jose
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

D. Waite
Ping Identity
M. Jones
Self-Issued Consulting
J. Miller
Ping Identity

2 March 2026

JSON Web Proof
draft-ietf-jose-json-web-proof-13

Abstract

The JOSE set of standards established JSON-based container formats for Keys, Signatures, and Encryption. They also established IANA registries to enable the algorithms and representations used for them to be extended. Since those were created, newer cryptographic algorithms that support selective disclosure and unlinkability have matured and started seeing early market adoption. The COSE set of standards likewise does this for CBOR-based containers, focusing on the needs of environments which are better served using CBOR, such as constrained devices and networks.

This document defines a new container format similar in purpose and design to JSON Web Signature (JWS) and COSE Signed Messages called a _JSON Web Proof (JWP)_. Unlike JWS, which integrity-protects only a single payload, JWP can integrity-protect multiple payloads in one message. It also specifies a new presentation form that supports selective disclosure of individual payloads, enables additional proof computation, and adds a Presentation Header to prevent replay.

Table of Contents

1. Introduction

The JOSE specifications are very widely deployed and well supported, enabling use of cryptographic primitives with a JSON representation. JWTs [RFC7519] are one of the most common representations for identity and access claims. For instance, they are used by the OpenID Connect and Secure Telephony Identity Revisited (STIR) standards. Also, JWTs are used by W3C's Verifiable Credentials and are used in many decentralized identity systems.

With these new use cases, there is an increased focus on adopting privacy-protecting cryptographic primitives. While such primitives are still an active area of academic and applied research, the leading candidates introduce new patterns that are not currently supported by JOSE or COSE. These new patterns are largely focused on two areas: supporting selective disclosure when presenting information and minimizing correlation through the use of Zero-Knowledge Proofs (ZKPs) in addition to traditional signatures.

There are a growing number of these cryptographic primitives that support selective disclosure while protecting privacy across multiple presentations. Examples used in the context of Verifiable Credentials are:

* CL Signatures (https://eprint.iacr.org/2012/562.pdf)
* IDEMIX (http://www.zurich.ibm.com/idemix)
* BBS signatures, described in [I-D.irtf-cfrg-bbs-signatures]
* MerkleDisclosureProof2021 (https://github.com/transmute-industries/merkle-disclosure-proof-2021)
* Mercurial Signatures (https://eprint.iacr.org/2020/979)
* PS Signatures (https://eprint.iacr.org/2015/525.pdf)
* U-Prove (https://www.microsoft.com/en-us/research/project/u-prove/)
* Spartan (https://github.com/microsoft/Spartan)

All of these follow the same pattern of taking multiple claims (a.k.a., "attributes" or "messages" in the literature) and binding them together into a single issued token. These are then later securely one-way transformed into a presentation that reveals potentially only a subset of the original claims, predicate proofs about the claim values, or proofs of knowledge of the claims.

| Editor’s Note: This draft is still early and incomplete. There will be significant changes to the algorithms as currently defined here. Please do not use any of these definitions or examples for anything except personal experimentation and learning. Contributions and feedback are welcomed at https://github.com/ietf-wg-jose/json-web-proof (https://github.com/ietf-wg-jose/json-web-proof).

2. Conventions and Definitions

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.

The roles of "issuer", "holder", and "verifier" are used as defined by the VC Data Model [VC-DATA-MODEL-2.0]. The term "presentation" is also used as defined by this source, but the term "credential" is avoided in this specification to minimize confusion with other definitions.

2.1. Terminology

The following terms are used throughout this series of documents:

binding: A mechanism, indicated in an issued JWP, for how to verify a presentation was created by the intended holder.

linkability: The property where multiple presentations may be correlated to a single issued JWP, either through consistency in the cryptographic integrity or due to particulars of JWP usage by an application. Such issued JWPs may be referred to as single-use, as multiple uses may leak unintended knowledge.

unlinkability: The property of issuance and presentation algorithms and of application usage, where one presentation can only be correlated with other presentations based on holder-disclosed information.

2.2. Abbreviations

* ZKP: Zero-Knowledge Proof
* JWP: JSON Web Proof (this specification)
* JPA: JSON Proof Algorithms [I-D.ietf-jose-json-proof-algorithms]
* JPT: JSON Proof Token [I-D.ietf-jose-json-proof-token]
* CPT: CBOR Proof Token [I-D.ietf-jose-json-proof-token]

3. Background

A _JSON Web Proof (JWP)_ is very similar to a JWS [RFC7515] or COSE Signed Message [RFC9052], with the addition that it can contain multiple individual secured payloads instead of a single one. JWP-supporting algorithms are then able to separate and act on the individual payloads contained within.

The intent of JSON Web Proof is to establish a common container format for multiple ordered payloads that can be integrity-verified against a cryptographic proof value also in the container. It does not create or specify any cryptographic protocols, multi-party protocols, or detail any algorithm-specific capabilities.

To fully support the newer privacy primitives, JWP utilizes the three roles of issuer, holder, and verifier, as defined by the VC Data Model [VC-DATA-MODEL-2.0]. There are also two forms of a JWP: the issued form created by an issuer for a holder, and the presented form created by a holder for a verifier.

The four principal interactions used by JWP are issue, confirm, present, and verify.

A JWP is initially created by the issuer using the issue interaction. A successful result is an issued JWP that has a single Issuer Header, one or more payloads, and an initial proof value that contains the issuing algorithm output. The holder, upon receiving an issued JWP, then uses the confirm interaction to check the integrity protection of the Header and all payloads using the proof value.

After validation, the holder uses the present interaction to apply any selective disclosure choices, perform privacy-preserving transformations for unlinkability, and add a Presentation Header that ensures the resulting presented JWP cannot be replayed. The verifier then uses the verify interaction to ensure the integrity protection of the Headers and any disclosed payloads, along with verifying any additional ZKPs covering non-disclosed payloads.

While issue and confirm only occur when a JWP is initially created by the issuer, the present and verify steps may be safely repeated by a holder on an issued JWP. The resulting presented JWP is only unlinkable when supported by the underlying algorithm.

Algorithm definitions that support JWPs are in separate companion specifications - just as the JSON Web Algorithms [RFC7518] specification does for JWS and JWE [RFC7516]. The JSON Proof Algorithms (JPA) [I-D.ietf-jose-json-proof-algorithms] specification defines how an initial set of algorithms are used with JWP.

4. JSON Web Proof Header

A Header consists of an integrity-protected set of Header Parameters that apply to the JWP.

A Header is represented either as a JSON Object holding JSON-formatted Header Parameters, or a CBOR map holding CBOR-formatted Header Parameters.

In addition, there are two kinds of Headers - an Issuer Header (see Section 6.1.1) supplied on issue by the issuer, and an additional Presentation Header (see Section 6.2.1) supplied on present by the holder. Both Headers should have their integrity protected, and are mandatory to present unmodified to the verifier.

5. Header Parameter

A Header Parameter is a name/value pair supplying information within a Header in a JWP. A Header Parameter may provide information specific to the proof algorithm in use for the JWP, it may identify the issuer of the proof, it may describe the application purpose or format of the JWP, as well as provide other potential metadata.

A Header Parameter may be represented as JSON or as CBOR. When represented using JSON, each Header Parameter has a string label and has a JSON-structured value within a JSON Object. When described using CBOR, each parameter has either an integer (int) or string (tstr) label, and has a CBOR-structured value within a CBOR map.

The Header Parameter labels within the Header MUST be unique. CBOR processing MUST reject messages if two Headers with the same parameter label are encountered. JSON processing SHOULD reject messages received with the same parameter label, but MAY instead represent only the lexically last member with that label, as specified in Section 15.12 ("The JSON Object") of ECMAScript 5.1 [ECMAScript]. JSON processing MUST take one of these two approaches with regards to encountering duplicate Header Parameter labels.

Implementations are required to understand the specific Header Parameters defined by this specification that are designated as "MUST be understood" and process them in the manner defined in this specification. All other Header Parameters defined by this specification that are not so designated MUST be ignored when not understood. Unless listed as a critical Header Parameter, per Section 5.2.4, all Header Parameters not defined by this specification MUST be ignored when not understood.

5.1. Header Parameter Labeling Requirements

As labels are the mechanism for semantically distinguishing parameter names, it is important to describe the mechanism to reduce the risk of conflicts.

There are three strategies for labeling Header Parameters:

1. Registered parameter labels. These labels are coordinated through the IANA "JSON Web Proof Header Parameters" registry, which protects against parameters having the same label.

2. Collision-resistant parameter labels. These labels are not coordinated through IANA, but are otherwise namespaced to prevent conflict. One example would be a string label representing the URI of a controlled resource, such as the HTTPS-hosted documentation of the Header Parameter.

3. Private parameter labels. These labels are not coordinated through IANA or another party, but are expected to only be used for testing or in closed environments.

These classes of Header Parameters are intentionally parallel to those in Section 4 of [RFC7515].

5.2. Registered Header Parameter Labels

The following Header Parameter names for use in JWPs are registered in the IANA "JSON Web Proof Header Parameters" registry established by Section 11.1, with meanings as defined in the subsections below.

As indicated by the common registry, Header Parameters used in the Issuer Header (see Section 6.1) and the Presentation Header Section 6.2 share a common Header Parameter space; when a parameter is used by both forms, its usage must be compatible between them.

5.2.1. "alg" (Algorithm) Header Parameter

The alg (algorithm) Header Parameter identifies the cryptographic algorithm used to secure the JWP. The JWP Proof value is not valid if the alg value does not represent a supported algorithm or if there is not a key for use with that algorithm associated with the party that secured the content. alg values should either be registered in the IANA "JSON Web Proof Algorithms" registry established by [I-D.ietf-jose-json-proof-algorithms] or be a value that contains a Collision-Resistant Name.

As a JSON-formatted Header Parameter, the alg value is a case-sensitive ASCII string containing a StringOrURI value. As a CBOR-formatted Header Parameter, this value may also be an integer value.

The list of defined alg values for this use can be found in the IANA "JSON Web Proof Algorithms" registry established by [I-D.ietf-jose-json-proof-algorithms]; the initial contents of this registry are registered by [I-D.ietf-jose-json-proof-algorithms].

Use of this Header Parameter is REQUIRED.

5.2.2. "kid" (Key ID) Header Parameter

The kid (Key ID) Header Parameter is a hint indicating which key was used to secure the JWP. This parameter allows originators to explicitly signal a change of key to recipients.

The structure of the kid value is unspecified.

When kid is used for a JSON-formatted Header, its value MUST be a case-sensitive string. When referencing a JWK, the kid value is matched to the JWK kid parameter value.

When kid is used for a CBOR-formatted Header, its value is a binary string. When referencing a COSE Key, the kid value is matched to the COSE_Key kid structure member.

Use of this Header Parameter is OPTIONAL.

5.2.3. "typ" (Type) Header Parameter

The typ (type) Header Parameter is used by JWP applications to declare the media type [IANA.MediaTypes] of this complete JWP. This is intended for use by the application when more than one kind of object could be present in an application data structure that can contain a JWP; the application can use this value to disambiguate among the different kinds of objects that might be present. It will typically not be used by applications when the kind of object is already known. This parameter is ignored by JWP implementations; any processing of this parameter is performed by the JWP application. Use of this Header Parameter is OPTIONAL.

For COSE-formatted Headers, typ MAY also instead be an integer value which corresponds to the IANA "CoAP Content-Formats" registry [IANA.CoAP.Formats], which describes the corresponding media type, as described in [RFC9596].

Per [RFC2045], all media type values, subtype values, and parameter names are case insensitive. However, parameter values are case sensitive unless otherwise specified for the specific parameter.

To keep messages compact in common situations, it is RECOMMENDED that producers omit an "application/" prefix of a media type value in a typ Header Parameter when no other '/' appears in the media type value. A recipient using the media type value MUST treat it as if "application/" were prepended to any typ value not containing a '/'. For instance, a typ value of example SHOULD be used to represent the application/example media type, whereas the media type application/example;part="1/2" cannot be shortened to example;part="1/2".

The typ value jwp can be used by applications to indicate that this object is a JWP using the JWP Compact Serialization. Other type values can also be used by applications, including those using the +jwp media type structured syntax suffix.

It is RECOMMENDED that the typ Header Parameter be used for explicit typing, in parallel to the recommendations in Section 3.11 of [RFC8725].

5.2.4. "crit" (Critical) Header Parameter

The crit (critical) Header Parameter indicates that extensions to this specification and/or [I-D.ietf-jose-json-proof-algorithms] are being used that MUST be understood and processed. Its value is an array listing the Header Parameter labels present in the JWP Header that use those extensions. For JSON-formatted Headers this is a list of strings, while for CBOR-formatted Headers it is a list containing string and/or int values.

If any of the listed extension Header Parameters are not understood and supported by the recipient, then the JWP is invalid. Producers MUST NOT include Header Parameter names defined by this specification or [I-D.ietf-jose-json-proof-algorithms] for use with JWP, duplicate names, or names that do not occur as Header Parameter names within the Header in the crit list. Producers MUST NOT use the empty list [] as the crit value. Recipients MAY consider the JWP to be invalid if the critical list contains any Header Parameter names defined by this specification or [I-D.ietf-jose-json-proof-algorithms] for use with JWP or if any other constraints on its use are violated. When used, this Header Parameter MUST be integrity protected; therefore, it MUST occur only within the Header. Use of this Header Parameter is OPTIONAL. This Header Parameter MUST be understood and processed by implementations.

5.2.5. "iek" (Issuer Ephemeral Key) Header Parameter

The iek (Issuer Ephemeral Key) represents the public key used by the issuer for indirect signatures within certain algorithms. This is an ephemeral key that MUST be unique for each issued JWP.

This Header Parameter is references a JSON Web Key (JWK) public key value when represented as a JSON-formatted Header, and a COSE Key Object when represented as a CBOR-formatted Header.

It MUST contain only public key parameters and SHOULD contain only the minimum parameters necessary to represent the key; other parameters included can be checked for consistency and honored, or they can be ignored.

When present, this Header Parameter MUST be understood and processed by implementations.

5.2.6. "hpk" (Holder Presentation Key) Header Parameter

The hpk (Holder Presentation Key) represents the public key with certain algorithms, and is used by the holder for proof of possession and integrity protection of the Presented Header.

The issuer MUST validate that the holder has possession of this key through a trusted mechanism, such as requiring the signature of a unique nonce value from the holder before issuing the JWP.

This Header Parameter is references a JSON Web Key (JWK) public key value when represented as a JSON-formatted Header, and a COSE Key Object when represented as a CBOR-formatted Header.

It MUST contain only public key parameters and SHOULD contain only the minimum parameters necessary to represent the key; other parameters included can be checked for consistency and honored, or they can be ignored.

If holder unlinkability is required, this value MUST not be repeated in multiple issued JWPs; a different holder presentation key MUST be included in each issuance.

This Header Parameter MUST be understood and processed by implementations when present.

5.2.7. "hpa" (Holder Presentation Algorithm) Header Parameter

The hpa (Holder Presentation Algorithm) Header Parameter represents the algorithm to be used by the holder for presenting a JWP when using an asymmetric algorithm and a Holder Presentation Key.

This Header Parameter SHOULD be included when appropriate for the JWP algorithm unless a single appropriate algorithm is negotiated through other means.

This Header Parameter references the name of a JSON Web Algorithm (JWA) when represented as a JSON-formatted Header, and an integer or text value when represented as a CBOR-formatted Header.

This Header Parameter MUST be understood and processed by implementations when present.

5.2.8. "iss" (Issuer) Header Parameter

The iss (issuer) Header Parameter identifies the principal that issued the JWP. The processing of this claim is generally application specific.

The iss value is a case-sensitive string containing a StringOrURI value. Its definition is intentionally parallel to the iss claim defined in [RFC7519].

Use of this Header Parameter is OPTIONAL.

5.2.9. "aud" (Audience) Header Parameter

The aud (audience) Header Parameter identifies the recipients that the JWP is intended for. Each principal intended to process the JWP MUST identify itself with a value in the audience Header Parameter. If the principal processing the Header Parameter does not identify itself with a value in the aud Header Parameter when this Header Parameter is present, then the JWP MUST be rejected.

In the general case, the aud value is an array of case-sensitive strings, each containing a StringOrURI value. In the special case when the JWP has one audience, the aud value MAY be a single case-sensitive string containing a StringOrURI value.

The interpretation of audience values is application specific.

Its definition is intentionally parallel to the aud claim defined in [RFC7519].

Use of this Header Parameter is OPTIONAL.

5.2.10. "nonce" (Nonce) Header Parameter

The nonce (nonce) Header Parameter is used to associate protocol state with a presented JWP. Usage is protocol-specific, but examples include requiring a unique nonce in requests as part of a strategy to prevent replay, or for associating a JWP back to the context where it was requested.

When used as a JSON-formatted Header, the value is a case-sensitive string value.

When used as a CBOR-formatted Header, the value is a binary string.

This definition is intentionally parallel to the nonce claim registered in the IANA "JSON Web Token Claims" registry [IANA.JWT].

Use of this Header Parameter is OPTIONAL.

5.3. Public Header Parameter Names

Additional Header Parameter names can be defined by those using JWPs. However, in order to prevent collisions, any new Header Parameter name should either be registered in the IANA "JSON Web Proof Header Parameters" registry established by Section 11.1 or be a Public Name (a value that contains a Collision-Resistant Name). In each case, the definer of the name or value needs to take reasonable precautions to make sure they are in control of the part of the namespace they use to define the Header Parameter name.

New Header Parameters should be introduced sparingly, as they can result in non-interoperable JWPs.

5.4. Private Header Parameter Names

A producer and consumer of a JWP may agree to use Header Parameter names that are Private Names (names that are not Registered Header Parameter labels Section 5.2 or Public Header Parameter names Section 5.3.) Unlike Public Header Parameter names, Private Header Parameter names are subject to collision and should be used with caution.

6. JWP Forms

A JWP is always in one of two forms: the issued form or the presented form. A structural difference between the two forms is the number of Headers. An issued JWP has only an Issuer Header, while a presented JWP will have both the Issuer Header and an additional Presentation Header.

All JWP forms support multiple payloads, which are individual octet strings. The issued form will contain one or more ordered payload slots, each of which contain one piece of payload data. A presentation based on that issued JWP contains the same number of slots in the same order, but may choose to omit payload information on a slot-by-slot basis.

The JWP proof value is one or more octet strings that are only meant to be generated from and processed by the underlying JPA. Internally, the proof value may contain one or more cryptographic statements that are used to check the integrity protection of the Header(s) and all payloads. Each of these statements may be a ZKP or a traditional cryptographic signature. The algorithm is responsible for how these statements are serialized into a single proof value.

6.1. Issued Form

When a JWP is first created, it is always in the issued form. It will contain the Issuer Header along with all of the payloads.

The issued form can only be confirmed by a holder as being correctly formed and protected. It is NOT to be verified directly or presented as-is to a verifier. The holder SHOULD treat an issued JWP as private and use appropriately protected storage.

6.1.1. Issuer Header

The Issuer Header is always presented as-is to verifiers. Differences in Headers from two Issued JWP could unintentionally serve to differentiate these messages to verifiers, allowing grouping or correlation of credentials based on these variations. It is RECOMMENDED that the Issuer Header have the same representation (identical octet string sequence) for Issued JWP which are otherwise meant to not be distinguishable.

Every Issuer Header MUST have an alg value that identifies a valid JSON Proof Algorithm (JPA).

For example:

```
{
  "alg": "BBS"
}
```

6.1.2. Issuer Payloads

Payloads are represented and processed as individual octet strings and arranged in an ordered array of payload slots. All application context of the placement and encoding of each payload value is out of scope of this specification and SHOULD be well defined and documented by the application or other specifications.

JPAs MAY provide software interfaces that perform the encoding of individual payloads which accept native inputs such as numbers, sets, or elliptic curve points. This enables the algorithm to support

advanced features such as blinded values and predicate proofs. These interfaces would generate the octet string encoded payload value as well as include protection of that payload in the combined proof value.

6.1.3. Issuer Proof

The issuer proof is one or more octet strings that are opaque to applications. Individual proof-supporting algorithms are responsible for the contents and security of the proof value, along with any required internal structures.

The issuer proof is used by the holder to perform validation, checking that the Issuer Header and all payloads are properly encoded and protected by the given proof.

6.2. Presented Form

When an issued JWP is presented, it undergoes a transformation that adds a Presentation Header. While the payload slots are identical to the Issued JWP, the Presented JWP may have one or more payloads omitted, disclosing only a subset of the original issued payloads. The proof value will always be updated to add integrity protection of the Presentation Header along with the necessary cryptographic statements to verify the presented JWP.

When supported by the underling JPA, a single issued JWP can be used to safely generate multiple presented JWPs without becoming correlatable.

A JWP may also be single use, where an issued JWP can only be used once to generate a presented form. In this case, any additional presentations would be inherently correlatable. These are still useful for applications needing only selective disclosure or where new unique issued JWPs can be retrieved easily.

6.2.1. Presentation Header

The presented form of a JWP MUST contain a Presentation Header. It is added by the holder and MUST be integrity protected by the underling JPA.

This Header is used to ensure that a presented JWP cannot be replayed and is cryptographically bound to the verifier it was presented to.

While there are not any required Header Parameters in the Presentation Header, it MUST contain one or Header Parameters that uniquely identify the presented JWP to both the holder and verifier. For example, Header Parameters that would satisfy this requirement include nonce and aud.

6.2.2. Presentation Payloads

The presentation has one or more payload slots, each of which either contains the issued payload or represents that the payload is being omitted. The position of other payloads does not change when one is omitted; the resulting array will simply be sparse and only contain the disclosed payloads.

The disclosed payloads will always be in the same array positions to preserve any index-based references by the application between the issued and presented forms of the JWP. How the sparse array is represented is specific to the serialization used.

6.2.3. Algorithm Specific Proof Methods

In addition to disclosing the contents of a payload, some algorithms MAY support disclosing other information not representable as a payload.

For example, rather than releasing a date of birth, the algorithm may include proof information indicating that the subject is over a certain age at the time of the presentation. Such information is not disclosed as a payload, and instead would be included in the presentation proof.

A non-exhaustive list of such proof methods along with examples includes:

* Ranges: Age acceptable currently, Geolocation near a specified location
* (Non-)Membership: Contains a particular role, Credential is not considered revoked.
* Proof of Knowledge: Holder has proof of possession of a secret key or secret value
* Derived Values: (Verifier-specific) stable pseudonymous values

6.2.4. Presentation Proof

The presentation proof is one or more octet strings that are opaque to applications. Individual proof-supporting algorithms are responsible for the contents and security of the proof value, along with any required internal structures.

The proof of a presented JWP will always be different than the issued proof. At a minimum, it MUST be updated to include protection of the added Presentation Header.

Algorithms SHOULD generate an un-correlatable presentation proof in order to support multiple presentations from a single issued JWP.

The algorithm is responsible for representing selective disclosure of payloads in a presented proof. If multiple octet strings are insufficient for representing a proof as defined by an algorithm, the algorithm is responsible for defining how such information is represented within one or more octet strings.

7. Serializations

JWP defines two serializations: a JSON-based Compact Serialization and a CBOR Serialization. Both serializations represent one or more Headers, multiple Payload slots, and a single Proof (which may be composed of multiple octet strings).

The JWP Compact Serialization provides a JSON-based, space-efficient encoding of a JWP in URL-safe characters. Its design closely parallels the JWS Compact Serialization [RFC7515]. No representation parallel to the JWS JSON serialization is defined.

JWP CBOR Serialization provides a compact CBOR-based encoding suitable for constrained environments. Its design closely parallels COSE_Sign1 [RFC9338].

7.1. Compact Serialization

JWP Compact Serialization provides a JSON-based encoding of a JWP, expressed in URL-safe characters. In addition to the alphabet of unpadded BASE64URL-safe encoding, Compact Serialization uses the "." and "~" characters as separators. This serialization is inspired by JWS.

The Header MUST be JSON-formatted for Compact Serialization. This includes both Headers in presented form.

All binary data is BASE64URL encoded, including the octets of the UTF-8 encoded Headers and the individual payload slot data and the proof values.

Payload slots and proof values are each concatenated into a single text form by concatenating the BASE64URL encoded values using the ~ character.

Individual payload slots are allowed to have their payload data be omitted; if a payload is omitted, it is represented as a zero-length text value, potentially resulting in leading, trailing, or consecutive ~ characters in the concatenated form.

If a payload or proof value was a zero-length octet string, it does not get output as its zero-length BASE64URL-encoded form but as a single _ character. This character does not represent a valid BASE64URL-encoded octet string, allowing it to be distinguished from normally encoded data.

The issued form is created by concatenating the base64url-encoded Issuer Header, concatenated payloads, and concatenated proof separated each by a . character. The concatenated payloads MAY be omitted if the application is using detached payloads.

The presented form is created by concatenating the base64url-encoded Presentation Header, base64url-encoded Issuer Header, concatenated payloads, and concatenated proof separated each by a . character. The concatenated payloads MAY be omitted if the application is using detached payloads.

eyJhbGciOiJCQlMiLCJhdWQiOiJodHRwczovL3JlY2lwaWVudC5leGFtcGxlLmNvbSIsI
m5vbmNlIjoid3JtQlJrS3RYalEifQ.eyJhbGciOiJCQlMiLCJraWQiOiJIamZjcHlqdVp
RLU84WWUyaFFuTmJUOVJiYm5yb2JwdGRuRXhSMERValU4In0.MTcxNDUyMTYwMA~MTcxN
zE5OTk5OQ~IkRvZSI~IkpheSI~~~.j8Un-MjA3J3D5HN9_dC3WLgFx51MIWLc--NPxO1N
SIb7RZENkiwMo8pzHcq3uevulTEy0Ni_BqBQ6wrS19uBtfBAlsqf3boYAwroR4RIMFc7D
Ko3qhKgy_Y7A1Zt_2CIrfjEPj_PsaKCntHZRjGZs9ZA1tzvc3CyrGYlE4ssiEHU9AY_t1
eGCEro_nI8VAVhCVOyl-_GkhjstRLxACFlM8lAJ1MRrNPfIYKsMjFV5PlBkY9sab1dE5S
n7ZpQyQEl2g9jlLfYm2plGUPT4KV6mV0sWlmAT73XWQwsnc6WR_dhoL-QNLSBRgKXklDu
yN40M02Qy0SndGHx-W-rlvCD7LkoHbuKpX2GtyE6aR4EBxMsMtRZXLsfD0JzG37TfRw2F
II_BFslao_0XsWNbpUATyX5DvYB0Uvzd94a_B0eCuf-qfoLTlZIeqWZIO9kVVk3Iu1HSh
pL1vx21g7iVQM7Wo1jgkwjpcml7Nn4WGFee_s

| Figure: Compact Serialization of Presentation

7.2. CBOR Serialization

A CBOR-based serialization is also defined, which uses the CBOR for describing Header Parameters and does not use base64url encoding to ensure safety in text-based protocols. While this supports the same data model and algorithms, the difference in Header representations does not allow interchangeability with Compact Serialization.

The CBOR serialization provides a compact binary representation of a JWP. The serialization consists of two arrays, representing issued and presented forms.

The Headers MUST be CBOR formatted for CBOR serialization. This includes both the issued and Presented Headers in the presented form.

The issued form consists of a three-element array, while the presented form consists of a four-element array.

If an individual payload has been omitted, it is represented by the CBOR value nil. Payloads MUST be included unless the application is using detached payloads, which is represented by setting the payloads value as nil.

Two tags are defined for representing issued and presented JWPs. Applications MAY use their own tags to tag other specific types of JWPs.

```
CBOR_JWP_Issued = [
       IssuerHeader : empty_or_serialized_map,
       PayloadSlots : [+ payload] / nil,
       Proof : [+ bstr]
   ]

CBOR_JWP_Presented = [
      PresenterHeader : empty_or_serialized_map,
      IssuerHeaders : empty_or_serialized_map,
      PayloadSlots : [+ disclosable_payload] / nil,
      Proof : [+ bstr]
   ]

empty_or_serialized_map = bstr .cbor header_map

header_map = {
    * label => values
}

label = int / tstr

values = any

payload = bstr

disclosable_payload = payload / nil

Tagged_CBOR_JWP_Issued = #6.xxx (CBOR_JWP_Issued)

Tagged_CBOR_JWP_Presented = #6.yyy (CBOR_JWP_Presented)
```

Figure 1: CDDL [RFC8610] for CBOR Serializations.

8. Encrypted JSON Web Proofs

Access to JWPs containing non-public material by parties without legitimate access to the non-public information MUST be prevented. This can be accomplished by encrypting the JWP when potentially observable by such parties to prevent the disclosure of private information. The use of an Encrypted JWP is recommended for this purpose. The processing of Encrypted JWPs is identical to the processing of other JWEs.

For a JWP with JSON-formatted Headers, an Encrypted JWP is a JWE [RFC7516] with a JWP in Compact Serialization as its plaintext value. For a JWP with CBOR-formatted Headers, an Encrypted JWP should use COSE_Encrypt0 or COSE_Encrypt [RFC9052] with the CBOR Serialization as its plaintext.

The cty (content type) JWE/COSE Header Parameter is used to indicate that the content of the JWE is a JWP. The cty value of the JWE/COSE message SHOULD be the same as the typ (type) JWP Header Parameter value of the unencrypted JWP to be encrypted. If the JWP has no typ value, then the following JWE Header Parameter cty (content type) values SHOULD be used:

* jwp is used to indicate that the content of the JWE is a JWP using the JWP Compact Serialization.
* application/cwp is used to indicate that the plaintext of the COSE message is a JWP in CBOR Serialization.

The cty (content type) Header Parameter MUST be present unless the application knows that the encrypted content is a JWP by another means or convention, in which case the cty value MAY be omitted.

9. Detached Payloads

In some contexts, it is useful to make statements about payloads which are not themselves contained within the JWP, similar to "Detached Content" in JWS [RFC7515].

For this purpose, the compact, JSON and CBOR serializations allow for all payload slots to be omitted from a serialized form. While this is a legal serialization, it is not on its own able to be verified.

The recipient is expected to perform some sequence of steps defined by the application to recreate the array of payload slots, including order and optionality. This effectively recreates the fully specified serialization of the JWP.

10. Security Considerations

Notes to be expanded:

* Requirements for supporting algorithms, see JPA
* Application interface for verification
* Data minimization of the Header(s)
* To prevent accidentally introducing linkability, when an issuer uses the same key with the same grouping of payload types, they SHOULD also use the same Issuer Header. This Header SHOULD be represented by the same octets, avoiding distinguishing a JWP due to non-deterministic serialization.

11. IANA Considerations

The following registration procedure is used for all the registries established by this specification.

Values are registered on a Specification Required [RFC5226] basis after a three-week review period on the jose-reg-review@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of values prior to publication, the Designated Experts may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register JWP Header Parameter: example").

Within the review period, the Designated Experts will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful. Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention (using the iesg@ietf.org mailing list) for resolution.

Criteria that should be applied by the Designated Experts includes determining whether the proposed registration duplicates existing functionality, whether it is likely to be of general applicability or useful only for a single application, and whether the registration description is clear.

IANA must only accept registry updates from the Designated Experts and should direct all requests for registration to the review mailing list.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

11.1. JSON Web Proof Header Parameters Registry

This specification establishes the IANA "JSON Web Proof Header Parameters" registry for Header Parameter names, under the "JSON Object Signing and Encryption (JOSE)" registry group. The registry records the Header Parameter name and a reference to the specification that defines it. The same Header Parameter name can be registered multiple times, provided that the parameter usage is compatible between the specifications. Different registrations of the same Header Parameter name will typically use different Header Parameter Usage Locations values.

11.1.1. Registration Template

Header Parameter Name: The descriptive name of the parameter. (e.g., "Key Identifier").

Header Parameter JSON Label: The string label requested within a JSON context. (e.g., kid). Because a core goal of this specification is for the resulting representations to be compact, it is RECOMMENDED that the label be short -- not to exceed 8 characters without a compelling reason to do so. This label is case sensitive, and it is RECOMMENDED to avoid upper-case characters. Labels may not match another registered names in a case-insensitive manner unless the Designated Experts state that there is a compelling reason to allow an exception. This registry value SHOULD be supplied, but MAY be omitted if this Header Parameter will never be formatted as JSON.

Header Parameter CBOR Label: The string or integer label requested within a CBOR context (e.g., 2). This label may not match other integer values, match other string values in a case-insensitive manner, or be a differing string value from the JSON label unless the Designated Experts state that there is a compelling reason to allow an exception.

Header Parameter Usage Location(s): The Header Parameter usage locations, which should be one or more of the values Issued or Presented. Other values may be used with the approval of a Designated Expert.

Change Controller: For IETF Stream RFCs, list the IETF. For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s): Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

11.1.2. Initial Registry Contents

This section registers the Header Parameters defined in Section 5.2 in this registry.

11.1.2.1. Algorithm Header Parameter

* Header Parameter Name: Algorithm
* Header Parameter JSON Label: alg
* Header Parameter CBOR Label: 1
* Header Parameter Usage Location(s): Issued, Presented
* Change Controller: IETF
* Specification Document(s): Section 5.2.1 of this specification

11.1.2.2. Key ID Header Parameter

- Header Parameter Name: Key Identifier
- Header Parameter JSON Label: kid
- Header Parameter CBOR Label: 2
- Header Parameter Usage Location(s): Issued, Presented
- Change Controller: IETF
- Specification Document(s): Section 5.2.2 of this specification

11.1.2.3. Type Header Parameter

- Header Parameter Name: Type
- Header Parameter JSON Label: typ
- Header Parameter CBOR Label: 3
- Header Parameter Usage Location(s): Issued, Presented
- Change Controller: IETF
- Specification Document(s): Section 5.2.3 of this specification

11.1.2.4. Critical Header Parameter

- Header Parameter Name: Critical
- Header Parameter JSON Label: crit
- Header Parameter CBOR Label: 4
- Header Parameter Usage Location(s): Issued, Presented
- Change Controller: IETF
- Specification Document(s): Section 5.2.4 of this specification

11.1.2.5. Issuer Header Parameter

- Header Parameter Name: Issuer
- Header Parameter JSON Label: iss
- Header Parameter CBOR Label: 5
- Header Parameter Usage Location(s): Issued, Presented
- Change Controller: IETF
- Specification Document(s): Section 5.2.8 of this specification

11.1.2.6. Audience Header Parameter

- Header Parameter Name: Audience
- Header Parameter JSON Label: aud
- Header Parameter CBOR Label: 6
- Header Parameter Usage Location(s): Presented
- Change Controller: IETF
- Specification Document(s): Section 5.2.9 of this specification

11.1.2.7. Nonce Header Parameter

- Header Parameter Name: Nonce

* Header Parameter JSON Label: nonce
* Header Parameter CBOR Label: 7
* Header Parameter Usage Location(s): Presented
* Change Controller: IETF
* Specification Document(s): Section 5.2.10 of this specification

11.1.2.8. Issuer Ephemeral Key Header Parameter

* Header Parameter Name: Issuer Ephemeral Key
* Header Parameter JSON Label: iek
* Header Parameter CBOR Label: 8
* Header Parameter Usage Location(s): Issued
* Change Controller: IETF
* Specification Document(s): Section 5.2.5 of this specification

11.1.2.9. Holder Presentation Key Header Parameter

* Header Parameter Name: Holder Presentation Key
* Header Parameter JSON Label: hpk
* Header Parameter CBOR Label: 9
* Header Parameter Usage Location(s): Issued
* Change Controller: IETF
* Specification Document(s): Section 5.2.6 of this specification

11.1.2.10. Holder Presentation Algorithm Header Parameter

* Header Parameter Name: Holder Presentation Algorithm
* Header Parameter JSON Label: hpa
* Header Parameter CBOR Label: 10
* Header Parameter Usage Location(s): Issued
* Change Controller: IETF
* Specification Document(s): Section 5.2.7 of this specification

11.2. Media Type Registry

11.2.1. Registry Contents

This section registers the application/jwp media type [RFC2046] in the IANA "Media Types" registry [IANA.MediaTypes] in the manner described in [RFC6838], which can be used to indicate that the content is a JWP using the JWP Compact Serialization.

11.2.1.1. The application/jwp Media Type

* Type name: application
* Subtype name: jwp
* Required parameters: n/a
* Optional parameters: n/a

* Encoding considerations: 8bit; application/jwp values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') and tilde ('~') characters
* Security considerations: See Section 10 of this specification
* Interoperability considerations: n/a
* Published specification: this specification
* Applications that use this media type: TBD
* Fragment identifier considerations: n/a
* Additional information:
  - Magic number(s): n/a
  - File extension(s): n/a
  - Macintosh file type code(s): n/a
* Person & email address to contact for further information: Michael B. Jones, michael_b_jones@hotmail.com
* Intended usage: COMMON
* Restrictions on usage: none
* Author: Michael B. Jones, michael_b_jones@hotmail.com
* Change Controller: IETF
* Provisional registration? No

11.2.1.2. The application/cwp Media Type

* Type name: application
* Subtype name: cwp
* Required parameters: n/a
* Optional parameters: n/a
* Encoding considerations: 8bit; application/cwp values are represented as a CBOR data item.
* Security considerations: See Section 10 of this specification
* Interoperability considerations: n/a
* Published specification: this specification
* Applications that use this media type: TBD
* Fragment identifier considerations: n/a
* Additional information:
  - Magic number(s): n/a
  - File extension(s): n/a
  - Macintosh file type code(s): n/a
* Person & email address to contact for further information: Michael B. Jones, michael_b_jones@hotmail.com
* Intended usage: COMMON
* Restrictions on usage: none
* Author: Michael B. Jones, michael_b_jones@hotmail.com
* Change Controller: IETF
* Provisional registration? No

11.3. Structured Syntax Suffix Registry

11.3.1. Registry Contents

This section registers the following entries in the IANA "Structured Syntax Suffix" registry [IANA.StructuredSuffix] in the manner described in [RFC6838].

11.3.1.1. The +jwp Structured Syntax Suffix

* Name: JSON Web Proof (JWP)
* +suffix: +jwp
* References: Section 7.1 of this specification
* Encoding considerations: binary; JWP values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') and tilde ('~') characters
* Interoperability considerations: n/a
* Fragment identifier considerations: n/a
* Security considerations: See Section 10 of this specification
* Contact: Michael B. Jones, michael_b_jones@hotmail.com
* Author/Change controller: IETF

11.3.1.2. The +cwp Structured Syntax Suffix

* Name: CBOR Web Proof (CWP)
* +suffix: +cwp
* References: Section 7.1 of this specification
* Encoding considerations: 8bit; CWP values are represented as a CBOR data item.
* Interoperability considerations: n/a
* Fragment identifier considerations: n/a
* Security considerations: See Section 10 of this specification
* Contact: Michael B. Jones, michael_b_jones@hotmail.com
* Author/Change controller: IETF

12. References

12.1. Normative References

[I-D.ietf-jose-json-proof-algorithms]
Jones, M. B., Waite, D., and J. Miller, "JSON Proof Algorithms", Work in Progress, Internet-Draft, draft-ietf-jose-json-proof-algorithms-latest, <https://datatracker.ietf.org/doc/html/draft-ietf-jose-json-proof-algorithms>.

[I-D.ietf-jose-json-proof-token]
Jones, M. B., Waite, D., and J. Miller, "JSON Proof Token and CBOR Proof Token", Work in Progress, Internet-Draft, draft-ietf-jose-json-proof-token-latest, <https://datatracker.ietf.org/doc/html/draft-ietf-jose-json-proof-token>.

[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>.

[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>.

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

[RFC9596] Jones, M.B. and O. Steele, "CBOR Object Signing and Encryption (COSE) "typ" (type) Header Parameter", RFC 9596, DOI 10.17487/RFC9596, June 2024, <https://www.rfc-editor.org/info/rfc9596>.

12.2. Informative References

[ECMAScript]
Ecma International, "ECMAScript Language Specification, 5.1 Edition", ECMA 262, June 2011, <https://262.ecma-international.org/5.1/>.

[I-D.irtf-cfrg-bbs-signatures]
Looker, T., Kalos, V., Whitehead, A., and M. Lodder, "The BBS Signature Scheme", Work in Progress, Internet-Draft, draft-irtf-cfrg-bbs-signatures-10, 8 January 2026, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-bbs-signatures-10>.

[IANA.CoAP.Formats]
IANA, "CoAP Content-Formats",
<https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats>.

[IANA.JWT] IANA, "JSON Web Token",
<https://www.iana.org/assignments/jwt>.

[IANA.MediaTypes]
IANA, "Media Types",
<https://www.iana.org/assignments/media-types>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[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>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[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>.

[RFC9338] Schaad, J., "CBOR Object Signing and Encryption (COSE): Countersignatures", STD 96, RFC 9338, DOI 10.17487/RFC9338, December 2022, <https://www.rfc-editor.org/info/rfc9338>.

[VC-DATA-MODEL-2.0] Sporny, M., Jr, T. T., Herman, I., Cohen, G., and M. B. Jones, "Verifiable Credentials Data Model v2.0", 15 May 2025, <https://www.w3.org/TR/vc-data-model-2.0>.

Appendix A. Acknowledgements

This work was incubated in the DIF Applied Cryptography Working Group (https://identity.foundation/working-groups/crypto.html).

We would like to thank Brent Zundel and Emil Lundberg for their valuable contributions to this specification.

Appendix B. Document History

[[To be removed from the final specification]]

-13

* Examples are now built deterministically (using RFC 6979 deterministic ECDSA and seeded random number generation for BBS)

-12

* IANA Considerations section changes from IANA Early Review

-11

* Change Issuer Protected Header to Issuer Header
* Change Presentation Protected Header to Presentation Header
* Clarify usage of headers and header parameters

-10

* Replaced application/jwp+cbor with application/cwp.
* Registered +cwp structured syntax suffix and simplified +jwp suffix.
* Expanded descriptions of and rationale for the serializations.
* Payload nomenclature was clarified by adding the concept of payload slots. The issued form and presented form have a certain number of ordered payload slots, and a presentation may choose to omit information from a slot.

* Recommendation for Issuer Protected Header is to make the header static across issued JWPs if possible, to prevent unintentional correlation by verifiers
* Cleaned up the text around algorithmic support for additional proofs of knowledge outside of payload disclosure (e.g., range proofs)
* Clarify that a proof is made up of multiple octet strings
* (CDDL) The protected header definitions now reference empty_or_serialized_map properly
* (CDDL) payloads is now PayloadSlots, and represents you need at least one slot
* (CDDL) The issuer PayloadSlots contains payload, which is not omittable
* (CDDL) The presented form likewise references disclosable_payload, which is nillable to represent omission
* (CDDL) Proof is capitalized to match, and properly represents you need at least one bstr
* (CDDL) header_map, label and value definitions imported from RFC 9052
* Receive examples of additional types of algorithm-specific proofs from JPT
* Added Holder Presentation Algorithm (hpa) parameter to support the holder presenting using a different key type and algorithm than the issuer in SU and MAC family algorithms
* Change "proof_key" to "iek" and "presentation_key" to "hpk", both to shorten their JSON labels and to clarify owner/purpose.

-09

* Removed JSON serialization.
* Corrected informative reference to the IANA JWT registry.

-08

* Corrected instances of identifier proofs to proof.
* Added reference to [RFC9596] for COSE "typ" header parameter.
* Made some additional references normative.

-07

* Changing primary editor
* Broad changes to define a CBOR serialization, which leverages the new CBOR Protected Headers
* Deemphasis of JSON in some parts of the document to represent CBOR alternatives

* Rewrite Header Parameter parsing requirements for JSON to emphasize preference to fail on duplicate headers (to match CBOR behavior). Last-encountered remains as an option to match ECMA definition.
* Add option to use CoAP Formats for typ in CBOR mode to match COSE, as a compact indirection over full media types.
* Modify IANA registry template to account for CBOR Labels in header parameters
* Add application/jwp+cbor media type
* Modify example generation to use proof_key and presentation_key names
* Change proof_jwk to proof_key and presentation_jwk to presentation_key to better represent that the key may be JSON or CBOR-formatted.
* Moved the registry for proof_key and presentation_key to JWP where they are defined. Consolidated usage, purpose, and requirements from algorithm usage under these definitions.
* Clarified that proof_key and presentation_key are required by particular algorithms and are not more generally required for issued and presented JWPs.
* Move claims to JPT to live beside cid, and renumber CBOR labels so that they may be adjacent

-06

* Update reference to new repository home.
* Fixed #83: Added encrypted JWPs.
* Added additional clarification around the compact and JSON serializations
* Added text around fully detached payloads

-05

* Clarify the use of multiple octet strings in presentation proofs.
* Update BBS algorithm example in JSON serialization to show the proof as an array with a single octet string.
* Move single-use example appendix from JWP to JPA.
* Registered +jwp structured syntax suffix.

-04

* Refactoring figures and examples to be built from a common set across all three documents.

-03

* Improvements resulting from a full proofreading.
* Populated IANA Considerations section.

* Specified JWP Header Parameters.
* Specified representation of zero-length disclosed payloads for the compact serialization.
* Specified that algorithms may supply multiple octet strings for the proof, which are separated by ~ characters in the compact serialization.
* Updated to use BBS draft -05.
* Added Terminology Section.

-02

* Update reference to current BBS algorithm

-01

* Correct cross-references within group.

-00

* Created initial working group draft based on draft-jmiller-jose-json-web-proof-01

Authors' Addresses

David Waite
Ping Identity
Email: dwaite+jwp@pingidentity.com

Michael B. Jones
Self-Issued Consulting
Email: michael_b_jones@hotmail.com
URI: https://self-issued.info/

Jeremie Miller
Ping Identity
Email: jmiller@pingidentity.com