JSON Web Key (JWK)
draft-ietf-jose-json-web-key-18

Abstract

A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) data structure that represents a cryptographic key. This specification also defines a JSON Web Key Set (JWK Set) JSON data structure for representing a set of JWKs. Cryptographic algorithms and identifiers for use with this specification are described in the separate JSON Web Algorithms (JWA) specification and IANA registries defined by that 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 May 16, 2014.

Copyright Notice

Copyright (c) 2013 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 ................................................. 4
   1.1. Notational Conventions .............................. 4
2. Terminology .................................................. 4
3. JSON Web Key (JWK) Format .................................. 5
   3.1. "kty" (Key Type) Parameter .......................... 5
   3.2. "use" (Key Use) Parameter ........................... 6
   3.3. "alg" (Algorithm) Parameter .......................... 6
   3.4. "kid" (Key ID) Parameter ............................. 6
   3.5. "x5u" (X.509 URL) Parameter .......................... 7
   3.6. "x5t" (X.509 Certificate SHA-1 Thumbprint) Parameter .. 7
   3.7. "x5c" (X.509 Certificate Chain) Parameter ........... 8
4. JSON Web Key Set (JWK Set) Format .......................... 8
   4.1. "keys" Parameter ..................................... 9
5. String Comparison Rules ...................................... 9
6. Encrypted JWK and Encrypted JWK Set Formats ................. 9
7. IANA Considerations ......................................... 9
   7.1. JSON Web Key Parameters Registry ..................... 10
      7.1.1. Registration Template ........................... 11
      7.1.2. Initial Registry Contents ....................... 11
   7.2. JSON Web Key Use Registry ............................ 13
      7.2.1. Registration Template ........................... 13
      7.2.2. Initial Registry Contents ....................... 13
   7.3. JSON Web Key Set Parameters Registry ................. 14
      7.3.1. Registration Template ........................... 14
      7.3.2. Initial Registry Contents ....................... 14
   7.4. Media Type Registration ................................ 14
      7.4.1. Registry Contents ................................ 14
8. Security Considerations ..................................... 15
9. References .................................................. 16
   9.1. Normative References .................................. 16
   9.2. Informative References ............................... 18
Appendix A. Example JSON Web Key Sets ......................... 18
   A.1. Example Public Keys .................................. 18
   A.2. Example Private Keys .................................. 19
   A.3. Example Symmetric Keys ............................... 21
Appendix B. Example Use of "x5c" (X.509 Certificate Chain) Parameter ........................................... 21
Appendix C. Example Encrypted RSA Private Key .................. 22
   C.1. Plaintext RSA Private Key .............................. 23
   C.2. JWE Header ............................................ 26
   C.3. Content Encryption Key (CEK) .......................... 26
   C.4. Key Encryption ....................................... 27
1. Introduction

A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) [RFC4627] data structure that represents a cryptographic key. This specification also defines a JSON Web Key Set (JWK Set) JSON data structure for representing a set of JWKs. Cryptographic algorithms and identifiers for use with this specification are described in the separate JSON Web Algorithms (JWA) [JWA] specification and IANA registries defined by that specification.

Goals for this specification do not include representing certificate chains, representing certified keys, and replacing X.509 certificates.

JWKs and JWK Sets are used in the JSON Web Signature (JWS) [JWS] and JSON Web Encryption (JWE) [JWE] specifications.

Names defined by this specification are short because a core goal is for the resulting representations to be compact.

1.1. Notational Conventions

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]. If these words are used without being spelled in uppercase then they are to be interpreted with their normal natural language meanings.

BASE64URL(OCTETS) denotes the base64url encoding of OCTETS, per Section 2.

UTF8(STRING) denotes the octets of the UTF-8 [RFC3629] representation of STRING.

ASCII(STRING) denotes the octets of the ASCII [USASCII] representation of STRING.

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

2. Terminology

These terms defined by the JSON Web Signature (JWS) [JWS] specification are incorporated into this specification: "Base64url Encoding" and "Collision-Resistant Name".

These terms are defined for use by this specification:
JSON Web Key (JWK)  A JSON object that represents a cryptographic key.

JSON Web Key Set (JWK Set)  A JSON object that contains an array of JWKs as the value of its "keys" member.

## 3. JSON Web Key (JWK) Format

A JSON Web Key (JWK) is a JSON object. The members of the object represent properties of the key, including its value. This document defines the key parameters that are not algorithm specific, and thus common to many keys.

In addition to the common parameters, each JWK will have members that are specific to the kind of key being represented. These members represent the parameters of the key. Section 6 of the JSON Web Algorithms (JWA) [JWA] specification defines multiple kinds of cryptographic keys and their associated members.

The member names within a JWK MUST be unique; recipients MUST either reject JWKs with duplicate member names or use a JSON parser that returns only the lexically last duplicate member name, as specified in Section 15.12 (The JSON Object) of ECMAScript 5.1 [ECMAScript].

Additional members can be present in the JWK. If not understood by implementations encountering them, they MUST be ignored. Member names used for representing key parameters for different keys types need not be distinct. Any new member name should either be registered in the IANA JSON Web Key Parameters registry defined in Section 7.1 or be a value that contains a Collision-Resistant Name.

### 3.1. "kty" (Key Type) Parameter

The "kty" (key type) member identifies the cryptographic algorithm family used with the key. "kty" values should either be registered in the IANA JSON Web Key Types registry defined in [JWA] or be a value that contains a Collision-Resistant Name. The "kty" value is a case-sensitive string. This member MUST be present in a JWK.

A list of defined "kty" values can be found in the IANA JSON Web Key Types registry defined in [JWA]; the initial contents of this registry are the values defined in Section 6.1 of the JSON Web Algorithms (JWA) [JWA] specification.

The key type definitions include specification of the members to be used for those key types. Additional members used with "kty" values can also be found in the IANA JSON Web Key Parameters registry.
defined in Section 7.1.

3.2. "use" (Key Use) Parameter

The "use" (key use) member identifies the intended use of the key. Values defined by this specification are:

- "sig" (signature or MAC)
- "enc" (encryption)

Other values MAY be used. Key Use values can be registered in the IANA JSON Web Key Use registry defined in Section 7.2. The "use" value is a case-sensitive string. A "use" member SHOULD be present, unless the application uses another means or convention to determine the intended key usage.

When a key is used to wrap another key and a key use designation for the first key is desired, the "enc" (encryption) key use value SHOULD be used, since key wrapping is a kind of encryption. (The "alg" member can be used to specify the particular kind of encryption to be performed, when desired.)

3.3. "alg" (Algorithm) Parameter

The "alg" (algorithm) member identifies the algorithm intended for use with the key. The values used should either be registered in the IANA JSON Web Signature and Encryption Algorithms registry defined in [JWA] or be a value that contains a Collision-Resistant Name. Use of this member is OPTIONAL.

3.4. "kid" (Key ID) Parameter

The "kid" (key ID) member can be used to match a specific key. This can be used, for instance, to choose among a set of keys within a JWK Set during key rollover. The structure of the "kid" value is unspecified. When "kid" values are used within a JWK Set, different keys within the JWK Set SHOULD use distinct "kid" values. (One example in which different keys might use the same "kid" value is if they have different "kty" (key type) values but are considered to be equivalent alternatives by the application using them.) The "kid" value is a case-sensitive string. Use of this member is OPTIONAL.

When used with JWS or JWE, the "kid" value is used to match a JWS or JWE "kid" Header Parameter value.
3.5. "x5u" (X.509 URL) Parameter

The "x5u" (X.509 URL) member is a URI [RFC3986] that refers to a resource for an X.509 public key certificate or certificate chain [RFC5280]. The identified resource MUST provide a representation of the certificate or certificate chain that conforms to RFC 5280 [RFC5280] in PEM encoded form [RFC1421]. The key in the first certificate MUST match the public key represented by other members of the JWK. The protocol used to acquire the resource MUST provide integrity protection; an HTTP GET request to retrieve the certificate MUST use TLS [RFC2818] [RFC5246]; the identity of the server MUST be validated, as per Section 3.1 of HTTP Over TLS [RFC2818]. Use of this member is OPTIONAL.

While there is no requirement that members other than those representing the public key be populated when an "x5u" member is present, doing so may improve interoperability for applications that do not handle PKIX certificates. If other members are present, the contents of those members MUST be semantically consistent with the related fields in the first certificate. For instance, if the "use" member is present, then it needs to allow for only a subset of the usages that are permitted by the certificate. Similarly, if the "alg" member is present, it should represent an algorithm that the certificate allows.

3.6. "x5t" (X.509 Certificate SHA-1 Thumbprint) Parameter

The "x5t" (X.509 Certificate SHA-1 Thumbprint) member is a base64url encoded SHA-1 thumbprint (a.k.a. digest) of the DER encoding of an X.509 certificate [RFC5280]. The key in the certificate MUST match the public key represented by other members of the JWK. Use of this member is OPTIONAL.

If, in the future, certificate thumbprints need to be computed using hash functions other than SHA-1, it is suggested that additional related JWK parameters be defined for that purpose. For example, it is suggested that a new "x5t#S256" (X.509 Certificate Thumbprint using SHA-256) JWK parameter could be defined by registering it in the IANA JSON Web Key Parameters registry defined in Section 7.1.

As with the "x5u" member, members other than those representing the public key may also be populated when an "x5t" member is present. If other members are present, the contents of those members MUST be semantically consistent with the related fields in the referenced certificate. See the last paragraph of Section 3.5 for additional guidance on this.
3.7. "x5c" (X.509 Certificate Chain) Parameter

The "x5c" (X.509 Certificate Chain) member contains a chain of one or more PKIX certificates [RFC5280]. The certificate chain is represented as a JSON array of certificate value strings. Each string in the array is a base64 encoded ([RFC4648] Section 4 -- not base64url encoded) DER [ITU.X690.1994] PKIX certificate value. The PKIX certificate containing the key value MUST be the first certificate. This MAY be followed by additional certificates, with each subsequent certificate being the one used to certify the previous one. The key in the first certificate MUST match the public key represented by other members of the JWK. Use of this member is OPTIONAL.

As with the "x5u" member, members other than those representing the public key may also be populated when an "x5c" member is present. If other members are present, the contents of those members MUST be semantically consistent with the related fields in the first certificate. See the last paragraph of Section 3.5 for additional guidance on this.

4. JSON Web Key Set (JWK Set) Format

A JSON Web Key Set (JWK Set) is a JSON object representing a set of JWKs. The JSON object MUST have a "keys" member, which is an array of JWK objects.

The member names within a JWK Set MUST be unique; recipients MUST either reject JWK Sets with duplicate member names or use a JSON parser that returns only the lexically last duplicate member name, as specified in Section 15.12 (The JSON Object) of ECMAScript 5.1 [ECMAScript].

Additional members can be present in the JWK Set. If not understood by implementations encountering them, they MUST be ignored. Parameters for representing additional properties of JWK Sets should either be registered in the IANA JSON Web Key Set Parameters registry defined in Section 7.3 or be a value that contains a Collision-Resistant Name.

Implementations SHOULD ignore JWKs within a JWK Set that use "kty" (key type) values that are not understood by them, are missing required members, or for which values are out of the supported ranges.
4.1. "keys" Parameter

The value of the "keys" member is an array of JWK values. By default, the order of the JWK values within the array does not imply an order of preference among them, although applications of JWK Sets can choose to assign a meaning to the order for their purposes, if desired. This member MUST be present in a JWK Set.

5. String Comparison Rules

The string comparison rules for this specification are the same as those defined in Section 5.3 of [JWS].

6. Encrypted JWK and Encrypted JWK Set Formats

JWKs containing non-public key material will need to be encrypted in some contexts to prevent the disclosure of private or symmetric key values to unintended parties. The use of an Encrypted JWK, which is a JWE with the UTF-8 encoding of a JWK as its plaintext value, is recommended for this purpose. The processing of Encrypted JWKs is identical to the processing of other JWEs. A "cty" (content type) Header Parameter value of "jwk+json" MUST be used to indicate that the content of the JWE is a JWK, unless the application knows that the encrypted content is a JWK by another means or convention.

JWK Sets containing non-public key material will similarly need to be encrypted. The use of an Encrypted JWK Set, which is a JWE with the UTF-8 encoding of a JWK Set as its plaintext value, is recommended for this purpose. The processing of Encrypted JWK Sets is identical to the processing of other JWEs. A "cty" (content type) Header Parameter value of "jwk-set+json" MUST be used to indicate that the content of the JWE is a JWK Set, unless the application knows that the encrypted content is a JWK Set by another means or convention.

See Appendix C for an example encrypted JWK.

7. IANA Considerations

The following registration procedure is used for all the registries established by this specification.

Values are registered with a Specification Required [RFC5226] after a two-week review period on the [TBD]@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of values prior to publication, the Designated Expert(s)
may approve registration once they are satisfied that such a specification will be published.

Registration requests must be sent to the [TBD]@ietf.org mailing list for review and comment, with an appropriate subject (e.g., "Request for access token type: example"). [[ Note to the RFC Editor: The name of the mailing list should be determined in consultation with the IESG and IANA. Suggested name: jose-reg-review. ]]

Within the review period, the Designated Expert(s) will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful. Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention (using the iesg@iesg.org mailing list) for resolution.

Criteria that should be applied by the Designated Expert(s) includes determining whether the proposed registration duplicates existing functionality, determining whether it is likely to be of general applicability or whether it is useful only for a single application, and whether the registration makes sense.

IANA must only accept registry updates from the Designated Expert(s) and should direct all requests for registration to the review mailing list.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly-informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Expert(s).

### 7.1. JSON Web Key Parameters Registry

This specification establishes the IANA JSON Web Key Parameters registry for JWK parameter names. The registry records the parameter name, the key type(s) that the parameter is used with, and a reference to the specification that defines it. It also records whether the parameter conveys public or private information. This specification registers the parameter names defined in Section 3. The same JWK parameter name may be registered multiple times, provided that duplicate parameter registrations are only for key type specific JWK parameters; in this case, the meaning of the duplicate parameter name is disambiguated by the "kty" value of the JWK containing it.
7.1.1. Registration Template

Parameter Name:
The name requested (e.g., "example"). Because a core goal of this specification is for the resulting representations to be compact, it is RECOMMENDED that the name be short -- not to exceed 8 characters without a compelling reason to do so. This name is case-sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Expert(s) state that there is a compelling reason to allow an exception in this particular case. However, matching names may be registered, provided that the accompanying sets of "kty" values that the Parameter Name is used with are disjoint; for the purposes of matching "kty" values, "*" matches all values.

Parameter Description:
Brief description of the parameter (e.g., "Example description").

Used with "kty" Value(s):
The key type parameter value(s) that the parameter name is to be used with, or the value "*" if the parameter value is used with all key types. Values may not match other registered "kty" values in a case-sensitive manner when the registered Parameter Name is the same (including when the Parameter Name matches in a case-insensitive manner) unless the Designated Expert(s) state that there is a compelling reason to allow an exception in this particular case.

Parameter Information Class:
Registers whether the parameter conveys public or private information. Its value must be one of the words Public or Private.

Change Controller:
For Standards Track RFCs, state "IESG". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):
Reference to the document(s) that specify the parameter, preferably including URI(s) that can be used to retrieve copies of the document(s). An indication of the relevant sections may also be included but is not required.

7.1.2. Initial Registry Contents

  o Parameter Name: "kty"
Parameter Description: Key Type
Used with "kty" Value(s): *
Parameter Information Class: Public
Change Controller: IESG
Specification Document(s): Section 3.1 of [[ this document ]]

Parameter Name: "use"
Parameter Description: Key Use
Used with "kty" Value(s): *
Parameter Information Class: Public
Change Controller: IESG
Specification Document(s): Section 3.2 of [[ this document ]]

Parameter Name: "alg"
Parameter Description: Algorithm
Used with "kty" Value(s): *
Parameter Information Class: Public
Change Controller: IESG
Specification Document(s): Section 3.3 of [[ this document ]]

Parameter Name: "kid"
Parameter Description: Key ID
Used with "kty" Value(s): *
Parameter Information Class: Public
Change Controller: IESG
Specification Document(s): Section 3.4 of [[ this document ]]

Parameter Name: "x5u"
Parameter Description: X.509 URL
Used with "kty" Value(s): *
Parameter Information Class: Public
Change Controller: IESG
Specification Document(s): Section 3.5 of [[ this document ]]

Parameter Name: "x5t"
Parameter Description: X.509 Certificate SHA-1 Thumbprint
Used with "kty" Value(s): *
Parameter Information Class: Public
Change Controller: IESG
Specification Document(s): Section 3.6 of [[ this document ]]

Parameter Name: "x5c"
Parameter Description: X.509 Certificate Chain
Used with "kty" Value(s): *
Parameter Information Class: Public
Change Controller: IESG
This specification establishes the IANA JSON Web Key Use registry for JWK "use" member values. The registry records the key use value and a reference to the specification that defines it. This specification registers the parameter names defined in Section 3.2.

7.2.1. Registration Template

Use Member Value:
The name requested (e.g., "example"). Because a core goal of this specification is for the resulting representations to be compact, it is RECOMMENDED that the name be short -- not to exceed 8 characters without a compelling reason to do so. This name is case-sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Expert(s) state that there is a compelling reason to allow an exception in this particular case.

Use Description:
Brief description of the use (e.g., "Example description").

Change Controller:
For Standards Track RFCs, state "IESG". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):
Reference to the document(s) that specify the parameter, preferably including URI(s) that can be used to retrieve copies of the document(s). An indication of the relevant sections may also be included but is not required.

7.2.2. Initial Registry Contents

- Use Member Value: "sig"
  - Use Description: Signature or MAC
  - Change Controller: IESG
  - Specification Document(s): Section 3.2 of [[ this document ]]

- Use Member Value: "enc"
  - Use Description: Encryption
  - Change Controller: IESG
  - Specification Document(s): Section 3.2 of [[ this document ]]

- Use Member Value: "enc"
  - Use Description: Encryption
7.3. JSON Web Key Set Parameters Registry

This specification establishes the IANA JSON Web Key Set Parameters registry for JWK Set parameter names. The registry records the parameter name and a reference to the specification that defines it. This specification registers the parameter names defined in Section 4.

7.3.1. Registration Template

Parameter Name:
   The name requested (e.g., "example"). Because a core goal of this specification is for the resulting representations to be compact, it is RECOMMENDED that the name be short -- not to exceed 8 characters without a compelling reason to do so. This name is case-sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Expert(s) state that there is a compelling reason to allow an exception in this particular case.

Parameter Description:
   Brief description of the parameter (e.g., "Example description").

Change Controller:
   For Standards Track RFCs, state "IESG". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):
   Reference to the document(s) that specify the parameter, preferably including URI(s) that can be used to retrieve copies of the document(s). An indication of the relevant sections may also be included but is not required.

7.3.2. Initial Registry Contents

   o Parameter Name: "keys"
   o Parameter Description: Array of JWK values
   o Change Controller: IESG
   o Specification Document(s): Section 4.1 of [[this document]]

7.4. Media Type Registration

7.4.1. Registry Contents

This specification registers the "application/jwk+json" and "application/jwk-set+json" Media Types [RFC2046] in the MIME Media Types registry [IANA.MediaTypes], which can be used to indicate,
respectively, that the content is a JWK or a JWK Set.

- **Type Name:** application
- **Subtype Name:** jwk+json
- **Required Parameters:** n/a
- **Optional Parameters:** n/a
- **Encoding considerations:** 8bit; application/jwk+json values are represented as JSON object; UTF-8 encoding SHOULD be employed for the JSON object.
- **Security Considerations:** See the Security Considerations section of [[ this document ]]
- **Interoperability Considerations:** n/a
- **Published Specification:** [[ this document ]]
- **Applications that use this media type:** TBD
- **Additional Information:** Magic number(s): n/a, File extension(s): n/a, Macintosh file type code(s): n/a
- **Person & email address to contact for further information:** Michael B. Jones, mbj@microsoft.com
- **Intended Usage:** COMMON
- **Restrictions on Usage:** none
- **Author:** Michael B. Jones, mbj@microsoft.com
- **Change Controller:** IESG

---

### 8. Security Considerations

All of the security issues faced by any cryptographic application
must be faced by a JWS/JWE/JWK agent. Among these issues are
protecting the user's private and symmetric keys, preventing various
attacks, and helping the user avoid mistakes such as inadvertently
encrypting a message for the wrong recipient. The entire list of
security considerations is beyond the scope of this document, but
some significant considerations are listed here.

One should place no more trust in the data associated with a key than
in than the method by which it was obtained and in the
trustworthiness of the entity asserting an association with the key.
Any data associated with a key that is obtained in an untrusted
manner should be treated with skepticism.

Private and symmetric keys MUST be protected from disclosure to
unintended parties. One recommended means of doing so is to encrypt
JWKs or JWK Sets containing them by using the JWK or JWK Set value as
the plaintext of a JWE.

The security considerations in RFC 3447 [RFC3447] and RFC 6030
[RFC6030] about protecting private and symmetric keys, key usage, and
information leakage also apply to this specification.

The security considerations in XML DSIG 2.0
[W3C.CR-xmldsig-core2-20120124], about key representations also apply
to this specification, other than those that are XML specific.

The TLS Requirements in [JWS] also apply to this specification.

9. References

9.1. Normative References

[ECMAScript]
Ecma International, "ECMAScript Language Specification,

[IANA.MediaTypes]
Internet Assigned Numbers Authority (IANA), "MIME Media
Types", 2005.

[ITU.X690.1994]
International Telecommunications Union, "Information
Technology - ASN.1 encoding rules: Specification of Basic
Encoding Rules (BER), Canonical Encoding Rules (CER) and
Distinguished Encoding Rules (DER)", ITU-T Recommendation
[JWA] Jones, M., "JSON Web Algorithms (JWA)",
draft-ietf-jose-json-web-algorithms (work in progress),
November 2013.

[JWE] Jones, M., Rescorla, E., and J. Hildebrand, "JSON Web
Encryption (JWE)", draft-ietf-jose-json-web-encryption
(work in progress), November 2013.

[JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web
Signature (JWS)", draft-ietf-jose-json-web-signature (work
in progress), November 2013.

[RFC1421] Linn, J., "Privacy Enhancement for Internet Electronic
Mail: Part I: Message Encryption and Authentication

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
Extensions (MIME) Part Two: Media Types", RFC 2046,
November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate


[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO

Resource Identifier (URI): Generic Syntax", STD 66,

[RFC4627] Crockford, D., "The application/json Media Type for
JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
Encodings", RFC 4648, October 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
IANA Considerations Section in RFCs", BCP 26, RFC 5226,
May 2008.


[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
Housley, R., and W. Polk, "Internet X.509 Public Key
Infrastructure Certificate and Certificate Revocation List
Appendix A. Example JSON Web Key Sets

A.1. Example Public Keys

The following example JWK Set contains two public keys represented as JWKs: one using an Elliptic Curve algorithm and a second one using an RSA algorithm. The first specifies that the key is to be used for encryption. The second specifies that the key is to be used with the "RS256" algorithm. Both provide a Key ID for key matching purposes. In both cases, integers are represented using the base64url encoding of their big endian representations. (Long lines are broken are for display purposes only.)
A.2. Example Private Keys

The following example JWK Set contains two keys represented as JWKs containing both public and private key values: one using an Elliptic Curve algorithm and a second one using an RSA algorithm. This example extends the example in the previous section, adding private key values. (Line breaks are for display purposes only.)
### A.3. Example Symmetric Keys

The following example JWK Set contains two symmetric keys represented as JWKs: one designated as being for use with the AES Key Wrap algorithm and a second one that is an HMAC key. (Line breaks are for display purposes only.)

```json
{"keys":
  [
    {"kty":"oct",
     "alg":"A128KW",
     "k":"GawgguFyGrWKav7AX4VKUg"},

    {"kty":"oct",
     "k":"AyM1SysPpbyDfgZld3umj1qzKObwVMkoqQ-EstJQLr_T-1qS0gZH75aKLMN3Yj0iPS4hcgUuTwjAzZr1Z9CAow",
     "kid":"HMAC key used in JWS A.1 example"}
  ]
}
```

### Appendix B. Example Use of "x5c" (X.509 Certificate Chain) Parameter
The following is an example of a JWK with a RSA signing key represented both as an RSA public key and as an X.509 certificate using the "x5c" parameter:

```
{"kty":"RSA",
"use":"sig",
"kid":"1b94c",
"n":"vrj0fz9Ccdgx5nQudyhdoR17V-IubWMeZOywX_jj0hgAsz2J_pqYW08
PLbK_PdiVGKPrqzmdISl7sA25VENH1uCLNwBuUIC011_-7dYbrs4iJmQGQ
u2j8DsVyT1azpJC-NQ84Ty5KKthuCapod71I7w0lk9orSMHEwwZDCxTwq4a
YWAc8ch8t-emd9g9QVw7VMDC2BXksRng96X5bUYLy6AyHKv-nUy1wgjyQDwH
MTp1ColTJU-o-8SNzN1tmRoGE9uJKLdh5gFENabWnU5m1zQzPdws-qo-meMV
Vfj6bJvWprl2SUtcNvG2c32qvbWbzJ_JBP5eunqsIoivQ",
"e":"AQAB",
"x5c":
["MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEBAQUA
fL8vIGATz/FuLiMA0GCSqGSIb3DQEBBQUAMGIx
CzAJB
gNVBAYTAlVTMQswCQYDVQQIEwJDTzEPMA0GA1UEBxMGRGVudmV
0Xy3jizw50aXRIENvcnAuMRcwFQYDVQQDEw5CmlhbhIiBDYW1
wYMVsbDAlEwFw8xMzAyMjEz5MTVaFw0xODA4MTQyMjI5MTVaMG
NVBAYTAlVTMQswCQYDVQQIEwJDTzEPMA0GA1UEBxMGRGVudm
QKExNQaw5nIE1kZw50aXRIENvcnAuMRcwFQYDVQQDEw5Cmlhbh
YmVsbDCCASiWeMQcCggEBAL64z4n8/QnH
YMeZ9LncoaEde1iMm1jHjQsF/4fl9IAML9i6famFpDyzy31iRjg6
s5gLyC07A0VRJx1NbgizCablGjtdf/u3W7K+IiZhtELto/A7Fck9W6
SqzRv0Ee8uSirYbmgj6He4i08NCyvaK0i1QRMGQswUSi2quG0
fajr1rVAtwtgVSLEXIel+i1WIG8ugMh4y4/p1MtcIM42E88ZsE62QGgC7VP
PvejJz2dbZkaBhPbiZAS3YeYBDRWm1p10ZtWamT3EcEvqQpPnjL1XyW+oyV
akZdk1lq2ptgt9qr2l4m2f4tw+Xrp6rCKNb0CawEAATANBgkqhkig9w0BA
QUFAAOCQAEAh8zGlfSlcI0o3rYDPBB07aXNsbw4EČNIK0G0CETTUXmXL9KUL
+9g6LqC2z5iWLoGwnrncKcy6YxXPG9J1r9AqBNTqNg9QG0X90266X5C0p0e1
zFo+0wb1xztP3pHdFfQj610CDLeaS9V9Rq17hCcybEp0Gvve8fkn+f+BE
2Bo3UPGrpsHzua6p6f7tMs2zKhpBjKVMJyf/RuP2SmmIzmwn9jI51Yyho
4tpzd5rFXhRbg4Zw9C+2qok+2+qDM1iJ6g4P8HM1y8aLWrdsQTxkumGnTq
gaw+R+N5MtdtPTEQ0xFIBc2cJEYuMYT5MVPvACWpkA6SdS4xSvdX3IVOWA=="
]}
```

**Appendix C. Example Encrypted RSA Private Key**

This example encrypts an RSA private key to the recipient using "PBES2-HS256+A128KW" for key encryption and "A128CBC+HS256" for content encryption.

**NOTE:** Unless otherwise indicated, all line breaks are included solely for readability.
### C.1. Plaintext RSA Private Key

The following RSA key is the plaintext for the encryption operation, formatted as a JWK object:

```json
{
  "kty": "RSA",
  "kid": "juliet@capulet.lit",
  "use": "enc",
  "n": "t6Q8PSwi1dKjJ9hTP8hNYFlvdM7DflW9mWepOJhJ66w7nyoK1gPNqFMSQDo01256pTEKodhWriuijHVX7bCv0L1s4w5ACGPrAdGzC_SR0-Iqom-QFcNP8Sjg086MwoqQU Lýw1AGZ21wSdsPeryGFiNnj3Q1Q08Yns5jCtLcrWHL0
Pb1fEv45AuR1uWfVCySBWYnDyGxvjYGDASM-AqWS92I2Zi1gT-GqUmpg9XOC0Cc20rgxeZyLMhpHc1CKVAbY5-L32-1SeZ0-0s6U15-aXrk9Gw8cPUaX1
_I8sLGuSiVdt3C_Fn2PZ38i744FPFGGcC1gs2Wz-Q",
  "e": "AQAB",
  "d": "GRtbIQmHz0tysfgKdg4u_N-R_mZGU_9k7Q_jn1DnfTuMdSNpTeaSTyWFS
NkuAwnOeIQvYv11QbwVV25NY3ybec_1hUJtfr17bAXYEREWaC13hd1PKxY9U
vqPYPGR0IKTQTQrns-dV7J7hjI7LyckrPtmMrM8dWb04_PmaenNniQ0g00xnu
ToxutRZjfvG40x4ka3GQR9dC5CC2zvsUDMsX0fEUOyMQAC6p1M3h3tssu
rY15k9qMSPg90XlIJAxmzAxh_tWiZ0wK2y4xH9ts3Lqy1x8C1EwmeRDk2a
hecG85-0LkQT5VEpWkmj0i_qj5d8gcG9N6X52esAq",
  "p": "2rnSOv4hKSN8sS4cQgCQFbs08XbfDQum3sC43GqrTmkDld1ZK9uw-PIHf
QP0fKxXVrX-WE-ZEbrqvih_21CLUS7wA16xvArTikIaUXPPSYB9yk31s0Q8
UK96E3_0r\ADAYTIAJ3S-3M3xClFNgqh5H6DHtnETTQh3rCTT3YJws",
  "q": "1u_RiFPD7LByh3N4GLT90pSKYpOQZyiaZwBo1C0BNJQxaj10RwjsZu0c6I
edIs47B_coSK80Kjs9PaBzg-iYsrVvcQpAmu61rMHvjVT66TlVBLCYK
Y152ziqK0E_ym2nQkwsUX7eYBT7LbAHRK9GqocDES50808I4s",
  "dp": "KkMTWqBUEvWvZ2_Dbj1pPQqyHSHjJ90L5_x_M0zqYAJMcLMTzbUtKwqVDQ3
tbE03zIcohbDtt6sbfMwZgagpbpXNuBpo0O_f-a_HgMXK_lhqiqI4_yKs1w
Y52IwjeUn5rgRJ-yio1h41KR-v22pYeAEyhrhtWtxVqLCRVIDc",
  "dq": "AvfS6-gRtxvn0bWJoMNsxFYck1WnuEjQFluMgfWitQBWtfz1r71xDbkN9
GQTb9yqPoDoYaN06H7CFtrkxh0BPaj6nkF5KK3TqT5q5czykxMe3K6bBy
mxxkBc5wqUpX5EL5D5xfC6FeiaFWY63TmmEAu_iRFCOJ3Dea-ots",
  "qi": "1SQi-wCpyUreMErP1rSbLK7wNt0vs5EQPQmumVqwW57NBuczScEoPwmmUqq
abu9V0-Py4dQ57_bapoKRu1R0bvuFnU63SHEFglZQVJDMeAvmj4sm-Fp0
Yu_neotQ0hbI5gry7ajdyYy9-21Nx_76aBZ00uU9HCJ-UsFSO18"
}
```

The octets representing the Plaintext are:

```
```
The following example JWE Protected Header declares that:

- the Content Encryption Key is encrypted to the recipient using the PSE2-HS256+A128KW algorithm to produce the JWE Encrypted Key,
- the Salt (p2s) is \[ 217, 96, 147, 112, 150, 117, 70, 247, 127, 8, 155, 137, 174, 42, 80, 215 \],
- the Iteration Count (p2c) is 4096,
- the Plaintext is encrypted using the AES_128_CBC_HMAC_SHA_256 algorithm to produce the Ciphertext, and
- the content type is application/jwk+json.

```
{  "alg":"PBES2-HS256+A128KW",  "p2s":"2WCTcJZ1Rvd_CCJuJripQ1w",  "p2c":4096,  "enc":"A128CBC-HS256",  "cty":"jwk+json" }
```

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

```
eyJhbGciOiJQQkVTMi1IUzI1NitBMTI4S1ciLCJwMnMiOiIyV0NUY0paMVJ2ZF9DSnVKcmlwUTF3IiwicDJjIjo0MDk2LCJlbmMiOiIjBMTI4Q0JDLUhTMjU2IiwY3R5IjoiandrK2pzb24ifQ
```

C.3. Content Encryption Key (CEK)

Generate a 256 bit random Content Encryption Key (CEK). In this example, the value is:

C.4. Key Encryption

Encrypt the CEK with a shared passphrase using the "PBES2-HS256+A128KW" algorithm and the specified Salt and Iteration Count values to produce the JWE Encrypted Key. This example uses the following passphrase:

Thus from my lips, by yours, my sin is purged.

The octets representing the passphrase are:


The resulting JWE Encrypted Key value is:

[ 201, 236, 143, 112, 12, 234, 200, 211, 33, 241, 255, 65, 112, 63, 172, 146, 105, 107, 122, 0, 30, 21, 44, 21, 14, 61, 200, 57, 30, 253, 228, 83, 218, 82, 138, 80, 121, 254, 193, 121 ]

Encoding this JWE Encrypted Key as BASE64URL(JWE Encrypted Key) gives this value:

yeyPcAzqyNMh8f9BcD-skmlregAeFSwVDj3IOR795FPaUopQef7BeQ

C.5. Initialization Vector

Generate a random 128 bit JWE Initialization Vector. In this example, the value is:

[ 97, 239, 99, 214, 171, 54, 216, 57, 145, 72, 7, 93, 34, 31, 149, 156 ]

Encoding this JWE Initialization Vector as BASE64URL(JWE Initialization Vector) gives this value:

Ye9j1qs22DmRSAAddIh-VnA

C.6. Additional Authenticated Data

Let the Additional Authenticated Data encryption parameter be ASCII(BASE64URL(UTF8(JWE Protected Header))). This value is:

[ 123, 34, 97, 108, 103, 34, 58, 34, 80, 66, 69, 83, 50, 45, 72, 83, 50, 53, 54, 43, 65, 49, 50, 56, 75, 87, 34, 44, 34, 112, 50, 115, 34, 58, 34, 50, 87, 67, 84, 99, 74, 90, 49, 82, 118, 100, 95, 67, 74,
C.7. Content Encryption

Encrypt the Plaintext with AES_128_CBC_HMAC_SHA_256 using the CEK as the encryption key, the JWE Initialization Vector, and the Additional Authenticated Data value above. The resulting Ciphertext is:

```
```
The resulting Authentication Tag value is:


Encoding this JWE Ciphertext as BASE64URL(JWE Ciphertext) gives this value:
AwhB8xlrlKjFm02LGWEqg27H4T9fyZAbFv3p5ZicHpj64QyHC4qqlZ3JEmnZTqgo
wIqZ13jbyhB8lgePiqUi1hf6MHPLgz8w-5-eMe0qjvDUTjE07nt0erb8bwBQyZ6g
0kq3DEQ0g1fyXv8-F3JvNyWbwqD8IbcJd_70tjSHV-8Dlrrp-3JcrIe05Yk30i34zG
0iA1cEK21B11c_5E_11PII_wvvtRiUiG0YofQXakWd1_098Kap-UgmyWPFreU3J1JP
nb4DVe95oweFMGLQF10l2mnjaTDQwokoj_x_pql2vnpz8igulChiBoK1ilyQFJL2mOWB
wqhBo90j-0800as5mnLSvQMTFlr1EebTMZHM8ZB8EFw9fWwF0IrDQ3GKnMnhBZQ-3l
vqTc-Me0GA6D8hOnFD2p0i2bGHiJzwG12EaxBGRyaUpFLuljCLie1dKgoewhKkKZ
h04DKNM5Nbugf2atmu90p0lxd5peCUTr1G1MV17up5ZTXHTjgPDrb2N731UooCgAU
qHdg0hG0JVJObCTdjsJH4CF1SJsudhRvxvXy3HhJ2x7cKw3JRZU3_y16xYU6-s3GFpbi
rfqeqIPDBTHpcoCmmyrYjHyFglnq1BZorrtSrS95gf89fBjRqsaD7yUQGwQBwvy665
D0zvpTasvxfX_c0MWAl-neFaK0W_Px6g4EUDjG1GWXV9cLStLW0vdpApDIFLHfHe
PyagyHjouQUUgiq7BsyYWyrwaF06tgBBhV80omLFmEMDVpJaZUmuHzHw6tBDWgKzd-Ts-
UB9xhsp4JUS0Wnt5rGy0n2N_c1-TQXxmt5otol14MnxoAyQBpwIEgSH3Y4ZhhwKhBH
PjSoCdwuNdYbgPpb-YUVF-2NZodq10vWQBRSbPwYz_xBGkgD504LRtqRwC07CC
_CyyUrUEssPVsMJRX_U4LFE0c82tiDdqkJ0rfK5rL18nBE9soQD0saOOFQzi
GrBqxsDNYiAYAmxxxkos-13nX4qtByVx85sCE5U_0Mq67C0xZWMOPFrdDepaUV-Oy
rvoUInq81i8jKBKxETY2BgpPegKBYCxsAUCkAmkSCK9A1BxA0U0HyhTqt1lMks07AE
hNC2-YzPyx1FkhMo44Le6E_pFSM1mj9A6PINSge9C5G5tETYXGn6b1x2BhmtwRPSc
ro9LWhmvmaA7_Bx0bnFUXgKtv4zQ0BJ36UTk40T-JvkKwgfVwCfsaw5WCJH600
4j0p07d2yN7WMfAja2hTEabz9wumQ0TMhbduZ-QQN3p0ybSy7TSC1vVMe0Njrwc_cJre
hKTfmd1XGVidPzCpilr7ZqQrQFhFJ3-14mEQVnCawGn9ONHemczGOS-A-wtnmwj1
B1v_vqJRF4dpv4-4uhk4-4Lpqu3-11WFxrtzKcggq3tWtduRo5_QubQbUUT_VSCqgsFc
OmyWkoj561bxth19hq1XGwbLGfrR6MWh23vk01zn8Fvwi7FwEnRYSaFsnwaLai2S5
Tpb9jGvAd12H9HwzpBSNqHPzNKQ3NMDj13Fn8fz0OB53B3etbm_tnFqfcb13x3b1J5
Cw-xW1MghivPpGnMBT_ADp9xiS1yAM9qc1yeVXk-AigWBULN5uyWSgyXcp0Cjxv7HXM
3820UieT-Mylte-qndmMLhytxsZvbj0TSVRmhyEM1ZuAAsiSgs7uMqARqRdRLFETJES
Gmgj_4bZq9s6ve1LKksio_Q0srAbaLe55UY0f4Z5f0V5MPtyc0vDcNP1xlgNA
D1BFx_Z9kadMZQW6fAmSffl0eZoaMe41p9mMES0HB4sJgDcKtxQj1cXNyDYoZf718
H00BV_97zd6tVlO0MxwvFCatstV_R-GsBC218RvPSfYHwUt8R4HarpsD8Bu4c
r8_c8fc92Z78sQ08jFj0ja6Lx2On_ImzFNeXuwzO-Ska-QuevYz3X_L31OXL4lep-
7QSFgDOnHX0Fv1XwM5Dm3XozOuwp2TdpdKTz89eWv2uxVUViM8019atbfPKMG0A
v90m-A-6v5IxDUH-LWMIHLQ-g8vnswp-JaV0c4t6URVUuzijnNOtDCBGGVnHiJTCH1
88LQxsQfLHIN4fZ-2USGn1xG7tj0-iihit2ELGrv4v08E1BosTm0c3q3gGOpq0eOLBD
IHsrdZ_CCAlTc0HVKnEyqlM6qehM-5qP6y1QCIrwr

Encoding this JWE Authentication Tag as BASE64URL(JWE Authentication Tag) gives this value:

ffmPv_AEzIQ-8XGyW1j-Ew

C.8. Complete Representation

Assemble the final serialization of this result is the string BASE64URL(UTF8(JWE Protected Header)) || '.' || BASE64URL(JWE Encrypted Key) || '.' || BASE64URL(JWE Initialization Vector) || '.' || BASE64URL(JWE Ciphertext) || '.' || BASE64URL(JWE Authentication Tag).
The final result in this example is:

eyJhbGciOiJQQkVTMi1IUzI1NitBMTI4S1ciLCJwMnMiOiIyV0NUY0paMVJ2ZF9DSnVKeP1wUWF3IiwiDjijOi0MDk2LCJlbmMiOiJjbMTI4Q0JDLUhTMjU2IiwY3R5IjoiandrK2pzb24ifQ.

yeypCAzqyNMhfF9BcD-skmlregAEFSwVDj3I0R795FPaUopQef7BeQ.

Ye9jiqks22DrmSadRh-VnA.

AwhB8lxlr1kjfn02LWEqf27H4tg9fyZAbFv3pZiChp64QyHC44qqlZ3JEmnZTgQ0wIqZj13jbyHB8gLePiqUj1hfd6M2HFLGzw8L-mEeQ0jvDUTrE07p0erBk8bwBQYz6g0qK3DEQIglfYFxV8-FJvNBYwbqN1Bckd6_i7otjSHVD8Drp-3JcRie05YKyo31342G0iaC1kE2z1bIC1EI1PIIw-vwrtU1wGBYyqfXakWd1_098Kap-UgmyWPFreuU3J1JPNbD4Ve95owEFMGLOPfl02MnjayTDcWqokoJ_qpl2QVNZgliCLhBoK1lyQFJL2mOWBwqhb090j-o800as5mmLSvQMTFlbEItbMzHM6ZBE8WF9fwwFwF0DwQ3GKMNhmBZQ-31qvTc-M6-gwAD08PD0NPzP02ib2HGizw611EaX8GQyupLFujLClle1DKG0ewHuKkKkK404KMN5BqRt2aMu90pOLdx5peCUTR1qGMy17Up5ZXHTJgPDr5b2N731uoCAAGqHdgGyOHVJ-0bCdtdjsh4CF1sJSjdUhrVxyx3HJk2x7Cw7J1zR3y316xYU6-s3GPbFirfqqEipJDBTPc0mCmyrerYyJhfGnlqBZRotRs95qBf95bRQxsady7UQGQwBqWy665d0zpyTasvxf_c0MWAl-neFaKOW_Px6g4EUDJ1G7X5WV9cLStlw_p0odvApDlHLHYHePagyHjouQQUuQiy7BwYwraF60tgB8h8VomLNFMEmpDPjAZuMhWw6tB DwGKzd-ts_uhxb9p45mUsWnt5rG0yV0nN2n_c1-TQ1Lxmx5oto14MnxoAyBQpwIEdGSH3Y4zhwKBHkJoq6cdwuNdYbGpb-v2YNzZ0diQ10vWQBRSbPwry_zbGkgD564LRTqRwc07CC_CyuyUI1iEssPvMsJRX_U4LFE0c82T1d6dqK0jRufK5q1L81ne9soQD0SaOoFQz1GrBqxDsNY1AYamxkos-13n4x4tybYyV85sCE5U_0MqG7COxzwMOEPFDaepUcU-voyrvUnI98jLBkKETY2BgPegKBYCxsAUCpAmKSC9A1bxA0U0Hyhtq1tvMks07AEnHC2-YzxP1FkhM0s4LLe6E_PFsm1mjAJ6PSInsge9C5G5tETYX6An6bj1xZbHtmwrPScroc9LWhVAmAaA7_bxY0bfnFUXgWTk4vzQBjZJ36UtK40TB-QvkgfVWCFsa5WCHJ60o4jop07d2yN7Wmafj2hTeb29wuqM0THMbdzu-ZQhNS3yObSY7TS1CvVMe0Njrwf_cjRehKTFmdlxY1vDpxCPlr172ZqRqFhFBP-14meEqVnCalGWn9NHLemczGOS-A-wttnmwjB1v_vgJrF4Fdpv-4uHkQ-4Lpum3-11FWfxtZKcqu3tWTDuRo5_qeboqUUT_VSCgsFcOmyWkjoj561bxthN19hqXGWBFlgfrrR6MwhV3vk01zn8FVw17uFenRYSafswLAa1ZSTpbjgVad12H9HnwhzBS5gHqZ1QN3NMDj13Fn8f0z0JB83Etbn textStatusFqfb13k3Xb15JCw-Zw1MghVipGgNMBT-Adpx9SIyAM90q1yeVXK-AiGwBUL5NuWSyGxCp0cJxw7XHM38Z0UcBu-MytL-eqnMD7LxysVzCb0jTOSVrhYEM1ZuUAniugs7uMqAGQrdgRIETJEsgmjb-4zbq9s6v1LkKsi0_QQsrABaLe55UY6f4ZSf0VP5MypTocwWdcNP1xLgNaD1BFX-Z9KadMrQzw6fJmsffleOzaAm419pMESH0JB4sJGdcKtQjiciXNdYoFzFS7F8H00BV_Er7zd6IVtW0MxwkJCTatsv_R-GSBCh218rgVpF2yWvTuT8R4HarpsDBucf4r8_c8fc9Z78s081jFj0a6L2x0N_iMzFNXU6xwo-Ska-QeuvY3XZL31Z04c11-7q5fFdH0nx0xFv1xwz-D5mHDD3x0u2p2tTppdKTB9eW2vxUVv1M8019atatBPKMGAo9vom4-A6v5ixUH0-L1W18nswp-Jav0c4t6URVuzuNOoNdCBGgVnH1JTCHJ188LqxivsLHHIIU4rz-U2SGnLxGTj0-itih2ELGrVr4v0E81BosTmfc0x3qg6P0qe0OLBDIHsrdZ_CCAiTC0HVnMybq1M6qEhM-q5P6y1QCIrwg.

ffmPv_AEZIQ-8XYw1j-Ew
This specification is the work of the JOSE Working Group, which includes dozens of active and dedicated participants. In particular, the following individuals contributed ideas, feedback, and wording that influenced this specification:


Jim Schaad and Karen O'Donoghue chaired the JOSE working group and Sean Turner and Stephen Farrell served as Security area directors during the creation of this specification.

Appendix E. Document History

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

-18

- Changes to address editorial and minor issues #68, #69, #73, #74, #76, #77, #78, #79, #82, #85, #89, and #135.

- Added and used Description registry fields.

-17

- Refined the "typ" and "cty" definitions to always be MIME Media Types, with the omission of "application/" prefixes recommended for brevity, addressing issue #50.

- Added an example encrypting an RSA private key with "PBES2-HS256+A128KW" and "A128CBC-HS256". Thanks to Matt Miller for producing this!

- Processing rules occurring in both JWS and JWK are now referenced in JWS by JWK, rather than duplicated, addressing issue #57.

- Terms used in multiple documents are now defined in one place and incorporated by reference. Some lightly used or obvious terms were also removed. This addresses issue #58.

-16
Changes to address editorial and minor issues #41, #42, #43, #47, #51, #67, #71, #76, #80, #83, #84, #85, #86, #87, and #88.

Changes to address editorial issues #48, #64, #65, #66, and #91.

Relaxed language introducing key parameters since some parameters are applicable to multiple, but not all, key types.

Applied spelling and grammar corrections.

Stated that recipients MUST either reject JWKs and JWK Sets with duplicate member names or use a JSON parser that returns only the lexically last duplicate member name.

Stated that when "kid" values are used within a JWK Set, different keys within the JWK Set SHOULD use distinct "kid" values.

Added optional "x5u" (X.509 URL), "x5t" (X.509 Certificate Thumbprint), and "x5c" (X.509 Certificate Chain) JWK parameters.

Added section on Encrypted JWK and Encrypted JWK Set Formats.

Added a Parameter Information Class value to the JSON Web Key Parameters registry, which registers whether the parameter conveys public or private information.

Registered "application/jwk+json" and "application/jwk-set+json" MIME types and "JWK" and "JWK-SET" typ header parameter values, addressing issue #21.

No changes were made, other than to the version number and date.

Expanded the scope of the JWK specification to include private and symmetric key representations, as specified by
- Defined that members that are not understood must be ignored.

- Changed the name of the JWK key type parameter from "alg" to "kty" to enable use of "alg" to indicate the particular algorithm that the key is intended to be used with.

- Clarified statements of the form "This member is OPTIONAL" to "Use of this member is OPTIONAL".

- Referenced String Comparison Rules in JWS.

- Added seriesInfo information to Internet Draft references.

- Changed the name of the JWK RSA modulus parameter from "mod" to "n" and the name of the JWK RSA exponent parameter from "xpo" to "e", so that the identifiers are the same as those used in RFC 3447.

- Changed the name of the JWK RSA exponent parameter from "exp" to "xpo" so as to allow the potential use of the name "exp" for a future extension that might define an expiration parameter for keys. (The "exp" name is already used for this purpose in the JWT specification.)

- Clarify that the "alg" (algorithm family) member is REQUIRED.

- Correct an instance of "JWK" that should have been "JWK Set".

- Applied changes made by the RFC Editor to RFC 6749's registry language to this specification.

- Indented artwork elements to better distinguish them from the body text.

- Refer to the registries as the primary sources of defined values and then secondarily reference the sections defining the initial
contents of the registries.


- Added this language to Registration Templates: "This name is case sensitive. Names that match other registered names in a case insensitive manner SHOULD NOT be accepted."

- Described additional open issues.

- Applied editorial suggestions.

-03

- Clarified that "kid" values need not be unique within a JWK Set.

- Moved JSON Web Key Parameters registry to the JWK specification.

- Added "Collision Resistant Namespace" to the terminology section.

- Changed registration requirements from RFC Required to Specification Required with Expert Review.

- Added Registration Template sections for defined registries.

- Added Registry Contents sections to populate registry values.

- Numerous editorial improvements.

-02

- Simplified JWK terminology to get replace the "JWK Key Object" and "JWK Container Object" terms with simply "JSON Web Key (JWK)" and "JSON Web Key Set (JWK Set)" and to eliminate potential confusion between single keys and sets of keys. As part of this change, the top-level member name for a set of keys was changed from "jwk" to "keys".

- Clarified that values with duplicate member names MUST be rejected.

- Established JSON Web Key Set Parameters registry.

- Explicitly listed non-goals in the introduction.

- Moved algorithm-specific definitions from JWK to JWA.
Reformatted to give each member definition its own section heading.

-01

Corrected the Magic Signatures reference.

-00

Created the initial IETF draft based upon draft-jones-json-web-key-03 with no normative changes.

Author's Address

Michael B. Jones
Microsoft

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