

JOSE Working Group	M. Jones
Internet-Draft	Microsoft
Intended status: Standards Track	March 12, 2012
Expires: September 13, 2012	

JSON Web Encryption JSON Serialization (JWE-JS) draft-jones-json-web-encryption-json-serialization- 01

Abstract

The JSON Web Encryption JSON Serialization (JWE-JS) is a means of representing encrypted content using JSON data structures. This specification describes a means of representing secured content as a JSON data object (as opposed to the JWE specification, which uses a compact serialization with a URL-safe representation). It enables the same content to be encrypted to multiple parties (unlike JWE). Cryptographic algorithms and identifiers used with this specification are enumerated in the separate JSON Web Algorithms (JWA) specification. The JSON Serialization for related digital signature and HMAC functionality is described in the separate JSON Web Signature JSON Serialization (JWS-JS) specification.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in **RFC 2119** [RFC2119].

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 <http://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 September 13, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

- 1. Introduction**
- 2. Terminology**

- [3. JSON Serialization](#)
- [4. Example JWE-JS](#)
- [5. IANA Considerations](#)
- [6. Security Considerations](#)
- [7. Open Issues and Things To Be Done \(TBD\)](#)
- [8. References](#)
 - [8.1. Normative References](#)
 - [8.2. Informative References](#)
- [Appendix A. Acknowledgements](#)
- [Appendix B. Document History](#)
- [§ Author's Address](#)

1. Introduction

TOC

The JSON Web Encryption JSON Serialization (JWE-JS) is a format for representing encrypted content as a JSON **RFC 4627** [RFC4627] object. It enables the same content to be encrypted to multiple parties (unlike JWE **[JWE]**.) The encryption mechanisms are independent of the type of content being encrypted. Cryptographic algorithms and identifiers used with this specification are enumerated in the separate JSON Web Algorithms (JWA) **[JWA]** specification. The JSON Serialization for related digital signature and HMAC functionality is described in the separate JSON Web Signature JSON Serialization (JWS-JS) **[JWS-JS]** specification.

2. Terminology

TOC

This specification uses the same terminology as the JSON Web Encryption (JWE) **[JWE]** specification.

3. JSON Serialization

TOC

The JSON Serialization represents encrypted content as a JSON object with members for each of four constituent parts: a `headers` member whose value is a non-empty array of Encoded JWE Header values, an `encrypted_keys` member whose value is a non-empty array of Encoded JWE Encrypted Key values, a `ciphertext` member whose value is an Encoded JWE Ciphertext value, and an `integrity_values` member whose value is a non-empty array of Encoded JWE Integrity Value values. The number of elements in each of the arrays **MUST** be the same.

Unlike the compact serialization used by JWEs, content using the JSON Serialization **MAY** be encrypted to more than one recipient. Each recipient requires:

- a JWE Header value specifying the cryptographic parameters used to encrypt the JWE Encrypted Key to that recipient and the parameters used to encrypt the plaintext to produce the JWE Ciphertext; these values are represented as Encoded JWE Header values that are elements of the non-empty array contained in the `headers` member.
- a JWE Encrypted Key value; these values are represented as Encoded JWE Encrypted Key values that are corresponding elements of the non-empty array contained in the `encrypted_keys` member.
- a JWE Integrity Value that ensures the integrity of the Ciphertext and the parameters used to create it; these values are represented as Encoded JWE Integrity Value values that are corresponding elements of the non-empty array contained in the `integrity_values` member.

Therefore, the syntax is:

```
{"headers":["<header 1 contents>","...", "<header N contents>"],
"encrypted_keys":["<key 1 contents>","...", "<key N contents>"],
"ciphertext":"<ciphertext contents>"}
```

```
"integrity_values":["<value 1 contents>","...", "<value N contents>"]
}
```

The contents of the Encoded JWE Header, Encoded JWE Encrypted Key, Encoded JWE Ciphertext, and Encoded JWE Integrity Value values are exactly as specified in JSON Web Encryption (JWE) [JWE]. They are interpreted and validated in the same manner, with each corresponding `headers`, `encrypted_keys`, and `integrity_values` value being created or validated together. The arrays MUST have the same number of elements.

The *i*'th JWE Encrypted Key value and the *i*'th JWE Integrity Value are computed using the parameters of *i*'th JWE Header value in the same manner described in the JWE specification. This has the desirable result that each Encoded JWE Encrypted Key value in the `encrypted_keys` array and each Encoded JWE Integrity Value in the `integrity_values` array are identical to the values that would have been computed for the same header and payload in a JWE, as is the JWE Ciphertext value.

All recipients use the same JWE Ciphertext value, resulting in potentially significant space savings if the message is large. Therefore, all header parameters that specify the treatment of the JWE Ciphertext value MUST be the same for all recipients. In particular, this means that the `enc` (encryption method) header parameter value in the JWE Header for each recipient MUST be the same, as MUST be the `iv` (initialization vector) value (when required for the algorithm).

4. Example JWE-JS

TOC

This section contains an example using the JWE JSON Serialization. This example demonstrates the capability for encrypting the same plaintext to multiple recipients.

Two recipients are present in this example: the first using the RSA-PKCS1_1.5 algorithm to produce the JWE Encrypted Key and the second using the ECDH-ES algorithm to produce the JWE Encrypted Key. The Plaintext is encrypted using the AES-256-CBC algorithm to produce the JWE Ciphertext. The two Decoded JWE Header Segments used are:

```
{ "alg": "RSA1_5",
  "enc": "A256CBC",
  "int": "HS256",
  "iv": "Mz-mW_4JHfg",
  "x5t": "7no0Pq-hJ1_hCnvWh6IeYI2w9Q0" }
```

and:

```
{ "alg": "ECDH-ES",
  "enc": "A256CBC",
  "epk": {
    "alg": "EC",
    "crv": "P-256",
    "x": "235RT7iKTI3KwvS4_mIwUhX60C_X2I-bs0pS5w7MGA4",
    "y": "ZSv0dtvXs4o2XsIVhFzgiMTSg9uSsyta0vC-XRtfoIM"},
  "jku": "https://example.com/public_key.jwk" }
```

The complete JSON Web Encryption JSON Serialization (JWE-JS) for these values is as follows (with line breaks for display purposes only):

```
{ "headers": [
  "eyJhbGciOiJSU0ExXzUiLA0KICJlbmMiOiJBMjU2Q0JDIiwNCiAiaW50Ijois",
  "FMNyNTYiLA0KICJpdii6Ik16LW1XXzRKSgZnIiwNCiAieDV0Ijoian25vT1BxLW",
  "hKMV9oQ252V2g2SWVZSTJ30VEwIn0",
  "eyJhbGciOiJFQ0RILUVTIiwNCiAiZW5jIjoiqTI1NkNCQyIsDQogImVwayI6e",
  "w0KICAIyWxnIjoirUMiLA0KICAIy3J2IjoiuC0yNTYiLA0KICAieCI6IjIzNVJUN",
  "Ij0" ] }
```

```
21LVEkzS1d2UzRfbU13VWhYNk9DX1gySS1ic09wUzV3N01HQTQiLA0KICAieSI6I
lpTdjBkdHZYczRvMlhzSVZoRnpnaU1UU2c5dVNzeXRhT3ZDLVhSdGZvSU0ifSwNC
iAiamt1IjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9wdWJsaWNfa2V5Lmp3ayJ9"],
  "encrypted_keys": [
    "TBD_encrypted_key_1_value_TBD",
    "TBD_encrypted_key_2_value_TBD"],
  "ciphertext": "TBD_ciphertext_value_TBD",
  "integrity_values": [
    "TBD_integrity_1_value_TBD",
    "TBD_integrity_2_value_TBD"]
}
```

TBD: Finish this example.

5. IANA Considerations

TOC

This specification makes no requests of IANA.

6. Security Considerations

TOC

The security considerations for this specification are the same as those for the JSON Web Encryption (JWE) [\[JWE\]](#) specification.

7. Open Issues and Things To Be Done (TBD)

TOC

The following items remain to be done in this draft:

- Complete the example.
- Track changes that occur in the JWE spec.

8. References

TOC

8.1. Normative References

TOC

- [JWA]** [Jones, M.](#), "[JSON Web Algorithms \(JWA\)](#)," March 2012.
- [JWE]** [Jones, M.](#), [Rescorla, E.](#), and [J. Hildebrand](#), "[JSON Web Encryption \(JWE\)](#)," March 2012.
- [RFC2119]** [Bradner, S.](#), "[Key words for use in RFCs to Indicate Requirement Levels](#)," BCP 14, RFC 2119, March 1997 ([TXT](#), [HTML](#), [XML](#)).
- [RFC4627]** Crockford, D., "[The application/json Media Type for JavaScript Object Notation \(JSON\)](#)," RFC 4627, July 2006 ([TXT](#)).

8.2. Informative References

TOC

- [I-D.rescorla-jsms]** Rescorla, E. and J. Hildebrand, "[JavaScript Message Security Format](#)," draft-rescorla-jsms-00 (work in progress), March 2011 ([TXT](#)).
- [JSE]** Bradley, J. and N. Sakimura (editor), "[JSON Simple Encryption](#)," September 2010.
- [JWS-JS]** [Jones, M.](#), [Bradley, J.](#), and [N. Sakimura](#), "[JSON Web Signature JSON Serialization \(JWS-JS\)](#)," March 2012.

TOC

Appendix A. Acknowledgements

TOC

JSON serializations for encrypted content were previously explored by **JSON Simple Encryption** [JSE] and **JavaScript Message Security Format** [I-D.rescorla-jsms].

Appendix B. Document History

TOC

-01

- Tracked changes between JOSE JWE draft -00 and -01, which added an integrity check for non-AEAD algorithms.

-00

- Created the initial version incorporating JOSE working group input and drawing from the JSON Serialization previously proposed in draft-jones-json-web-token-01.
-

Author's Address

TOC

Michael B. Jones
Microsoft
Email: mbj@microsoft.com
URI: <http://self-issued.info/>