Abstract

Key Managed JSON Web Signature (KMJWS) represents content that is integrity protected with a Message Authentication Code (MAC) in which key management is employed for the MAC key. This representation reuses key management functionality already present in the JSON Web Encryption (JWE) specification and MAC functionality already present in the JSON Web Signature (JWS) 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 November 28, 2015.

Copyright Notice

Copyright (c) 2015 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

---

Key Managed JSON Web Signature (KMJWS)
draft-jones-jose-key-managed-json-web-signature-01

Abstract

Key Managed JSON Web Signature (KMJWS) represents content that is integrity protected with a Message Authentication Code (MAC) in which key management is employed for the MAC key. This representation reuses key management functionality already present in the JSON Web Encryption (JWE) specification and MAC functionality already present in the JSON Web Signature (JWS) 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 November 28, 2015.

Copyright Notice

Copyright (c) 2015 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 ................................................. 3
   1.1. Notational Conventions ................................. 3
2. Terminology .................................................. 3
3. Example KMJWS ................................................ 4
4. KMJWS Contents ................................................ 5
5. Header Parameters ............................................ 6
6. Serializations ............................................... 6
   6.1. JWS Compact Serialization ............................... 7
   6.2. JWS JSON Serialization ................................. 7
      6.2.1. General KMJWS JSON Serialization Syntax .......... 7
      6.2.2. Flattened KMJWS JSON Serialization Syntax ....... 9
7. Distinguishing between KMJWS, JWS, and JWE Objects ........... 9
8. IANA Considerations .......................................... 10
   8.1. JWS and JWE Header Parameter Registration .......... 10
      8.1.1. Registry Contents ................................ 10
9. Security Considerations ....................................... 10
10. References .................................................. 10
    10.1. Normative References ................................ 10
    10.2. Informative References ............................... 11
Appendix A. Example KMJWS using RSAES OAEPE and HMAC SHA-256 .... 11
   A.1. JOSE Header ............................................ 11
   A.2. Payload .................................................. 12
   A.3. JWS Signing Input ...................................... 12
   A.4. Integrity Protection ................................... 13
   A.5. Key Encryption ......................................... 13
   A.6. Complete Representation ................................ 15
Appendix B. Acknowledgements .................................... 15
Appendix C. Document History ................................... 15
Author’s Address ............................................... 16
1. Introduction

Key Managed JSON Web Signature (KMJWS) represents content that is integrity protected with a Message Authentication Code (MAC) in which key management is employed for the MAC key. This representation reuses key management functionality already present in the "JSON Web Encryption (JWE)" [JWE] specification and MAC functionality already present in the "JSON Web Signature (JWS)" [JWS] specification.

A KMJWS is neither a JWS nor a JWE, but incorporates elements of both. Specifically, the Key Management algorithms registered in the JSON Web Signature and Encryption Algorithms Registry [IANA.JOSE.Algs] are used to provide MAC keys in the same way that they are used to provide content encryption keys in "JSON Web Encryption (JWE)" [JWE]. Likewise, the MAC algorithms registered in this registry are used to integrity protect the JWS Payload and JWS Protected Header in the same way that they are used to integrity protect the JWS Payload and JWS Protected Header in "JSON Web Signature (JWS)" [JWS].

1.1. Notational Conventions

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 "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119]. The interpretation should only be applied when the terms appear in all capital letters.

UTF8(STRING) denotes the octets of the UTF-8 [RFC3629] representation of STRING, where STRING is a sequence of zero or more Unicode [UNICODE] characters.

ASCII(STRING) denotes the octets of the ASCII [RFC20] representation of STRING, where STRING is a sequence of zero or more ASCII characters.

The concatenation of two values A and B is denoted as A || B.

2. Terminology

This specification uses the same terminology as the "JSON Web Signature (JWS)" [JWS], "JSON Web Encryption (JWE)" [JWE], and "JSON Web Algorithms (JWA)" [JWA] specifications.

These terms are defined by this specification:
3. Example KMJWS

This section provides an example of a KMJWS. Its computation is described in more detail in Appendix A, including specifying the key values used.

The following example JWS Protected Header declares that:

- The MAC Key is encrypted using the RSAES OAEP [RFC3447] algorithm to produce the KMJWS Encrypted Key.
- The JWS Protected Header and the JWS Payload are integrity protected using the HMAC SHA-256 [RFC2104] [SHS] algorithm.

{"alg":"RSA-OAEP","mac":"HS256"}

Encoding this JWS Protected Header as BASE64URL(UTF8(JWS Protected Header)) gives this value:

eyJhbGciOiJSU0EtT0FFUCIsIm1hYyI6IkhTMjU2In0

The payload in this example is the ASCII representation of the text "What I have written, I have written." The value BASE64URL(JWS Payload) is:

V2hhdCBJIGhhdmUgd3JpdHRlbiwgSSBoYXZlIHdyaXR0ZW4u

Computing the HMAC of the JWS Signing Input ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload)) with the HMAC SHA-256 algorithm using the MAC Key specified in Appendix A.4 and base64url-encoding the result yields this BASE64URL(JWS Signature) value:
The MAC Key is encrypted using the RSAES OAEP algorithm and the RSA key specified in Appendix A.5. The resulting BASE64URL(KMJWS Encrypted Key) value (with line breaks for display purposes only) is:

```
OKOawDo13gRp2ojaHV7LFp2czgV7T6DVZKTYkOMTYUmKoTCVJRgckCL9kiMT03JGep
ipsEdY3mx_etLbbW5rFr05kLzcSr4gKAg7YN7e9jwQRb23nfa6c9d-StnImGyFDb
Sv04vVuxIp5Zmsl9NnXK2K2da14BBs4rzVltdYwam_lDp5Xn2AYpQd576FdIKLaV
mqgfwX7XRxv2322i-vDxRfqNzo_tETKzpVLzfiwQyeyPGLBIO56YJ7eObdvVje8
1860pamavo35UgoRdbYAcboh9QcfylQr66oc6vFWXRc2_2T2LawVCWITy3brGPi
6Uk1fCpmfIfj7iGdXXHzg
```

Concatenating these values in the order 
Header.Payload.Signature.Encrypted_Key with period (‘.’) characters 
between the parts yields this complete KMJWS representation using the 
KMJWS Compact Serialization (with line breaks for display purposes 
only):

```
eyJhbGciOiJSU0EtT0FFUCIsIm1hYyI6IkhTMjU2Iiw6
V2hhdCBJIGhhdmUgd3JpdHRlbiwgSSBoYXZlIHdyAXR02W4u
NjTOnXAAxtr7dA6RSxYkZcD6F-n5BOrlHRTxiTLptKM
OKOawDo13gRp2ojaHV7LFpZczgV7T6DVZKTYkOMTYUmKoTCVJRgckCL9kiMT03JGep
ipsEdY3mx_etLbbW5rFr05kLzcSr4gKAg7YN7e9jwQRb23nfa6c9d-StnImGyFDb
Sv04vVuxIp5Zmsl9NnXK2K2da14BBs4rzVltdYwam_lDp5Xn2AYpQd576FdIKLaV
mqgfwX7XRxv2322i-vDxRfqNzo_tETKzpVLzfiwQyeyPGLBIO56YJ7eObdvVje8
1860pamavo35UgoRdbYAcboh9QcfylQr66oc6vFWXRc2_2T2LawVCWITy3brGPi
6Uk1fCpmfIfj7iGdXXHzg
```

4. KMJWS Contents

A Key Managed JSON Web Signature (KMJWS) represents these logical values:

- JOSE Header
- JWS Payload
- JWS Signature
- KMJWS Encrypted Key

For a KMJWS, just as it is for a JWS, the JOSE Header members are the union of the members of these values:
The JWS Payload and JWS Signature are likewise the same for a KMJWS as they are for a JWS.

The KMJWS Encrypted Key is the one value present in a KMJWS that is not present in a JWS. It enables key management for the MAC Key.

5. Header Parameters

A KMJWS uses these Header Parameter fields to convey the key management and MAC algorithms used:

- **alg**
  - The key management algorithm employed. This parameter has the same meaning, syntax, and processing rules as the "alg" Header Parameter defined in Section 4.1.1 of [JWE], except that the key being encrypted or determined is the MAC key, rather than the Content Encryption Key.

- **mac**
  - This parameter has the same meaning, syntax, and processing rules as the "alg" Header Parameter defined in Section 4.1.1 of [JWS], except that the algorithm MUST be a MAC algorithm and the MAC key is determined by the key management procedure employed.

The "dir" (direct) key management algorithm defined in Section 4.5 of [JWA] MUST NOT be used as the "alg" Header Parameter value, as doing so would unnecessarily create a second equivalent representation for content integrity-protected with a MAC with no key management. The normal JWS representation for this MUST be used instead.

The "jku", "jwk", "kid", "x5u", "x5c", "x5t", "x5t#S256", "typ", and "crit" Header Parameters defined in Section 4.1 of [JWE] are used identically, except that the key being encrypted or determined is the MAC key, rather than the Content Encryption Key. The "cty" Header Parameter defined in Section 4.1.10 of [JWS] is used identically, except that the JWS Payload is that of a KMJWS, rather than a JWS. The "enc" and "zip" Header Parameters defined in Section 4.1 of [JWE] MUST NOT be used.

6. Serializations

Like JWSs, KMJWSs can utilize one of two different serializations: the KMJWS Compact Serialization or the KMJWS JSON Serialization.
6.1. JWS Compact Serialization

Like the JWS Compact Serialization, the KMJWS Compact Serialization represents MACed content as a compact, URL-safe string. This string is:

```
BASE64URL(UTF8(JWS Protected Header)) || '.' ||
BASE64URL(JWS Payload) || '.' ||
BASE64URL(JWS Signature) || '.' ||
BASE64URL(KMJWS Encrypted Key)
```

Only one MAC is supported by the KMJWS Compact Serialization and it provides no syntax to represent a JWS Unprotected Header value.

6.2. JWS JSON Serialization

Like the JWS JSON Serialization, the KMJWS JSON Serialization represents MACed content as a JSON object. This representation is neither optimized for compactness nor URL-safe.

Also like the JWS JSON Serialization, two closely related syntaxes are defined for the KMJWS JSON Serialization: a fully general syntax, with which content can be secured with more than one MAC operation, and a flattened syntax, which is optimized for the single MAC case.

6.2.1. General KMJWS JSON Serialization Syntax

The following members are defined for use in top-level JSON objects used for the fully general KMJWS JSON Serialization syntax:

- **payload**: This is the same as the JWS "payload" member.
- **signatures**: This is the same as the JWS "signatures" member.

The following members are defined for use in the JSON objects that are elements of the "signatures" array:

- **protected**: This is the same as the JWS "protected" member.
- **header**: This is the same as the JWS "header" member.
signature

This is the same as the JWS "signature" member.

encrypted_key

The "encrypted_key" member MUST be present and contain the value BASE64URL(KMJWS Encrypted Key).

At least one of the "protected" and "header" members MUST be present for each MAC computation so that "alg" and "mac" Header Parameter values are conveyed.

Additional members can be present in both the JSON objects defined above; if not understood by implementations encountering them, they MUST be ignored.

The Header Parameter values used when creating or validating individual MAC values are the union of the two sets of Header Parameter values that may be present: (1) the JWS Protected Header represented in the "protected" member of the MAC’s array element, and (2) the JWS Unprotected Header in the "header" member of the MAC’s array element. The union of these sets of Header Parameters comprises the JOSE Header. The Header Parameter names in the two locations MUST be disjoint.

Each JWS Signature value is computed using the parameters of the corresponding JOSE Header value in the same manner as for the JWS Compact Serialization. This has the desirable property that each JWS Signature value represented in the "signatures" array is identical to the value that would have been computed for the same parameter in the KMJWS Compact Serialization, provided that the JWS Protected Header value for that MAC computation (which represents the integrity protected Header Parameter values) matches that used in the KMJWS Compact Serialization.
In summary, the syntax of a KMJWS using the general KMJWS JSON Serialization is as follows:

```
{
   "payload":"<payload contents>",
   "signatures":[
       "protected":"<integrity-protected header 1 contents>",
       "header":<non-integrity-protected header 1 contents>,
       "signature":"<signature 1 contents>",
       "encrypted_key":"<encrypted key 1 contents>",
       ...
       "protected":"<integrity-protected header N contents>",
       "header":<non-integrity-protected header N contents>,
       "signature":"<signature N contents>",
       "encrypted_key":"<encrypted key N contents>"
   ]
}
```

### 6.2.2. Flattened KMJWS JSON Serialization Syntax

The flattened KMJWS JSON Serialization syntax is based upon the general syntax, but flattens it in the same way that the flattened JWS JSON Serialization syntax flattens its general syntax.

In summary, the syntax of a KMJWS using the flattened KMJWS JSON Serialization is as follows:

```
{
   "payload":"<payload contents>",
   "protected":"<integrity-protected header contents>",
   "header":<non-integrity-protected header contents>,
   "signature":"<signature contents>",
   "encrypted_key":"<encrypted key contents>"
}
```

### 7. Distinguishing between KMJWS, JWS, and JWE Objects

While KMJWSs have characteristics of both JWSs and JWEs, these methods can be used to distinguish KMJWSs from either of them. This section augments the information in Section 9 of [JWE].

- If the object is using a compact serialization, the number of base64url-encoded segments separated by period (‘.’) characters will differ. KMJWSs have four segments separated by three period (‘.’) characters and the others do not.

- If the object is using a JSON serialization, the members used will be different. KMJWSs have both a "payload" and an "encrypted_key"
member and the others do not.

- The JOSE Header for a KMJWS can also be distinguished from the JOSE Header for a JWS or JWE by determining whether a "mac" (MAC algorithm) member exists. If the "mac" member exists, it is a KMJWS; otherwise, it is not.

8. IANA Considerations

8.1. JWS and JWE Header Parameter Registration

This specification registers the "mac" (MAC algorithm) Header Parameter defined in Section 5 in the IANA JSON Web Signature and Encryption Header Parameters registry defined in [JWS].

8.1.1. Registry Contents

- Header Parameter Name: "mac"
- Header Parameter Description: MAC Algorithm
- Header Parameter Usage Location(s): KMJWS
- Change Controller: IETF
- Specification Document(s): Section 5 of [[ this document ]]

9. Security Considerations

The key management security considerations from [JWE] apply. The integrity protection security considerations from [JWS] apply. The algorithm security considerations from [JWA] apply.

10. References

10.1. Normative References

[IANA.JOSE.Algs]


Appendix A. Example KMJWS using RSAES OAEP and HMAC SHA-256

This example secures the payload using RSAES OAEP for key encryption and HMAC SHA-256 for integrity protection.

A.1. JOSE Header

The following example JWS Protected Header declares that:
The MAC Key is encrypted using the RSAES OAEP [RFC3447] algorithm to produce the KMJWS Encrypted Key.

The JWS Protected Header and the JWS Payload are integrity protected using the HMAC SHA-256 [RFC2104] [SHS] algorithm.

{"alg":"RSA-OAEP","mac":"HS256"}

Encoding this JWS Protected Header as BASE64URL(UTF8(JWS Protected Header)) gives this value:

eyJhbGciOiJSU0EtT0FFUCIsIm1hYyI6IkhTMjU2In0

A.2. Payload

The payload in this example is the ASCII representation of the text "What I have written, I have written." The representation of this payload (using JSON array notation) is:


The value BASE64URL(JWS Payload) is:

V2hhdCBJIGhhdmUgd3JpdHRlbiwgSSBoYXZlIHdyaXR0ZW4u

A.3. JWS Signing Input

Combining these as BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload) gives this string (with line breaks for display purposes only):

eyJhbGciOiJSU0EtT0FFUCIsIm1hYyI6IkhTMjU2In0

V2hhdCBJIGhhdmUgd3JpdHRlbiwgSSBoYXZlIHdyaXR0ZW4u

The resulting JWS Signing Input value, which is the ASCII representation of above string, is the following octet sequence (using JSON array notation):

A.4. Integrity Protection

Compute the HMAC of the JWS Signing Input with the HMAC SHA-256 algorithm. This example uses the MAC Key below:

\[ [177, 161, 244, 84, 143, 225, 115, 63, 180, 3, 255, 107, 154, 212, 246, 138, 7, 110, 91, 112, 46, 34, 105, 47, 130, 203, 46, 122, 234, 64, 252] \]

The resulting JWS Signature value is:


Base64url-encoding the result yields this BASE64URL(JWS Signature) value:

NjTOnXAAxtr7dA6RSxYkZcD6F-n5BOrLHRTxiTLptKM

A.5. Key Encryption

Encrypt the MAC Key with the recipient's public key using the RSAES OAEP algorithm to produce the KMJWS Encrypted Key. This example uses the RSA key represented in JWK [JWK] format below (with line breaks within values for display purposes only):
The resulting KMJWS Encrypted Key value is:

```
```

Jones                   Expires November 28, 2015              [Page 14]
Encoding this KMJWS Encrypted Key as BASE64URL(KMJWS Encrypted Key) gives this value (with line breaks for display purposes only):

OKOawDo13gRp2ojaHV7LFpZcgV776DVZKTyKOMTYUmKoTCVJRgckCL9kiMT03JGeipsEdY3mx_etLbbWSrFr05klzczSr4qKAq7YN7e9jwQRb23nfa6c9d-StnImGyFDbSv04uVuxIp52ms1gNxKMKK2Da14B8B4rzVRltdYwam_1DPx5A2YpQdb76FdIKLaVmggfwX7XRrvx232i-vDxRfQnzo_tETKzpVLzfiwQyeyPGLBIO56YJ7eObdv0jje81860ppamavo35UgaDyAbc9hOqcfylQr66oc6vFXRc2_2T2LawVCWITy3brGPi6UklfCpIMfIj7iGdXKHhzg

A.6. Complete Representation

Assemble the final representation: The Compact Serialization of this result is the string BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload) || '.' || BASE64URL(JWS Signature) || '.' || BASE64URL(KMJWS Encrypted Key).

The final result in this example (with line breaks for display purposes only) is:

eyJhbGciOiJSU0EtT0FFUCIsIm1hYyI6IkhTMjU2In0.
V2hhdCBJIGhhdmUgd3JpdHRlbiwgSSBoYXZlIHdyXR0ZW4u.
NjTOnXAAx7r7dA6RSxYkZcD6F-n5BOrLHRTxiTLptKM.
OKOawDo13gRp2ojaHV7LFpZcgV776DVZKTyKOMTYUmKoTCVJRgckCL9kiMT03JGeipsEdY3mx_etLbbWSrFr05klzczSr4qKAq7YN7e9jwQRb23nfa6c9d-StnImGyFDbSv04uVuxIp52ms1gNxKMKK2Da14B8B4rzVRltdYwam_1DPx5A2YpQdb76FdIKLaVmggfwX7XRrvx232i-vDxRfQnzo_tETKzpVLzfiwQyeyPGLBIO56YJ7eObdv0jje81860ppamavo35UgaDyAbc9hOqcfylQr66oc6vFXRc2_2T2LawVCWITy3brGPi6UklfCpIMfIj7iGdXKHhzg

Appendix B. Acknowledgements

Richard Barnes and Jim Schaad contributed to discussions on the possibility of introducing key management as an option for JWSs using MACs.

Appendix C. Document History

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

Jones Expires November 28, 2015
Prohibited using "dir" as the "alg" Header Parameter value.

Created draft-jones-jose-key-managed-json-web-signature.

Author’s Address

Michael B. Jones
Microsoft

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