JOSE Working Group	M. Jones
Internet-Draft	Microsoft
Intended status: Standards Track	July 16, 2012
Expires: January 17, 2013	

TOC

JSON Web Encryption JSON Serialization (JWE-JS)
draft-jones-jose-jwe-json-serialization-01

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 January 17, 2013.

Copyright Notice

Table of Contents

1. **Introduction**
   - 1.1. **Notational Conventions**
2. **Terminology**
3. **JSON Serialization**
4. **Example JWE-JS**
5. **IANA Considerations**
6. **Security Considerations**
7. **Open Issues**
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 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 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, the `int` (integrity algorithm) value, and the `kdf` (key derivation function) value, when present.

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","int":"HS256","iv":"AxY8DCtDaGlsbGljb3RoZQ"}
```

and:

```
{"alg":"RSA-OAEP","enc":"A128CBC","int":"HS256","iv":"AxY8DCtDaGlsbGljb3RoZQ"}
```

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 ciphertext is shared).

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

```
{"headers":[
  "eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDIiwiaW50IjoiSFMyNTYiLCJp
   diI6IkF4WThEQ3REYUdsc2JHbGpiM1JvWlEifQ",
  "eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkExMjhDQkMiLCJpbnQiOiJIUzI1NiIs
   Iml2IjoiQXhZOERDdERhR2xzYkdsamIzUm9aUSJ9"],
 "encrypted_keys":[
  "IPI_z172hSWHMFgED8EG9DM6hIXU_6NaO1DImCn0vNeuoBq847Sl6qw_GHSYHJUQ
   XtXJq7S_CxWVrI82wjrOyaQca5tLZRZc45BfKHeqByThKI261QevEK56SyAwwXfK
   KZjSvkQ5dwTFSgfy76rMSUvVynHYEhdCatBF9HWTAiXPx7hgZixG1FeP_QCmOylz
   2VClVyYFCbjKREOwBFf-puNYfO75S3LNlJUtTsGGQL2oTKpMsEiUTdefkje91VX9
   h8g7908lFsggbjV7NicJsufuXxnTj1fcWIrRDeNIOmakiPEODi0gTSz0ou-W-LWK
```

```
    -3T1zYlOIiIKBjsExQKZ-w",
   "gyhQHQYGyPZQP21Oxd6TdJjrmNkals3Jin2631_eaW-8tPEZxjeNA1lJD7gi2tAQ
    X9ERZkbD8-9-8Gq9HkpJIhINX4TkqmCynmT8kQfjiv5t8KuTI_OjhD-I0CfvWs3T
    7yf2W6_vQcs1ezsapKPGj92i6Z1xpWgtDuK5YwU3PLAEfNeAgBK0fM9x9DkIuhBN
    7O9LWRtP6FnYKaygc5-tzR_O9nTJ0u4ImsBPGaHHQEfvzMBtQgAPQCdkDiITxSdJ
    OdvBQVmg7uGKOybF42R-gzP63lxAqiYmp6DVPlPYydIEB2lCZxGUJIsmEM0qwXt1
    GEmj_aYGhCUrkPSidvE6Ag"],
  "ciphertext":"_Z_djlIoC4MDSCKireWS2beti4Q6iSG2UjFujQvdz-_PQdUcFNkO
   ulegD6BgjgdFLjeB4HHOO7UHvP8PEDu0a0sA2a_-CI0w2YQQ2QQe35M",
  "integrity_values":[
   "c41k4T4eAgCCt63m8ZNmiOinMciFFypOFpvid7i6D0k",
   "NX62w-GLPhXVJuQxXnbWrBKLkt9j14IULcMdJ9kzeF0"]
 }
```


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

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.

8. References

TOC

8.1. Normative References

TOC

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

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

TOC

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

-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/