

Proposed revisions to JSON-LD signature formats

Technical Specification

by Kevin Poulsen

Last Updated August 3rd, 2018

Status: under review

- C O N F I D E N T I A L -

Copyright © 2018 by The Authenticity Institute All Rights Reserved

Scope

This specification proposes revisions to the draft Linked Data Signatures 1.0 document to minimize ill-advised modifications to signed content. In addition, it proposes a method of implementing signature chains that proves knowledge of prior signatures.

By extension, this also proposes revisions to Verifiable Credential formats.

External documents

Linked Data Signatures 1.0 - <https://w3c-dvcg.github.io/ld-signatures/>

Verifiable Claims/Credentials Data Model and Representations - <https://www.w3.org/2018/03/13-vcwg-minutes.html>

EXAMPLE 5: A more complex verifiable claim

This example is from the Verifiable Claims Data Model and Representations at <https://www.w3.org/TR/verifiable-claims-data-model/>

When looking at this, the question comes to mind: What exactly is signed? A digital signature is supposed to indicate that the content hasn't changed since signed. To recreate the hash, you need to know exactly which portion of this credential is signed:

```
{
  "@context": [
    "https://w3id.org/identity/v1",
    "https://w3id.org/security/v1"
  ],
  "id": "http://example.gov/credentials/3732",
  "type": ["Credential", "PassportCredential"],
  "name": "Passport",
  "issuer": "https://example.gov",
  "issued": "2010-01-01",
  "claim": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "name": "Alice Bobman",
    "birthDate": "1985-12-14",
    "gender": "female",
    "nationality": {
      "name": "United States"
    },
    "address": {
      "type": "PostalAddress",
      "addressStreet": "372 Sumter Lane",
      "addressLocality": "Blackrock",
      "addressRegion": "Nevada",
      "postalCode": "23784",
      "addressCountry": "US"
    },
    "passport": {
      "type": "Passport",
      "name": "United States Passport",
      "documentId": "123-45-6789",
      "issuer": "https://example.gov",
      "issued": "2010-01-07T01:02:03Z",
      "expires": "2020-01-07T01:02:03Z"
    }
  },
  "signature": {
    "type": "LinkedDataSignature2015",
    "created": "2016-06-21T03:40:19Z",
    "creator": "https://example.com/jdoe/keys/1",
    "domain": "json-ld.org",
    "nonce": "783b4dfa",
    "signatureValue":
      "Rxi7Kb/tDbGHFAs6ddHjVLsHDiNyYzxs2MPmNG8G47oS06N8i0D
      is5mUePiZII4+p/ewc0TjvH7aJxnKEePC09IrlqaHn01TfmTut2r
      vXxESJNzur0qoNq2yXl+TqUwmDXoHZF+jQ7gCsmYqTWhhsG5uf09
      oydMzPoCb9ibsNk="
  }
}
```

EXAMPLE 2: A simple signed Linked Data document

This example is from the Draft Community Group Report 27 February 2018 at <https://w3c-dvcg.github.io/ld-signatures/>

Just looking at this, the same vagueness is present. However, this document is specific about how the signature hash is initially created then re-created in sections 7.1 and 7.2

```
{
  "@context": "https://w3id.org/identity/v1",
  "title": "Hello World!",
  "proof": {
    "type": "RsaSignature2018",
    "creator": "https://example.com/i/pat/keys/5",
    "created": "2017-09-23T20:21:34Z",
    "domain": "example.org",
    "nonce": "2bbgh3dgjg2302d-d2b3gi423d42",
    "proofValue": "eyJ0eXAiOiJK...gFWFOEjXk"
  }
}
```

From section 7.1:

1. Create a copy of document, hereafter referred to as output.
 2. Generate a canonicalized document by canonicalizing document according to a [canonicalization algorithm](#) (e.g. the GCA2015 [RDF-DATASET-NORMALIZATION] algorithm).
 3. Create a value `tbv` that represents the data to be signed, and set it to the result of running the [Create Verify Hash Algorithm](#), passing the information in options.
1. Digitally sign `tbv` using the `privateKey` and the the digital signature algorithm (e.g. JSON Web Signature using RSASSA-PKCS1-v1_5 algorithm). The resulting string is the `signatureValue`.
 2. Add a **signature node to output** containing a [linked data signature](#) using the appropriate type and `signatureValue` values as well as all of the data in the signature options (e.g. creator, created, and if given, any additional signature options such as `nonce` and `domain`).
 3. Return output as the [signed linked data document](#).

From section 7.2:

1. Get the [public key](#) by dereferencing its URL identifier in the `signature` node of the default graph of signed document. Confirm that the [linked data document](#) that describes the `public key` specifies its owner and that its owner's URL identifier can be dereferenced to reveal a bi-directional link back to the key. Ensure that the key's owner is a trusted entity before proceeding to the next step.
2. Let document be a copy of signed document.
3. **Remove any signature nodes from the default graph in document** and save it as signature.
4. Generate a canonicalized document by canonicalizing document according to the [canonicalization algorithm](#) (e.g. the GCA2015 [RDF-DATASET-NORMALIZATION] algorithm).
5. Create a value `tbv` that represents the data to be verified, and set it to the result of running the [Create Verify Hash Algorithm](#), passing the information in signature.
6. Pass the `signatureValue`, `tbv`, and the [public key](#) to the [signature algorithm](#) (e.g. JSON Web Signature using RSASSA-PKCS1-v1_5 algorithm). Return the resulting boolean value.

Proposed JSON-LD signature format

The problem with the instructions in sections 7.1 and 7.2 is that the signed content is modified by the signer. Then, the receiver needs to undo the modifications then canonicalize the content in order to verify the signature. In theory, this is doable as long as the instructions are very clear. However, this seems very ill-advised. It could create a situation where the signature is impossible to verify.

My initial reformatting of this took the entire “proof” or “signature” key-value pair out of the signed content. However, in the Linked Data Signatures 1.0, issue 6 states:

ISSUE 6

The signature parameters should be included as headers and values in the data to be signed.

Given the standards referenced above, the following format is proposed for JSON-LD Signatures:

1. The signed content is the value of the “signedContent” key-value pair.
2. The signature value is in the key-value pair immediately following.
3. The signature metadata is contained in the signed content

With this format, the signed content is NOT modified at all between the time the signature is created and the time the signature is verified.

```
{
  "@context": "?",
  "signedContent": {
    "@context": "https://w3id.org/identity/v1",
    "title": "Hello World!",
    "proofMetadata": {
      "type": "RsaSignature2018",
      "creator": "https://example.com/i/pat/keys/5",
      "created": "2017-09-23T20:21:34Z",
      "domain": "example.org",
      "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
    }
  },
  "proofValue": "eyJ0eXAiOiJK...gFWFOEjXk"
}
```

Q: If a time authority is used, would the “created” value be inside or outside of the signed content?

EXAMPLE 4: A signature chain in a Linked Data document

While working on this document, it occurred to me why the JSON-LD signatures were modifying the signed content in order to verify signatures: to enable signature sets.

The example below shows a signature chain. A signature set is very similar (example 3). This example is from the Draft Community Group Report 27 February 2018 at <https://w3c-dvcg.github.io/ld-signatures/>

In order to verify multiple signatures on the same content (not chained), any existing signature metadata needs to be removed, then the relevant signature metadata needs to be inserted. As bad as that is (it modifies the signed content), the alternative is to duplicate the entire signed content for each signature. That may be a more "correct" way to do it. But if efficiency is key, then a more compact way of verifying multiple signatures may be helpful.

```
{
  "@context": "https://w3id.org/identity/v1",
  "title": "Hello World!",
  "proofChain": [{
    "type": "RsaSignature2018",
    "creator": "http://example.com/i/pat/keys/5",
    "created": "2017-09-23T20:21:34Z",
    "domain": "example.org",
    "nonce": "2bbgh3dgjg2302d-d2b3gi423d42",
    "proofValue": "eyi0iJKJ0eXA...0EjgFWFXk"
  }, {
    "type": "RsaSignature2018",
    "creator": "http://bank.example.com/notary/keys/7f3j",
    "created": "2017-09-23T20:24:12Z",
    "domain": "example.org",
    "nonce": "83jj4hd62j49gk38",
    "proofValue": "eyi0iJJ0eXAK...EjXkgFWF0"
  }]
}
```

Proposed format for signature sets

The goal of this proposed format for signature sets is to:

Allow multiple signatures on the same content with minimal modification of the content after it has been signed. This is for situations where it's impractical to duplicate the entire content for each signature.

To verify multiple signatures, the metadata for each signature would be assigned to the proofMetadata key-value pair in the signedContent. Then, a hash of the signedContent would be compared to the decrypted proofValue.

```
{
  "@context": "?",
  "signedContent": {
    "@context": "https://w3id.org/identity/v1",
    "title": "Hello World!",
    "proofMetadata": {
      "type": "RsaSignature2018",
      "creator": "https://example.com/i/pat/keys/5",
      "created": "2017-09-23T20:21:34Z",
      "domain": "example.org",
      "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
    }
  },
  "proof": [{
    "proofMetadata": {
      "type": "RsaSignature2018",
      "creator": "https://example.com/i/pat/keys/5",
      "created": "2017-09-23T20:21:34Z",
      "domain": "example.org",
      "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
    },
    "proofValue": "eyJ0eXAi0iJK...gFWF0EjXk"
  }, {
    "proofMetadata": {
      "type": "RsaSignature2018",
      "creator": "https://example.com/i/pat/keys/5",
      "created": "2017-09-23T20:21:34Z",
      "domain": "example.org",
      "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
    },
    "proofValue": "eyJ0eXAi0iJK...gFWF0EjXk"
  }, {
    "proofMetadata": {
      "type": "RsaSignature2018",
      "creator": "https://example.com/i/pat/keys/5",
      "created": "2017-09-23T20:21:34Z",
      "domain": "example.org",
      "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
    },
    "proofValue": "eyJ0eXAi0iJK...gFWF0EjXk"
  }]
}
```

Q: With this technique, would the signedContent still need to be canonicalized?

Proposed format for signature chains

Signature chains are fundamentally different from signature sets. The signatures are to be applied in a certain order. This might imply that each subsequent signer is aware of the previous signature.

The example 4 above for signature chains allows for the order of the signatures to be verified by the "created" date/time. But, each signature doesn't prove knowledge of prior signatures. The following format:

1. eliminates the need to modify signed content in order to verify the signature chain and
2. proves knowledge of prior signatures at the time of each signing by nesting signatures. Prior signatures are signed.

This becomes less human-readable. So, the highlight colors were added to clearly show how each new signature brackets the prior signed content with signature.

```
{
  "@context": "?",
  "signedContent": {
    "@context": "?",
    "signedContent": {
      "@context": "?",
      "signedContent": {
        "@context": "https://w3id.org/identity/v1",
        "title": "Hello World!",
        "proofMetadata": {
          "type": "RsaSignature2018",
          "creator": "https://example.com/i/pat/keys/5",
          "created": "2017-09-23T20:21:34Z",
          "domain": "example.org",
          "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
        }
      }
    }
  },
  "proofValue": "eyJ0eXAi0iJK...gFWF0EjXk",
  "proofMetadata": {
    "type": "RsaSignature2018",
    "creator": "https://example.com/i/pat/keys/5",
    "created": "2017-09-23T20:21:34Z",
    "domain": "example.org",
    "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
  }
},
  "proofValue": "eyJ0eXAi0iJK...gFWF0EjXk",
  "proofMetadata": {
    "type": "RsaSignature2018",
    "creator": "https://example.com/i/pat/keys/5",
    "created": "2017-09-23T20:21:34Z",
    "domain": "example.org",
    "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
  }
},
  "proofValue": "eyJ0eXAi0iJK...gFWF0EjXk",
  "proofMetadata": {
    "type": "RsaSignature2018",
    "creator": "https://example.com/i/pat/keys/5",
    "created": "2017-09-23T20:21:34Z",
    "domain": "example.org",
    "nonce": "2bbgh3dgjg2302d-d2b3gi423d42"
  }
}
},
  "proofValue": "eyJ0eXAi0iJK...gFWF0EjXk"
}
```