atlcCpr6A\", \"v\":
      \"1940-01-01\"}",
    "email": "{\"s\": \"woZIMokulfwyF_do1czRaA\", \"v\":
      \"johndoe@example.com\"}",
    "family_name": "{\"s\": \"ZXPEdf3K8mtRBKDAMjEcBQ\", \"v\":
      \"Doe\"}",
    "given_name": "{\"s\": \"btsLJCwSb0B7gtVLPMjjqA\", \"v\":
      \"John\"}",
    "other_secret_club_membership_no": "{\"s\":
      \"Fj8RxKoVno-9SOVOEUoMpw\", \"v\": \"42\"}",
    "phone_number": "{\"s\": \"YJSPlYo_aenthOCkapFRTg\", \"v\":
      \"+1-202-555-0101\"}",
    "sub": "{\"s\": \"Rj94TRxr3nvOw2WKtujLSA\", \"v\":
      \"6c5c0a49-b589-431d-bae7-219122a9ec2c\"}"
  }
}
```

The Verifier would learn this information via the HS-Disclosures JWT:

```
{
  "nonce": "XZOUco1u_gEPknxS78sWWg",
  "aud": "https://example.com/verifier",
  "sd_hs_disclosures": {
    "given_name": "{\"s\": \"btsLJCwSb0B7gtVLPMjjqA\", \"v\":
      \"John\"}",
    "family_name": "{\"s\": \"ZXPEdf3K8mtRBKDAMjEcBQ\", \"v\":
      \"Doe\"}",
    "birthdate": "{\"s\": \"OErzfd2Gy6jw1atlcCpr6A\", \"v\":
      \"1940-01-01\"}",
    "address": {
      "region": "{\"s\": \"qwybxKQUee9A0mMhzGC-Pg\", \"v\":
        \"Anystate\"}",
      "country": "{\"s\": \"l-6DlGlNloOsAUlBhMOt_Q\", \"v\":
        \"US\"}"
    },
    "HS4QoeE9ty-I8BZTEupSzw": "{\"s\": \"iq6rolXF0SyWSsdCeaETNg\",
      \"v\": \"23\", \"n\": \"secret_club_membership_no\"}"
  }
}
```

The Verifier would decode the data as follows:

```
{
  "given_name": "John",
  "family_name": "Doe",
  "birthdate": "1940-01-01",
  "address": {
    "region": "Anystate",
    "country": "US"
  },
  "secret_club_membership_no": "23"
}
```

A.5.2. Example 6: All Claim Names Blinded

In this example, all claim names are blinded. The user data includes a non-standard delivery_address claim to show that even though the same claim name appears at different places within the structure, different salts and blinded claim names are used for them:

```
{
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "given_name": "John",
  "family_name": "Doe",
  "email": "johndoe@example.com",
  "phone_number": "+1-202-555-0101",
  "secret_club_membership_no": "23",
  "address": {
    "street_address": "123 Main St",
    "locality": "Anytown",
    "region": "Anystate",
    "country": "US"
  },
  "delivery_address": {
    "street_address": "123 Main St",
    "locality": "Anytown",
    "region": "Anystate",
    "country": "US"
  },
  "birthdate": "1940-01-01"
}
```

The resulting SD-JWT payload:

```
{
  "cnf": {
    "jwk": {
      "e": "AQAB",
      "kty": "RSA",
      "n": "pm4bOHBg-oYhAyPWzR56AWX3rUIXp11_ICDkGgS6W3ZWLts-hzwI3x65
        659kg4hVo9dbGoCJE3ZGF_eaetE30UhBUEgpGwrDrQiJ9zqprmcFfr3qvvkG
        jtth8Zgl1eM2bJcOwE7PCBHWTKWYs152R7g6Jg2OVph-a8rq-q79MhKG5QoW
        _mTz10QT_6H4c7PjWG1fjh8hpWNnbP_pv6d1zSwZfc5fl6yVRL0DV0V3lGHK
        e2Wqf_eNGjBrBLVklDTk8-stX_MWLcR-EGmXAOv0UBWitS_dXJKJu-vXJyw1
        4nHSGuxTIK2hx1pttMft9CsvqimXKeDTU14qQL1eE7ihcw"
    }
  },
  "exp": 1516247022,
  "iat": 1516239022,
  "iss": "https://example.com/issuer",
  "sd_digest_derivation_alg": "sha-256",
  "sd_digests": {
    "2lrQaXAeV85isgBjuHAOfw":
      "2kT2ohIPzxb8Mt2aa8YJ7Rj_SmTUrSIfzCz8zVXix5E",
    "HTQvLIU4zz7NkMr5p4KDWw":
      "4Hyw9wnR-uEvbJPSyQdrZMz6JY99mLqR_9m_lntD4_s",
    "HmNQxl6SFAx_Su6uDR94lg":
      "tmuay_zrl23ZdDX1hIyI48a4huCiTf70chBEvAl2Qf0",
    "_ZzazarE9UrZHTv3BnHJ1w":
      "uW7QffEkT_Hw4q_LrsIDV5vcQGh7ubQdKSOJc5qXRiQ",
    "address": {
      "MXDpEmt5sRxRxlDw8YAdfA":
        "9Lu5UyimpvSrpJU9R80aEpzemufK8eRH5QEKo5xLJj0",
      "SdQneYafbrvTMuPyyQhj2A":
        "f54HJqBhU7gC1MHPaMYzzlr9vg96qE3eo4kP2zXoKTc",
      "abIR4EGTTgQKNXnmxoY5qA":
        "QDUyK4ACX0mVPfCm7uVQFwbBPNo6_xI6-3fHZaHEQW0",
      "zno2BBCk2a7pk49dcZYnqw":
        "Ho4wYzUNdQow3TBdPmH5Fbq-4Me_fx8pECok3NJMfFM"
    },
    "aqdLloEIUy4FVDdmmSo48w":
      "C5XJjFnU9CX-k_xIo7qxX_CsLKcR5GDqJJ3MBy_o1Zg",
    "cIsqMhsylJDPtEpoqVGLvQ":
      "wWR4GNiwmcPPbTfuIwphr3j4Vs95TCjUzytdiPC7434",
    "delivery_address": {
      "Q7vBvGQFVCw8keOQLY1SVg":
        "sPeIeivTQeApuZ2piXouWMEm1xA_liTae8BsEOQ7z9M",
      "bMMGdJM0qO_zqVo75zun1w":
        "xTM2Ojec5bxvHY6sOt5c47LeMErCR7TSc1tJ51v28tQ",
      "bxNsq8p-Jobl47JNkhNOMA":
        "AF3X_wkKrY4KHiajZ5vhv7CzUp-ATXe-Jtl5x7QUAcg",
      "kR7kfLZF-3YiQ5VRgsY3yA":
        "crbT6qlk8nmEkwqO_GsFUUQHNq7DxoU0ziMh22Cxe7M"
    },
    "vEA0i5_1JvuDfS7hH7TWZw":
      "d9Wa_qCEbikmrXt_1refkreitUPIbZWNn5miQGZWPKg"
  }
}
```

The II-Disclosures Object: {#example-simple*structured*all*blinded-iid*payload}

```
{
  "sd_ii_disclosures": {
    "2lrQaXAeV85isgBjuHAOfw": "{\"s\": \"PdxYWdt_MFsC6qce2uiVLQ\",
      \"v\": \"+1-202-555-0101\", \"n\": \"phone_number\"}",
    "HTQvLIU4zz7NkMr5p4KDWw": "{\"s\": \"353CLP3ZZFmxJQ6aZ_HDYg\",
      \"v\": \"John\", \"n\": \"given_name\"}",
    "HmNQxl6SFAx_Su6uDR94lg": "{\"s\": \"Qb5pmhvwzr4aRd7g7QVckA\",
      \"v\": \"23\", \"n\": \"secret_club_membership_no\"}",
    "_ZzazarE9UrZHTv3BnHJ1w": "{\"s\": \"yL66N684FNAao5hWfBqc6A\",
      \"v\": \"1940-01-01\", \"n\": \"birthdate\"}",
    "address": {
      "MXDpEmt5sRxRxlDw8YAdfA": "{\"s\": \"3VzdS1O4wRgglXFk_ENJ2g\",
        \"v\": \"Anytown\", \"n\": \"locality\"}",
      "SdQneYafbrvTMuPyyQhj2A": "{\"s\": \"VQz3c8LhaQCy7hqEsusPPA\",
        \"v\": \"US\", \"n\": \"country\"}",
      "abIR4EGTTgQKNXnmxoY5qA": "{\"s\": \"BhX1rStOsN3_vk_Kx4IgOg\",
        \"v\": \"Anystate\", \"n\": \"region\"}",
      "zno2BBCk2a7pk49dcZYnqw": "{\"s\": \"u2jvKsy0g-inkL3RAcpssw\",
        \"v\": \"123 Main St\", \"n\": \"street_address\"}"
    },
    "aqdLloEIUy4FVDdmmSo48w": "{\"s\": \"Y3d_N7vZNfNp7KWDmCpJlA\",
      \"v\": \"Doe\", \"n\": \"family_name\"}",
    "cIsqMhsylJDPtEpoqVGLvQ": "{\"s\": \"bbdW6Rtr4YEaDvydH4Yerw\",
      \"v\": \"johndoe@example.com\", \"n\": \"email\"}",
    "delivery_address": {
      "Q7vBvGQFVCw8keOQLY1SVg": "{\"s\": \"nBOOpTNOcCScA_MHr9P9SQ\",
        \"v\": \"Anystate\", \"n\": \"region\"}",
      "bMMGdJM0qO_zqVo75zun1w": "{\"s\": \"urI5m4JPtDbe9rRQbXgtEg\",
        \"v\": \"US\", \"n\": \"country\"}",
      "bxNsq8p-Jobl47JNkhNOMA": "{\"s\": \"LojbKO3mpEE6WTgSL5EzMg\",
        \"v\": \"123 Main St\", \"n\": \"street_address\"}",
      "kR7kfLZF-3YiQ5VRgsY3yA": "{\"s\": \"e925I1ajysz2xx9kzyzveg\",
        \"v\": \"Anytown\", \"n\": \"locality\"}"
    },
    "vEA0i5_1JvuDfS7hH7TWZw": "{\"s\": \"i_rQHJJUvGFdOgVVM8H8Ww\",
      \"v\": \"6c5c0a49-b589-431d-bae7-219122a9ec2c\", \"n\":
      \"sub\"}"
  }
}
```

Here, the Holder decided only to disclose a subset of the claims to the Verifier:

```
{
  "nonce": "XZOUco1u_gEPknxS78sWWg",
  "aud": "https://example.com/verifier",
  "sd_hs_disclosures": {
    "HTQvLIU4zz7NkMr5p4KDWw": "{\"s\": \"353CLP3ZZFmxJQ6aZ_HDYg\",
      \"v\": \"John\", \"n\": \"given_name\"}",
    "aqdLloEIUy4FVDdmmSo48w": "{\"s\": \"Y3d_N7vZNfNp7KWDmCpJlA\",
      \"v\": \"Doe\", \"n\": \"family_name\"}",
    "_ZzazarE9UrZHTv3BnHJ1w": "{\"s\": \"yL66N684FNAao5hWfBqc6A\",
      \"v\": \"1940-01-01\", \"n\": \"birthdate\"}",
    "address": {
      "abIR4EGTTgQKNXnmxoY5qA": "{\"s\": \"BhX1rStOsN3_vk_Kx4IgOg\",
        \"v\": \"Anystate\", \"n\": \"region\"}",
      "SdQneYafbrvTMuPyyQhj2A": "{\"s\": \"VQz3c8LhaQCy7hqEsusPPA\",
        \"v\": \"US\", \"n\": \"country\"}"
    }
  }
}
```

The Verifier would decode the HS-Disclosures JWT and SD-JWT as follows:

```
{
  "given_name": "John",
  "family_name": "Doe",
  "birthdate": "1940-01-01",
  "address": {
    "region": "Anystate",
    "country": "US"
  }
}
```

Appendix B. Document History

[[To be removed from the final specification]]

-01

* introduce blinded claim names

* explain why JSON-encoding of values is needed

* explain merging algorithm ("processing model")

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

*sd_digest_derivation_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

*fix 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