JOSE Working Group	M. Jones
Internet-Draft	Microsoft
Intended status: Standards Track	October 15, 2012
Expires: April 18, 2013	

TOC

JSON Web Encryption JSON Serialization (JWE-JS) draft-jones-jose-jwe-json-serialization-02

Abstract

The JSON Web Encryption JSON Serialization (JWE-JS) is a means of representing encrypted content using JavaScript Object Notation (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 described in the separate JSON Web Algorithms (JWA) specification. The JSON Serialization for related digital signature and MAC functionality is described in the separate JSON Web Signature JSON Serialization (JWS-JS) specification.

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 April 18, 2013.

Copyright Notice

Table of Contents

---

1. Introduction

TOC

The JSON Web Encryption JSON Serialization (JWE-JS) is a format for representing encrypted content as a JavaScript Object Notation (JSON) **[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 described in the separate JSON Web Algorithms (JWA) **[JWA]** specification. The JSON Serialization for related digital signature and MAC functionality is described in the separate JSON Web Signature JSON Serialization (JWS-JS) **[JWS-JS]** specification.

1.1. Notational Conventions

TOC

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 Key words for use in RFCs to Indicate Requirement Levels **[RFC2119]**.

---

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 a `recipients` member containing an array of per-recipient information, an `initialization_vector` member containing a shared Encoded JWE Initialization Vector value, and a `ciphertext` member containing a shared Encoded JWE Ciphertext value. Each member of the `recipients` array is a JSON object with a `header` member containing an Encoded JWE Header value, an `encrypted_key` member containing an Encoded JWE Encrypted Key value, and an `integrity_value` member containing an Encoded JWE Integrity Value value.

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; this is represented as an Encoded JWE Header value in the `header` member of an object in the `recipients` array.
- a JWE Encrypted Key value used to encrypt the ciphertext; this is represented as an Encoded JWE Encrypted Key value in the `encrypted_key` member of the same object in the `recipients` array.
- a JWE Integrity Value that ensures the integrity of the Ciphertext and the parameters used to create it; this is represented as an Encoded JWE Integrity Value value in the `integrity_value` member of the same object in the `recipients` array.

Therefore, the syntax is:

```
{"recipients":[
  {"header":"<header 1 contents>",
   "encrypted_key":"<encrypted key 1 contents>",
   "integrity_value":"<integrity value 1 contents>"},
  ...
  {"header":"<header N contents>",
   "encrypted_key":"<encrypted key N contents>",
   "integrity_value":"<integrity value N contents>"}],
 "initialization_vector":"<initialization vector contents>",
 "ciphertext":"<ciphertext contents>"
}
```

The contents of the Encoded JWE Header, Encoded JWE Encrypted Key, Encoded JWE Initialization Vector, 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 `header`, `encrypted_key`, and `integrity_value` value being created and validated together.

Each JWE Encrypted Key value and the corresponding JWE Integrity Value are computed using the parameters of the corresponding 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 `recipients` array and each Encoded JWE Integrity Value in the same array element are identical to the values that would have been computed for the same parameters in a JWE, as is the shared JWE Ciphertext value.

All recipients use the same JWE Ciphertext and JWE Initialization Vector values, 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. This primarily means that the `enc` (encryption method) header parameter value in the JWE Header for each recipient MUST be the same.

TOC

4. Example JWE-JS

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 RSAES-PKCS1-V1_5 algorithm to encrypt the Content Master Key (CMK) and the second using RSAES OAEP to encrypt the CMK. The Plaintext is encrypted using the AES CBC algorithm and the same block encryption parameters to produce the common JWE Ciphertext value. The two Decoded JWE Header Segments used are:

```
{"alg":"RSA1_5","enc":"A128CBC+HS256"}
```

and:

```
{"alg":"RSA-OAEP","enc":"A128CBC+HS256"}
```

The keys used for the first recipient are the same as those in Appendix A.2 of **[JWE]**, as is the plaintext used. The asymmetric encryption key used for the second recipient is the same as that used in Appendix A.1 of **[JWE]**; the block encryption keys and parameters for the second recipient are the same as those for the first recipient (which must be the case, since the initialization vector and ciphertext are shared).

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

```
{"recipients":[
  {"header":
```

```
      "eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDK0hTMjU2In0",
     "encrypted_key":
      "O6AqXqgVlJJ4c4lp5sXZd7bpGHAw6ARkHUeXQxD1cAW4-X1x0qtj_AN0mukqE
       Ol4Y6UOwJXIJY9-G1ELK-RQWrKH_StR-AM9H7GpKmSEji8QYOcMOjr-u9H1Lt
       _pBEieG802SxWz0rbFTXRcj4BWLxcpCtjUZ31AP-sc-L_eCZ5UNl0aSRNqFsk
       uPkzRsFZRDJqSSJeVOyJ7pZCQ83fli19Vgi_3R7XMUqluQuuc7ZHOWixi47jX
       lBTlWRZ5iFxaS8G6J8wUrd4BKggAw3qX5XoIfXQVlQZE0Vmkq_zQSIo5LnFKy
       owooRcdsEuNh9B9Mkyt0ZQElG-jGdtHWjZSOA",
     "integrity_value":
      "RBGhYzE8_cZLHjJqqHuLhzbgWgL_wV3LDSUrcbkOiIA"},
    {"header":
      "eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkExMjhDQkMrSFMyNTYifQ",
     "encrypted_key":
      "myoFYZHErXG4gMVWl9UrFOCFIwvOUudYrxTsRsOt6maTc3W8G1FqGVOIBSZve
       BdZz2LqS42xta5OXEwLYaocObUxtfH9H8vMsjO-mBo7U9mp_PkS9PqVJMkeEe
       PLhzNLH0ecq7nYT6AFr5sSt4WMOPjSwHVQWtx43fZt4HvYaE_vgeSrxdi8KAb
       xbLzK_-qcYT6H7cwOMZrT6SFcXgLXESuKpF0azSGQtUmo0MLICP0YPBecGLTo
       PiveOH2awKZx0FkzPwi4JmOIvnAJ_wVQQJDVELwO9SIoF8olCQRHGyZ9rzDrr
       GRkoYgm2jVz-x0BuFVQFa4ZNufudtiT8pQxKg",
     "integrity_value":
      "i45dXWFjRKk805VtjIw_8iqGq1r9qPV7ULDLbnNAC_Q"}],
   "initialization_vector":
    "AxY8DCtDaGlsbGljb3RoZQ",
   "ciphertext":
    "1eBWFgcrz40wC88cgv8rPgu3EfmC1p4zT0kIxxfSF2zDJcQ-iEHk1jQM95xAdr5
     Z"
  }
```


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. References

TOC

7.1. Normative References

TOC

[JWA] Jones, M., "JSON Web Algorithms (JWA)," October 2012.

[JWE] Jones, M., Rescorla, E., and J. Hildebrand, "JSON Web Encryption (JWE)," October 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).

7.2. 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)," October 2012.

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. Open Issues

TOC

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

The following items remain to be considered or done in this draft:

- Track changes that occur in the JWE spec.

Appendix C. Document History

TOC

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

-02

- Changed to use an array of structures for per-recipient values, rather than a set of parallel arrays.
- Promoted Initialization Vector from being a header parameter to being a top-level JWE element. This saves approximately 16 bytes in the compact serialization, which is a significant savings for some use cases. Promoting the Initialization Vector out of the header also avoids repeating this shared value in the JSON serialization.

-01

- Added a complete JWE-JS example.
- Generalized language to refer to Message Authentication Codes (MACs) rather than Hash-based Message Authentication Codes (HMACs).

-00

- Renamed draft-jones-json-web-encryption-json-serialization to draft-jones-jose-jwe-json-serialization to have "jose" be in the document name so it can be included in the Related Documents list at http://datatracker.ietf.org/wg/jose/. No normative changes.

draft-jones-json-web-encryption-json-serialization-02

- Updated examples to track updated algorithm properties in the JWA spec.
- Tracked editorial changes made to the JWE spec.

draft-jones-json-web-encryption-json-serialization-01

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

draft-jones-json-web-encryption-json-serialization-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/