JSON Object Signing and Encryption (JOSE)
Internet-Draft
Intended status: Informational
Expires: March 10, 2019

S. Erdtman
Spotify AB
A. Rundgren
Independent
M. Jones
Microsoft

September 6, 2018

Cleartext JSON Web Signature (JWS)
draft-erdtman-jose-cleartext-jws-01

Abstract

Cleartext JSON Web Signature (JWS) is a means of signing JSON objects directly without representing the JSON to be signed in a non-JSON representation, such as base64url-encoded JSON. The signature and information about the signature is added to the JSON object when it is signed. The signature calculation for signing the JSON object uses the JSON canonicalization defined by [I-D.rundgren-json-canonicalization-scheme]. Cleartext JWS builds on the JWS, JWA, and JWK specifications, reusing data structures and semantics from these specifications, where applicable.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 10, 2019.

Table of Contents

1. Introduction

Cleartext JSON Web Signature (JWS) represents a signed JSON object directly as a JSON object [RFC8259], without representing the JSON to be signed in a non-JSON representation, such as base64url-encoded JSON. The signature and information about the signature is added to the JSON object when it is signed. The signature calculation for signing the JSON object uses the JSON canonicalization defined by [I-D.rundgren-json-canonicalization-scheme]. By including the

signature information in the JSON object to be signed, it is easy to inspect data in transit and when archived, integrity can be guaranteed.

Cleartext JWS builds on the JWS [RFC7515], JWA [RFC7518], and JWK [RFC7517] specifications, reusing data structures and semantics from these specifications, where applicable. Cryptographic algorithm identifiers used by this specification come from the IANA "JSON Web Signature and Encryption Algorithms" registry [IANA.JOSE.Algorithms].

There are three essential differences between Cleartext JWS and JWS:

- Cleartext JWS can only sign JSON objects, rather than arbitrary data.
- Cleartext JWS signature information is included within the signed data.
- Cleartext JWS depends on predictable JSON Serialization, rather than base64url-encoding the data to be signed.

The table below is a comparison of JWS and Cleartext JWS:

	JWS	Cleartext JWS
Data to be Signed	Arbitrary data	JSON or JavaScript objects
Encoding of Signed Data	Base64url	None
Encoding of Header Parameters	Base64url	None
URL Friendly	Core feature	Out of scope

In the following example, note that the signature information is included in the JSON object. The members in the "__cleartext_signature" object are the JWS Header Parameters for the signature. The "signature" member contains the base64url-encoded signature value. (Line breaks within values are for display purposes only.)

```
{
  "iss": "joe",
  "exp": 1300819380,
  "escapeMe": "\u20ac$\u000F\u000aA'\u0042\u0022\u005c\\\"\/",
  "numbers": [1e+30,4.5,6],
  "__cleartext_signature": {
    "alg": "ES256",
    "kid": "example.com:p256",
    "signature": "pXP0GFHms0SntctNk1G1pHZfccVYdZkmAJktY_hpMsI
                  AckzX7wZJIJNlsBzmJ1_7LmKATiW-YHHZjsYdT96JZw"
  }
}
```

The key in Appendix A.3 can be used for verifying the example signature.

Note: Recreating the example signature using the example private key would normally result in a different "signature" value since ECDSA includes random data in the signature calculation.

2. 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.

3. The Signature Object

When signing JSON data with Cleartext JWS, a JSON object with the JWS Header Parameters is created and placed within the JSON data to be signed. In addition to the already defined JWS Header Parameters, Cleartext JWS defines two new Header Parameters: "signature" for the base64url-encoded signature value and "signers" to support multiple signers within the same signature object.

The identifier for the Cleartext Signature Object in the JSON data to be signed MUST be "__cleartext_signature", unless the application specifies that a different identifier is to be used.

3.1. Signature Scope

The scope of a signature (the data that is actually signed) comprises all values including child objects of the signature object except for the "signature" member. If multiple signers are present, only the data pertaining to all signers and the data specific to that signer

are included (but not the data specific to other signers). See Section 4.3 for more about the multiple signatures case.

3.2. The "signature" Header Parameter

The "signature" Header Parameter contains the base64url-encoded JWS Signature as a string.

3.3. The "signers" Header Parameter

The optional "signers" Header Parameter contains an array of sets of Header Parameters that are specific to each signer, including the "signature" value for each signer. See Section 4.3 for more about the multiple signatures case.

4. Producing and Consuming Cleartext JWSs

4.1. Message Signature or MAC Computation

To create a Cleartext JWS, the following steps are performed. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps.

1. Create the application specific JSON object to be signed. In this step the JSON should not be canonicalized.

2. Create the "__cleartext_signature" object with the Header Parameters to be used and add it as a top-level member to the JSON object to be signed with the key "__cleartext_signature".

3. Canonicalize the JOSN object to be signed using the canocalization process defined by [I-D.rundgren-json-canonicalization-scheme]. Let the output of the canonicalization be the cleartext input to the signature algorithm.

4. Compute the JWS Signature in the manner defined for the particular algorithm being used over the canonicalize JSON object to be signed. The "alg" (algorithm) Header Parameter MUST be present in the "__cleartext_signature" member, with the algorithm value accurately representing the algorithm used to construct the JWS Signature.

5. Add the "signature" member to the signature object (__cleartext_signature) within the original application JSON object. with the value BASE64URL(JWS Signature).

4.2. Message Signature or MAC Validation

When validating a Cleartext JWS, the following steps are performed. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps. If any of the listed steps fails, then the input MUST be rejected.

When there are multiple JWS Signature values, it is an application decision which of the JWS Signature values must successfully validate for the Cleartext JWS to be accepted. In some cases, all must successfully validate, or the Cleartext JWS will be considered invalid. In other cases, only a specific JWS Signature value needs to be successfully validated. However, in all cases, at least one JWS Signature value MUST successfully validate, or the Cleartext JWS MUST be considered invalid.

1. Parse the application JSON data, including the signature object.

2. Verify that the implementation understands and can process all fields that it is required to support, whether required by this specification, by the algorithm being used, or by the "crit" Header Parameter value, and that the values of those parameters are also understood and supported.

3. Save and remove the "signature" member from the signature object (__cleartext_signature) and base64url-decode the encoded representation of the JWS Signature.

4. Canonicalize the signed object, including the signature object (__cleartext_signature), by following the rules by [I-D.rundgren-json-canonicalization-scheme] and let the output be input to the signature algorithm.

5. Validate the JWS Signature against the JWS Signing Input, i.e., the canonicalize data, in the manner defined for the algorithm being used, which MUST be accurately represented by the value of the "alg" (algorithm) Header Parameter value, which MUST be present. Record whether the validation succeeded or not.

6. Return a result indicating whether or not the Cleartext JWS was successfully validated.

7. For later validation of the signed JSON object, put the "signature" member back into the signature object (__cleartext_signature) within the application JSON object.

4.3. Multiple Signatures

Multiple signers using different keys can independently add signatures to a JSON object in the manner described in this section.

The signature procedure is essentially the same as for single signatures but also includes the following:

- There MUST be an additional JWS Header parameter "signers", holding an array of signature objects.
- Each signature requires its own canocalization process. During this process, the signature objects for other signatures MUST be (temporarily) removed.
- The canonicalized data in the "signers" value MUST include the array brackets ([]) containing the data specific to this signature but MUST NOT include the data for other signatures. The resulting array will be single-valued, with no commas separating additional elements.
- A given Header Parameter MUST NOT occur in both the top-level signature object and a signature object within the "signers" value. Any Header Parameter occurring in the top-level signature object applies to all signatures.
- A signature object is equivalent to an ordinary signature object, but MAY exclude the "alg" Header parameter if it is present in the top-level signature object itself. If in the top-level signature object, all enclosed signature objects MUST use the same algorithm as well as not including the "alg" Header parameter. See Appendix A.1 for an example.
- Likewise, if a "crit" Header parameter is specified in the top-level signature object, it MUST be applied to all signature objects and MUST NOT be present in them individually. See Appendix A.2 for an example.

The following example shows a multiply signed object:

```
{
  "iss": "joe",
  "exp": 1300819380,
  "escapeMe": "\u20ac$\u000F\u000aA'\u0042\u0022\u005c\\\"\/",
  "numbers": [1e+30,4.5,6],
  "__cleartext_signature": {
    "signers": [{
      "alg": "ES256",
      "kid": "example.com:p256",
      "signature": "83gr5rmjKgngLTaPpxuQWiZaQmlQ555jLHNcZLmcBpg
                    X7JZLeqrNhIrQRg3jTsNwh1RuibDYBzCsaxVUkhGEKg"
    },{
      "alg": "RS256",
      "kid": "example.com:r2048",
      "signature": "PVQeL8XtjnetambQe98FuMBDuijwWTIFXouyNjL8WX0
                    WvamWkHjv34Iz8VOHHWr9w8t14FXJJuQ22j-h5BR7qP
                    xE7cBVS8XSltR7VvcNidfn-r-TtAVwDwn7Iz_Gk-RI7
                    QIv4ctbreYt1myG64Ikw38OEmNURCxzf9h9w3tvA3R8
                    ZE3MYgELFaQRowSW92JC1HhGZRijzHoIzvH6l_GULP_
                    hf7kggwFNtRrzN8DLXbhBhGaoP-O0cNZsCWY2hbNU6L
                    7km6bdrqHdq88DSOEGg_-5T6qUsIAYbmCgUK7XBi2q-
                    DRPQZYnxr5570mj9Nkh0hpZ-VfAC2ftbzxFAB7ZYg"
    }]
  }
}
```

The ECDSA signature can be validated using the key in Appendix A.3 and the RSA signature can be validated using the key in Appendix A.5.

5. IANA Considerations

5.1. JSON Header Parameters Registry

This section registers the following Header Parameters in the IANA "JSON Web Signature and Encryption Header Parameters" registry [IANA.JOSE.HeaderParameters].

5.1.1. Registry Contents

- Header Parameter Name: "signature"
- Header Parameter Description: The base64url-encoded signature value
- Header Parameter Usage Location(s): "Cleartext JWS"
- Change Controller: IESG

- Specification Document(s): Section 3.2
- Header Parameter Name: "signers"
- Header Parameter Description: List of signature objects, each with a set of Header Parameters and a signature value
- Header Parameter Usage Location(s): "Cleartext JWS"
- Change Controller: IESG
- Specification Document(s): Section 3.3

6. Security Considerations

The same security considerations apply to this specification as do for JWS [RFC7515].

7. References

7.1. Normative References

[I-D.rundgren-json-canonicalization-scheme]
Rundgren, A., "JSON Canonicalization Scheme (JCS)", draft-rundgren-json-canonicalization-scheme-01 (work in progress), June 2018.

[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>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

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

[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>.

7.2. Informative References

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

[IANA.JOSE.HeaderParameters] IANA, "JSON Web Signature and Encryption Header Parameters", <https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-header-parameters>.

Appendix A. Test Vectors

This section contains a set of test vectors. (Line breaks within values are for display purposes only.)

A.1. Multiple Signatures with Top-Level "alg" Header Parameter

```
{
  "iss": "joe",
  "exp": 1300819380,
  "escapeMe": "\u20ac$\u000F\u000aA'\u0042\u0022\u005c\\\"\/",
  "numbers": [1e+30,4.5,6],
  "__cleartext_signature": {
    "alg": "ES256",
    "signers": [{
      "kid": "example.com:p256",
      "signature": "En1Iyg45g1HBsxYdu-SR1fjt2nixOEtMWRrVA9EO7N8QfZtrs
                    cEfNl0tkIthqKWXGGPNmWKSw9Bc0Cj6kjHMKA"
    },{
      "kid": "example.com:p256-2",
      "signature": "RVNSVosrweujm36TDq9661oZiORdPMe-A-v-TADFO_mm6ss96
                    QnVP_BqT9kIc7nSlW7lOeMWk5Tq4kL4d3M_Mw"
    }]
  }
}
}
```

The first signature can be verified using the key in Appendix A.3 and the second signature can be verified using the key in Appendix A.4.

A.2. Multiple Signatures with Top-Level "crit" Header Parameter

```
{
  "iss": "joe",
  "exp": 1300819380,
  "escapeMe": "\u20ac$\u000F\u000aA'\u0042\u0022\u005c\\\"\/",
  "numbers": [1e+30,4.5,6],
  "__cleartext_signature": {
    "crit": ["otherExt","https://example.com/extension"],
    "signers": [{
      "alg": "ES256",
      "kid": "example.com:p256",
      "otherExt": "Other Data",
      "signature": "S9PqQU5z5zThIGUvErzf7oo8EetiUFEI1v8flisWJzw0HqqY-
                    OuT_pDq1rG4gsSRAFjrazurl4NGyyxcPfeXzw"
    },{
      "alg": "RS256",
      "kid": "example.com:r2048",
      "otherExt": "Cool Stuff",
      "https://example.com/extension": {
        "life-is-great": true
      },
      "signature": "O-cnYTtgvyGmgX1YMQkcnRE0lnBw1EduMNVpdblKP-Iy0S143
                    BBvXrCQoEW9oTkQm7X9wkJoohWQyU4qvojoxQxmf6GQ0tEXEI
                    HqN7ixkPh_3ySXTl-gKTPiA5UL-GV44AS-k6N71qp8XhLORmU
                    m68UlTWBZaOXL0JTGjsCyGpuwNiAQbx39ZbjabvGq4NfpPIQC
                    2yjx_SKoPMiia54Mp0hz8U_S3oyAmHrG2mKFYrJ7k43aeDHK1
                    RNRu8XrW2w-Ffh4KigpClAq4q272ZSsjizfYPPjW3gqInjMZz
                    Qd8yZj5Bi5vCDcBOEKZMDoog-UzIy8SbZNl85TlkhK70oNRQ"
    }]
  }
}
```

The first signature can be verified using the key in Appendix A.3 and the second signature can be verified using the key in Appendix A.5.

A.3. Elliptic Curve Key "example.com:p256"

Elliptic Curve private key, represented as a JWK:

```
{
  "kid": "example.com:p256",
  "kty": "EC",
  "crv": "P-256",
  "x": "censDzcMEkgiePz6DXB7cDuwFemshAFR90UNVQFCg8Q",
  "y": "xq8rze6ewG0-eVcSF72J77gKiD0IHnzpwHaU7t6nVeY",
  "d": "nEsftLbi5u9pI8B0-drEjIuJzQgZie3yeqUR3BwWDl4"
}
```

A.4. Elliptic Curve Key "example.com:p256-2"

Elliptic Curve private key, represented as a JWK:

```
{
  "kid": "example.com:p256-2",
  "kty": "EC",
  "crv": "P-256",
  "x": "RgdKcWxBsnqeryzoEv3B5KE9qAQc-nBZEV_A23uQoPs",
  "y": "73UtZIe1Qfil1WM9Hq1ZiPXWnI1Tu7N__goVvTyjURk",
  "d": "2jlPu5M9ISDkk-cpPgj6XGvZMhrFUfPujtQy2LtMOss"
}
```

A.5. RSA Key "example.com:r2048"

RSA private key, represented as a JWK:

```
{
  "kid": "example.com:r2048",
  "kty": "RSA",
  "n": "hFWEXArvaZEpSP5qNX7x4C4Hl28GJQTNvnDwkfqiWs63kXbdyPeS06bz6GnY3
        tfQ_093nGauWsimqKBmGAGMPtsV83Qxw1OIeO4ujbIIb9pema0qtVqs0MWlHx
        klZGFkYfAmbuEUFxYDeLDHe0bkkXbSlB7_t8pCSvc8HLgHjEQjYOlFRwjR0D-
        uLo-xgsCbpmCtYkB5lcT_zFgpRgY4zJNLSv7GZiz2S4Fc5ArGjd34lL47-L8b
        ozuYjqNOv9sqX0Zgll5XaJ1ndvr7UqZu1xQFgm38reoM3IarBP_SkEFbt_v9i
        ak602VO3k28fQhMaocP7JWR2YLT3kZM0-WTFw",
  "e": "AQAB",
  "d": "Q6iBYpnIrB2mkQZagP1lZuvBv9_osVaSZpLRvKD7DxhvbDTs0coaTJIoVCSB1
        _VZip8zlUg-TnYWF1Liv9VSwfQ7ddxrcOUtej60mId0ntNz2HhbxJsWjiru8E
        ZoArl0nEovLDNxlRgRMEyZwOKPC_xHT6nFrk7_s9pR5pEEcubGLAVBKnLCoPd
        Lr-CBjCvWfJo73W5AZxoSb8MdWQOi5viXHURpr1Y_uBRsMuclovM56Vt05etM
        sB1AbcTLUDwAuYrZWa1c08ql60ft7b3v6Q_rCL7EHtFU3PHAuP0mV7tM5BfAP
        f4T0g9pbr4GOw7eqQCiYgPFE7gmCR_PDxv5YQ",
  "p": "6DIM343hAtj1hQprJaVQ3T8YeIytIQ7Ma544C0A8BX-irjJfARy4fAlTSyBFe
        auZ0WdbMGtKpAIgNVmfCfuP7W1bXw7UaxpqsQlbw54K1VtBs8xG-lee_2YQ3l
        UlIiC1at6L0jxWYNkvp-LIfU2F5ZQir5ZWVXwgdMcgoNBABMc",
  "q": "keacq0goV7pAtG2h33OAk-XOSclIF1agvEMMOKuud5V-vGQ6OaYldlYqZmSGg
        F7RVlX0GZO70nPqatjd2G-tI8wEq5K_xmLQurUPFW8g___z0CTgJ62KbjFxCt
        Gny5rsObX9im6cCc_EOtWZRaApzO8ykxfo1QcEjT4k1na7DzE",
  "dp": "nPmJPnFal2Q5x_GdMlwq6QhI8OaZ_OlWRcM3PFP2v_jj8ERZehUCm8hqKTXu
         Ai2C1dC8E2XVlj9hqu-l10fcq7Tsurz52laHnpwnD35-8HK7XmRR79jgwuUr
         rkN90S6vt0ow2La15s-tqiBlTmDkjqqxMGfAghZiktA0PMPNI-0",
  "dq": "D3c1lkZw2FPK9hVE-m3A7GyIwHOQq8CoCyzER-GS_eQf6hJpxaCiCfg6SF5R
         j5v9brxvwqJRX46gA7F3WrED1m6S9Cj7ISlqXNBCiBAenGRiUOcHx8zyhpnB
         FNeChOeoMLnk5V6yNawLbf0kYSgIJkwYvVTkfmhfCCXVO9KcI5E",
  "qi": "wV0NzfCakfog1NFjtPzcga1MtkpizgPkxcP9LjNdvXW2YQZhM6GIEGjsu3iv
         TrHrrM-4_bTQHOoTtfIY7wdqBKlwQTJOI0dH9FbNJ4ecGojRwgv83TN8aNKh
         17Tt44jI5oibs2P-31B_VW9R1wwhnnOuCYpABfoSbtHIoCRme5I"
}
```


Appendix B. Acknowledgements

This document builds on the work done in the JOSE working group, so a big thanks goes out to all involved in that work. It is specifically inspired by JWS, so special thanks are due to the authors of that document, Michael B. Jones, John Bradley, and Nat Sakimura.

Appendix C. Open Issues

The following open issues remain to be addressed in this specification.

- The signature creation and validation steps for the multiple signatures case needs to be added to Section 4.1 and Section 4.2.

Appendix D. Document History

[[to be removed by the RFC Editor before publication as an RFC]]

-01

- Changed canocalization from ES6 serialization to [I-D.rundgren-json-canonicalization-scheme].
- "signature object" is now used consistently through out the specification.

-00

- Initial version.

Authors' Addresses

Samuel Erdtman
Spotify AB
Birger Jarlsgatan 61, 4tr
Stockholm 113 56
Sweden

Email: erdtman@spotify.com

Anders Rundgren
Independent
Montpellier
France

Email: anders.rundgren.net@gmail.com

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: http://self-issued.info/