JOSE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 27, 2015

M. Jones
Microsoft

October 24, 2014

JSON Web Key (JWK)
draft-ietf-jose-json-web-key-36

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 that represents 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 April 27, 2015.

Copyright Notice

Copyright (c) 2014 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. Example JWK 5
4. JSON Web Key (JWK) Format 5
4.1. "kty" (Key Type) Parameter 6
4.2. "use" (Public Key Use) Parameter 6
4.3. "key_ops" (Key Operations) Parameter 7
4.4. "alg" (Algorithm) Parameter 8
4.5. "kid" (Key ID) Parameter 8
4.6. "x5u" (X.509 URL) Parameter 8
4.7. "x5c" (X.509 Certificate Chain) Parameter 9
4.8. "x5t" (X.509 Certificate SHA-1 Thumbprint) Parameter . . . 9
4.9. "x5t#S256" (X.509 Certificate SHA-256 Thumbprint) Parameter 10
5. JSON Web Key Set (JWK Set) Format 10
5.1. "keys" Parameter 10
6. String Comparison Rules 11
7. Encrypted JWK and Encrypted JWK Set Formats 11
8. IANA Considerations 11
8.1. JSON Web Key Parameters Registry 13
8.1.1. Registration Template 13
8.1.2. Initial Registry Contents 14
8.2. JSON Web Key Use Registry 15
8.2.1. Registration Template 15
8.2.2. Initial Registry Contents 16
8.3. JSON Web Key Operations Registry 16
8.3.1. Registration Template 16
8.3.2. Initial Registry Contents 17
8.4. JSON Web Key Set Parameters Registry 18
8.4.1. Registration Template 18
8.4.2. Initial Registry Contents 19
8.5. Media Type Registration 19
8.5.1. Registry Contents 19
9. Security Considerations 20
9.1. Key Provenance and Trust 20
9.2. Preventing Disclosure of Non-Public Key Information . . . 20
9.3. RSA Private Key Representations and Blinding 21
9.4. Key Entropy and Random Values 21
10. References 21
10.1. Normative References 21
10.2. Informative References 23

Appendix A. Example JSON Web Key Sets 24
A.1. Example Public Keys 24
A.2. Example Private Keys 24
A.3. Example Symmetric Keys 26
Appendix B. Example Use of "x5c" (X.509 Certificate Chain) Parameter 26
Appendix C. Example Encrypted RSA Private Key 27
C.1. Plaintext RSA Private Key 28
C.2. JOSE Header 31
C.3. Content Encryption Key (CEK) 31
C.4. Key Derivation 32
C.5. Key Encryption 32
C.6. Initialization Vector 32
C.7. Additional Authenticated Data 33
C.8. Content Encryption 33
C.9. Complete Representation 36
Appendix D. Acknowledgements 37
Appendix E. Document History 38
Author's Address 45

1. Introduction

A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) [RFC7159] data structure that represents a cryptographic key. This specification also defines a JSON Web Key Set (JWK Set) JSON data structure that represents 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 new kinds of certificate chains, representing new kinds of certified keys, or 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", "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 of [JWS].

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

ASCII(STRING) denotes the octets of the ASCII [RFC20] 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", "Collision-Resistant Name", "Header Parameter", and "JOSE Header".

These terms defined by the Internet Security Glossary, Version 2 [RFC4949] are incorporated into this specification: "Ciphertext", "Digital Signature", "Message Authentication Code (MAC)", and "Plaintext".

These terms are defined by this specification:

JSON Web Key (JWK)
: A JSON object that represents a cryptographic key. The members of the object represent properties of the key, including its value.

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

3. Example JWK

This section provides an example of a JWK. The following example JWK declares that the key is an Elliptic Curve [DSS] key, it is used with the P-256 Elliptic Curve, and its x and y coordinates are the base64url encoded values shown. A key identifier is also provided for the key.

```
{"kty":"EC",
 "crv":"P-256",
 "x":"f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
 "y":"x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
 "kid":"Public key used in JWS A.3 example"
}
```

Additional example JWK values can be found in Appendix A.

4. JSON Web Key (JWK) Format

A JSON Web Key (JWK) is a JSON object that represents a cryptographic key. The members of the object represent properties of the key, including its value. This JSON object MAY contain white space and/or line breaks before or after any JSON values or structural characters, in accordance with Section 2 of RFC 7159 [RFC7159]. 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 key type-specific. 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; JWK parsers 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 8.1 or be a value that contains a Collision-Resistant Name.

4.1. "kty" (Key Type) Parameter

The "kty" (key type) member identifies the cryptographic algorithm family used with the key, such as "RSA" or "EC". "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 8.1.

4.2. "use" (Public Key Use) Parameter

The "use" (public key use) member identifies the intended use of the public key. The "use" parameter is employed to indicate whether a public key is used for encrypting data or verifying the signature on data.

Values defined by this specification are:

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

Other values MAY be used. The "use" value is a case-sensitive string. Use of the "use" member is OPTIONAL, unless the application requires its presence.

When a key is used to wrap another key and a Public Key Use designation for the first key is desired, the "enc" (encryption) key use value is used, since key wrapping is a kind of encryption. The "enc" value is also be used for public keys used for key agreement operations.

Additional Public Key Use values can be registered in the IANA JSON Web Key Use registry defined in Section 8.2. Registering any extension values used is highly recommended when this specification is used in open environments, in which multiple organizations need to have a common understanding of any extensions used. However, unregistered extension values can be used in closed environments, in which the producing and consuming organization will always be the same.

4.3. "key_ops" (Key Operations) Parameter

The "key_ops" (key operations) member identifies the operation(s) that the key is intended to be used for. The "key_ops" parameter is intended for use cases in which public, private, or symmetric keys may be present.

Its value is an array of key operation values. Values defined by this specification are:

- "sign" (compute digital signature or MAC)
- "verify" (verify digital signature or MAC)
- "encrypt" (encrypt content)
- "decrypt" (decrypt content and validate decryption, if applicable)
- "wrapKey" (encrypt key)
- "unwrapKey" (decrypt key and validate decryption, if applicable)
- "deriveKey" (derive key)
- "deriveBits" (derive bits not to be used as a key)

(Note that the "key_ops" values intentionally match the "KeyUsage" values defined in the Web Cryptography API [WebCrypto] specification.)

Other values MAY be used. The key operation values are case-sensitive strings. Duplicate key operation values MUST NOT be present in the array. Use of the "key_ops" member is OPTIONAL, unless the application requires its presence.

Multiple unrelated key operations SHOULD NOT be specified for a key because of the potential vulnerabilities associated with using the same key with multiple algorithms. Thus, the combinations "sign" with "verify", "encrypt" with "decrypt", and "wrapKey" with "unwrapKey" are permitted, but other combinations SHOULD NOT be used.

Additional Key Operations values can be registered in the IANA JSON Web Key Operations registry defined in Section 8.3. The same considerations about registering extension values apply to the "key_ops" member as do for the "use" member.

The "use" and "key_ops" JWK members SHOULD NOT be used together; however, if both are used, the information they convey MUST be consistent. Applications should specify which of these members they use, if either is to be used by the application.

4.4. "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.

4.5. "kid" (Key ID) Parameter

The "kid" (key ID) member is used to match a specific key. This is 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.

4.6. "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, with each certificate delimited as specified in Section 6.1 of RFC 4945 [RFC4945]. 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 6 of RFC 6125 [RFC6125]. Use of this member is OPTIONAL.

While there is no requirement that optional JWK members providing key usage, algorithm, or other information be present when the "x5u" member is used, 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 MUST correspond to the usage that is specified in the certificate, when it includes this information. Similarly, if the "alg" member is present, it MUST correspond to the algorithm specified in the certificate.

4.7. "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, optional JWK members providing key usage, algorithm, or other information MAY also be present when the "x5c" member is used. 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 4.6 for additional guidance on this.

4.8. "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]. Note that certificate thumbprints are also sometimes known as certificate fingerprints. The key in the certificate MUST match the public key represented by other members of the JWK. Use of this member is OPTIONAL.

As with the "x5u" member, optional JWK members providing key usage, algorithm, or other information MAY also be present when the "x5t" member is used. 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 4.6 for additional guidance on this.

4.9. "x5t#S256" (X.509 Certificate SHA-256 Thumbprint) Parameter

The "x5t#S256" (X.509 Certificate SHA-256 Thumbprint) member is a base64url encoded SHA-256 thumbprint (a.k.a. digest) of the DER encoding of an X.509 certificate [RFC5280]. Note that certificate thumbprints are also sometimes known as certificate fingerprints. The key in the certificate MUST match the public key represented by other members of the JWK. Use of this member is OPTIONAL.

As with the "x5u" member, optional JWK members providing key usage, algorithm, or other information MAY also be present when the "x5t#S256" member is used. 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 4.6 for additional guidance on this.

5. JSON Web Key Set (JWK Set) Format

A JSON Web Key Set (JWK Set) is a JSON object that represents a set of JWKs. The JSON object MUST have a "keys" member, with its value being an array of JWK objects. This JSON object MAY contain white space and/or line breaks.

The member names within a JWK Set MUST be unique; JWK Set parsers 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 8.4 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.

5.1. "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.

6. String Comparison Rules

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

7. Encrypted JWK and Encrypted JWK Set Formats

Access to JWKs containing non-public key material by parties without legitimate access to the non-public information MUST be prevented. This can be accomplished by encrypting the JWK when potentially observable by such parties to prevent the disclosure of private or symmetric key values. 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, in which case the "cty" value would typically be omitted.

JWK Sets containing non-public key material will also need to be encrypted under these circumstances. 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, in which case the "cty" value would typically be omitted.

See Appendix C for an example encrypted JWK.

8. IANA Considerations

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

Values are registered on a Specification Required [RFC5226] basis after a three-week review period on the jose-reg-review@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 jose-reg-review@ietf.org mailing list for review and comment, with an appropriate subject (e.g., "Request for access token type: example").

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@ietf.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 description is clear.

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).

[[Note to the RFC Editor and IANA: Pearl Liang of ICANN had requested that the draft supply the following proposed registry description information. It is to be used for all registries established by this specification.

- Protocol Category: JSON Object Signing and Encryption (JOSE)

- Registry Location: http://www.iana.org/assignments/jose

- Webpage Title: (same as the protocol category)

- Registry Name: (same as the section title, but excluding the word "Registry", for example "JSON Web Key Parameters")

]]

8.1. 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 4. 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.

8.1.1. Registration Template

Parameter Name:
: The name requested (e.g., "kid"). 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., "Key ID").

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-insensitive 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 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.

8.1.2. Initial Registry Contents

- Parameter Name: "kty"
- Parameter Description: Key Type
- Used with "kty" Value(s): *
- Parameter Information Class: Public
- Change Controller: IESG
- Specification Document(s): Section 4.1 of [[this document]]

- Parameter Name: "use"
- Parameter Description: Public Key Use
- Used with "kty" Value(s): *
- Parameter Information Class: Public
- Change Controller: IESG
- Specification Document(s): Section 4.2 of [[this document]]

- Parameter Name: "key_ops"
- Parameter Description: Key Operations
- Used with "kty" Value(s): *
- Parameter Information Class: Public
- Change Controller: IESG
- Specification Document(s): Section 4.3 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 4.4 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 4.5 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 4.6 of [[this document]]

- Parameter Name: "x5c"
- Parameter Description: X.509 Certificate Chain
- Used with "kty" Value(s): *
- Parameter Information Class: Public
- Change Controller: IESG
- Specification Document(s): Section 4.7 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 4.8 of [[this document]]

- Parameter Name: "x5t#S256"
- Parameter Description: X.509 Certificate SHA-256 Thumbprint
- Used with "kty" Value(s): *
- Parameter Information Class: Public
- Change Controller: IESG
- Specification Document(s): Section 4.9 of [[this document]]

8.2. JSON Web Key Use Registry

This specification establishes the IANA JSON Web Key Use registry for JWK "use" (public key use) member values. The registry records the public key use value and a reference to the specification that defines it. This specification registers the parameter names defined in Section 4.2.

8.2.1. Registration Template

Use Member Value:
: The name requested (e.g., "sig"). 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., "Digital Signature or MAC").

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.

8.2.2. Initial Registry Contents

- Use Member Value: "sig"
- Use Description: Digital Signature or MAC
- Change Controller: IESG
- Specification Document(s): Section 4.2 of [[this document]]

- Use Member Value: "enc"
- Use Description: Encryption
- Change Controller: IESG
- Specification Document(s): Section 4.2 of [[this document]]

8.3. JSON Web Key Operations Registry

This specification establishes the IANA JSON Web Key Operations registry for values of JWK "key_ops" array elements. The registry records the key operation value and a reference to the specification that defines it. This specification registers the parameter names defined in Section 4.3.

8.3.1. Registration Template

Key Operation Value:
The name requested (e.g., "sign"). 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.

Key Operation Description:
Brief description of the key operation (e.g., "Compute digital signature or MAC").

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.

8.3.2. Initial Registry Contents

- Key Operation Value: "sign"
- Key Operation Description: Compute digital signature or MAC
- Change Controller: IESG
- Specification Document(s): Section 4.3 of [[this document]]

- Key Operation Value: "verify"
- Key Operation Description: Verify digital signature or MAC
- Change Controller: IESG
- Specification Document(s): Section 4.3 of [[this document]]

- Key Operation Value: "encrypt"
- Key Operation Description: Encrypt content
- Change Controller: IESG
- Specification Document(s): Section 4.3 of [[this document]]

- Key Operation Value: "decrypt"
- Key Operation Description: Decrypt content and validate decryption, if applicable
- Change Controller: IESG
- Specification Document(s): Section 4.3 of [[this document]]

- Key Operation Value: "wrapKey"
- Key Operation Description: Encrypt key
- Change Controller: IESG
- Specification Document(s): Section 4.3 of [[this document]]

- Key Operation Value: "unwrapKey"
- Key Operation Description: Decrypt key and validate decryption, if applicable

- Change Controller: IESG
- Specification Document(s): Section 4.3 of [[this document]]

- Key Operation Value: "deriveKey"
- Key Operation Description: Derive key
- Change Controller: IESG
- Specification Document(s): Section 4.3 of [[this document]]

- Key Operation Value: "deriveBits"
- Key Operation Description: Derive bits not to be used as a key
- Change Controller: IESG
- Specification Document(s): Section 4.3 of [[this document]]

8.4. 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 5.

8.4.1. Registration Template

Parameter Name:
: The name requested (e.g., "keys"). 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., "Array of JWK values").

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.

8.4.2. Initial Registry Contents

- Parameter Name: "keys"
- Parameter Description: Array of JWK values
- Change Controller: IESG
- Specification Document(s): Section 5.1 of [[this document]]

8.5. Media Type Registration

8.5.1. 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] in the manner described in RFC 6838 [RFC6838], 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: OpenID Connect, Salesforce, Google, Android, Windows Azure, W3C WebCrypto API, numerous others
- Fragment identifier considerations: n/a
- 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
- Provisional registration? No

- Type Name: application
- Subtype Name: jwk-set+json
- Required Parameters: n/a
- Optional Parameters: n/a
- Encoding considerations: 8bit; application/jwk-set+json values are represented as a 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: OpenID Connect, Salesforce, Google, Android, Windows Azure, W3C WebCrypto API, numerous others
- Fragment identifier considerations: n/a
- 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
- Provisional registration? No

9. Security Considerations

All of the security issues that are pertinent to any cryptographic application must be addressed by JWS/JWE/JWK agents. Among these issues are protecting the user's asymmetric private and symmetric secret keys and employing countermeasures to various attacks.

9.1. Key Provenance and Trust

One should place no more trust in the data cryptographically secured by a key than in 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. See Section 10.3 of [JWS] for security considerations on key origin authentication.

The security considerations in Section 12.3 of XML DSIG 2.0 [W3C.NOTE-xmldsig-core2-20130411] about the strength of a digital signature depending upon all the links in the security chain also apply to this specification.

The TLS Requirements in Section 8 of [JWS] also apply to this specification.

9.2. Preventing Disclosure of Non-Public Key Information

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. Of course, this requires that there be a

secure way to obtain the key used to encrypt the non-public key information to the intended party and a secure way for that party to obtain the corresponding decryption key.

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.

9.3. RSA Private Key Representations and Blinding

The RSA Key blinding operation [Kocher], which is a defense against some timing attacks, requires all of the RSA key values "n", "e", and "d". However, some RSA private key representations do not include the public exponent "e", but only include the modulus "n" and the private exponent "d". This is true, for instance, of the Java RSAPrivateKeySpec API, which does not include the public exponent "e" as a parameter. So as to enable RSA key blinding, such representations should be avoided. For Java, the RSAPrivateCrtKeySpec API can be used instead. Section 8.2.2(i) of the Handbook of Applied Cryptography [HAC] discusses how to compute the remaining RSA private key parameters, if needed, using only "n", "e", and "d".

9.4. Key Entropy and Random Values

See Section 10.1 of [JWS] for security considerations on key entropy and random values.

10. References

10.1. Normative References

[ECMAScript]
Ecma International, "ECMAScript Language Specification, 5.1 Edition", ECMA 262, June 2011.

[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 X.690, 1994.

[JWA] Jones, M., "JSON Web Algorithms (JWA)", draft-ietf-jose-json-web-algorithms (work in progress), October 2014.

[JWE] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption (work in progress), October 2014.

[JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", draft-ietf-jose-json-web-signature (work in progress), October 2014.

[RFC20] Cerf, V., "ASCII format for Network Interchange", RFC 20, October 1969.

[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 Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

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

[RFC4945] Korver, B., "The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX", RFC 4945, August 2007.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 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 (CRL) Profile", RFC 5280, May 2008.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

10.2. Informative References

[DSS] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS PUB 186-4, July 2013.

[HAC] Menezes, A., van Oorschot, P., and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996, <http://cacr.uwaterloo.ca/hac/about/chap8.pdf>.

[Kocher] Kocher, P., "Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS, and Other Systems", In Proceedings of the 16th Annual International Cryptology Conference Advances in Cryptology, Springer-Verlag, pp. 104-113, 1996.

[MagicSignatures] Panzer (editor), J., Laurie, B., and D. Balfanz, "Magic Signatures", January 2011.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

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

[RFC6030] Hoyer, P., Pei, M., and S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, October 2010.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[W3C.NOTE-xmldsig-core2-20130411] Eastlake, D., Reagle, J., Solo, D., Hirsch, F., Roessler, T., Yiu, K., Datta, P., and S. Cantor, "XML Signature Syntax and Processing Version 2.0", World Wide Web Consortium Note NOTE-xmldsig-core2-20130411, April 2013,

<http://www.w3.org/TR/2013/NOTE-xmldsig-core2-20130411/>.

[WebCrypto]
Sleevi, R. and M. Watson, "Web Cryptography API", World Wide Web Consortium Draft, March 2014, <http://www.w3.org/TR/2014/WD-WebCryptoAPI-20140325/>.

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.)

```
  {"keys":
    [
      {"kty":"EC",
       "crv":"P-256",
       "x":"MKBCTNIcKUSDii11ySs3526iDZ8AiTo7Tu6KPAqv7D4",
       "y":"4Etl6SRW2YiLUrN5vfvVHuhp7x8PxltmWWlbbM4IFyM",
       "use":"enc",
       "kid":"1"},

      {"kty":"RSA",
       "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx
  4cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMs
  tn64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FDW2
  QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbI
  SD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqb
  w0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
       "e":"AQAB",
       "alg":"RS256",
       "kid":"2011-04-29"}
    ]
  }
```

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.)

```
{"keys":
  [
    {"kty":"EC",
     "crv":"P-256",
     "x":"MKBCTNIcKUSDii11ySs3526iDZ8AiTo7Tu6KPAqv7D4",
     "y":"4Etl6SRW2YiLUrN5vfvVHuhp7x8PxltmWWlbbM4IFyM",
     "d":"870MB6gfuTJ4HtUnUvYMyJpr5eUZNP4Bk43bVdj3eAE",
     "use":"enc",
     "kid":"1"},

    {"kty":"RSA",
     "n":"0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4
cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMst
n64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FDW2Q
vzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbIS
D08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqbw
0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
     "e":"AQAB",
     "d":"X4cTteJY_gn4FYPsXB8rdXix5vwsg1FLN5E3EaG6RJoVH-HLLKD9
M7dx5oo7GURknchnrRweUkC7hT5fJLM0WbFAKNLWY2vv7B6NqXSzUvxT0_YSfqij
wp3RTzlBaCxWp4doFk5N2o8Gy_nHNKroADIkJ46pRUohsXywbReAdYaMwFs9tv8d
_cPVY3i07a3t8MN6TNwm0dSawm9v47UiCl3Sk5ZiG7xojPLu4sbg1U2jx4IBTNBz
nbJSzFHK66jT8bgkuqsk0GjskDJk19Z4qwjwbsnn4j2WBii3RL-Us2lGVkY8fkFz
me1z0HbIkfz0Y6mqnOYtqc0X4jfcKoAC8Q",
     "p":"83i-7IvMGXoMXCskv73TKr8637FiO7Z27zv8oj6pbWUQyLPQBQxtPV
nwD20R-60eTDmD2ujnMt5PoqMrm8RfmNhVWDtjjMmCMjOpSXicFHj7XOuVIYQyqV
WlWEh6dN36GVZYk93N8Bc9vY41xy8B9RzzOGVQzXvNEvn7O0nVbfs",
     "q":"3dfOR9cuYq-0S-mkFLzgItgMEfFzB2q3hWehMuG0oCuqnb3vobLyum
qjVZQO1dIrdwgTnCdpYzBcOfW5r370AFXjiWft_NGEiovonizhKpo9VVS78TzFgx
kIdrecRezsZ-1kYd_s1qDbxtkDEgfAITAG9LUnADun4vIcb6yelxk",
     "dp":"G4sPXkc6Ya9y8oJW9_ILj4xuppu0lzi_H7VTkS8xj5SdX3coE0oim
YwxIi2emTAue0UOa5dpgFGyBJ4c8tQ2VF402XRugKDTP8akYhFo5tAA77Qe_Nmtu
YZc3C3m3I24G2GvR5sSDxUyAN2zq8Lfn9EUms6rY3Ob8YeiKkTiBj0",
     "dq":"s9lAH9fggBsoFR8Oac2R_E2gw282rT2kGOAhvIllETE1efrA6huUU
vMfBcMpn8lqeW6vzznYY5SSQF7pMdC_agI3nG8Ibp1BUb0JUiraRNqUfLhcQb_d9
GF4Dh7e74WbRsobRonujTYN1xCaP6TO61jvWrX-L18txXw494Q_cgk",
     "qi":"GyM_p6JrXySiz1toFgKbWV-JdI3jQ4ypu9rbMWx3rQJBfmt0FoYzg
UIZEVFEcOqwemRN81zoDAaa-Bk0KWNGDjJHZDdDmFhW3AN7lI-puxk_mHZGJ11rx
yR8O55XLSe3SPmRfKwZI6yU24ZxvQKFYItdldUKGzO6Ia6zTKhAVRU",
     "alg":"RS256",
     "kid":"2011-04-29"}
  ]
}
```

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.)

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

      {"kty":"oct",
       "k":"AyM1SysPpbyDfgZld3umj1qzKObwVMkoqQ-EstJQLr_T-1qS0gZH75
  aKtMN3Yj0iPS4hcgUuTwjAzZr1Z9CAow",
       "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":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgAsz2J_pqYW08
   PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1uCLNwBuUiCO11_-7dYbsr4iJmG0Q
   u2j8DsVyT1azpJC_NG84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4a
   YWAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKvj-nUy1wgzjYQDwH
   MTplCoLtU-o-8SNnZ1tmRoGE9uJkBLdh5gFENabWnU5m1ZqZPdwS-qo-meMv
   VfJb6jJVWRpl2SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
   "e":"AQAB",
   "x5c":
    ["MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEBBQUAMGIxCzAJB
    gNVBAYTAlVTMQswCQYDVQQIEwJDTzEPMA0GA1UEBxMGRGVudmVyMRwwGgYD
    VQQKExNQaW5nIElkZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1
    wYmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5MTVaMGIxCzAJBg
    NVBAYTAlVTMQswCQYDVQQIEwJDTzEPMA0GA1UEBxMGRGVudmVyMRwwGgYDV
    QQKExNQaW5nIElkZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
    YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL64zn8/QnH
    YMeZ0LncoXaEde1fiLm1jHjmQsF/449IYALM9if6amFtPDy2yvz3YlRij66
    s5gyLCyO7ANuVRJx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws6
    SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU1quGmFgHIXPLfnpn
    fajr1rVTAwtgV5LEZ4Iel+W1GC8ugMhyr4/p1MtcIM42EA8BzE6ZQqC7VPq
    PvEjZ2dbZkaBhPbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyVVk
    aZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEAATANBgkqhkiG9w0BA
    QUFAAOCAQEAh8zGlfSlcI0o3rYDPBB07aXNswb4ECNIKG0CETTUxmXl9KUL
    +9gGlqCz5iWLOgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5CpOe1
    zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCyybEpOGVwe8fnk+fbEL
    2Bo3UPGrpsHzUoaGpDftmWssZkhpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo
    4tpzd5rFXhjRbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkumGmTq
    gawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA6SdS4xSvdXK3IVfOWA=="]
  }
```


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 authenticated encryption operation, formatted as a JWK object:


```
  {
   "kty":"RSA",
   "kid":"juliet@capulet.lit",
   "use":"enc",
   "n":"t6Q8PWSi1dkJj9hTP8hNYFlvadM7DflW9mWepOJhJ66w7nyoK1gPNqFMSQRy
        O125Gp-TEkodhWr0iujjHVx7BcV0llS4w5ACGgPrcAd6ZcSR0-Iqom-QFcNP
        8Sjg086MwoqQU_LYywlAGZ21WSdS_PERyGFiNnj3QQlO8Yns5jCtLCRwLHL0
        Pb1fEv45AuRIuUfVcPySBWYnDyGxvjYGDSM-AqWS9zIQ2ZilgT-GqUmipg0X
        OC0Cc20rgLe2ymLHjpHciCKVAbY5-L32-lSeZO-Os6U15_aXrk9Gw8cPUaX1
        _I8sLGuSiVdt3C_Fn2PZ3Z8i744FPFGGcG1qs2Wz-Q",
   "e":"AQAB",
   "d":"GRtbIQmhOZtyszfgKdg4u_N-R_mZGU_9k7JQ_jn1DnfTuMdSNprTeaSTyWfS
        NkuaAwnOEbIQVy1IQbWVV25NY3ybc_IhUJtfri7bAXYEReWaCl3hdlPKXy9U
        vqPYGR0kIXTQRqns-dVJ7jahlI7LyckrpTmrM8dWBo4_PMaenNnPiQgO0xnu
        ToxutRZJfJvG4Ox4ka3GORQd9CsCZ2vsUDmsXOfUENOyMqADC6p1M3h33tsu
        rY15k9qMSpG9OX_IJAXmxzAh_tWiZOwk2K4yxH9tS3Lq1yX8C1EWmeRDkK2a
        hecG85-oLKQt5VEpWHKmjOi_gJSdSgqcN96X52esAQ",
   "p":"2rnSOV4hKSN8sS4CgcQHFbs08XboFDqKum3sc4h3GRxrTmQdl1ZK9uw-PIHf
        QP0FkxXVrx-WE-ZEbrqivH_2iCLUS7wAl6XvARt1KkIaUxPPSYB9yk31s0Q8
        UK96E3_OrADAYtAJs-M3JxCLfNgqh56HDnETTQhH3rCT5T3yJws",
   "q":"1u_RiFDP7LBYh3N4GXLT9OpSKYP0uQZyiaZwBtOCBNJgQxaj10RWjsZu0c6I
        edis4S7B_coSKB0Kj9PaPaBzg-IySRvvcQuPamQu66riMhjVtG6TlV8CLCYK
        rYl52ziqK0E_ym2QnkwsUX7eYTB7LbAHRK9GqocDE5B0f808I4s",
   "dp":"KkMTWqBUefVwZ2_Dbj1pPQqyHSHjj90L5x_MOzqYAJMcLMZtbUtwKqvVDq3
        tbEo3ZIcohbDtt6SbfmWzggabpQxNxuBpoOOf_a_HgMXK_lhqigI4y_kqS1w
        Y52IwjUn5rgRrJ-yYo1h41KR-vz2pYhEAeYrhttWtxVqLCRViD6c",
   "dq":"AvfS0-gRxvn0bwJoMSnFxYcK1WnuEjQFluMGfwGitQBWtfZ1Er7t1xDkbN9
        GQTB9yqpDoYaN06H7CFtrkxhJIBQaj6nkF5KKS3TQtQ5qCzkOkmxIe3KRbBy
        mXxkb5qwUpX5ELD5xFc6FeiafWYY63TmmEAu_lRFCOJ3xDea-ots",
   "qi":"lSQi-w9CpyUReMErP1RsBLk7wNtOvs5EQpPqmuMvqW57NBUczScEoPwmUqq
        abu9V0-Py4dQ57_bapoKRu1R90bvuFnU63SHWEFglZQvJDMeAvmj4sm-Fp0o
        Yu_neotgQ0hzbI5gry7ajdYy9-2lNx_76aBZoOUu9HCJ-UsfSOI8"
  }
```


The octets representing the Plaintext used in this example (using JSON array notation) are:


```
[123, 34, 107, 116, 121, 34, 58, 34, 82, 83, 65, 34, 44, 34, 107,
105, 100, 34, 58, 34, 106, 117, 108, 105, 101, 116, 64, 99, 97, 112,
117, 108, 101, 116, 46, 108, 105, 116, 34, 44, 34, 117, 115, 101, 34,
58, 34, 101, 110, 99, 34, 44, 34, 110, 34, 58, 34, 116, 54, 81, 56,
80, 87, 83, 105, 49, 100, 107, 74, 106, 57, 104, 84, 80, 56, 104, 78,
89, 70, 108, 118, 97, 100, 77, 55, 68, 102, 108, 87, 57, 109, 87,
```

101, 112, 79, 74, 104, 74, 54, 54, 119, 55, 110, 121, 111, 75, 49,
103, 80, 78, 113, 70, 77, 83, 81, 82, 121, 79, 49, 50, 53, 71, 112,
45, 84, 69, 107, 111, 100, 104, 87, 114, 48, 105, 117, 106, 106, 72,
86, 120, 55, 66, 99, 86, 48, 108, 108, 83, 52, 119, 53, 65, 67, 71,
103, 80, 114, 99, 65, 100, 54, 90, 99, 83, 82, 48, 45, 73, 113, 111,
109, 45, 81, 70, 99, 78, 80, 56, 83, 106, 103, 48, 56, 54, 77, 119,
111, 113, 81, 85, 95, 76, 89, 121, 119, 108, 65, 71, 90, 50, 49, 87,
83, 100, 83, 95, 80, 69, 82, 121, 71, 70, 105, 78, 110, 106, 51, 81,
81, 108, 79, 56, 89, 110, 115, 53, 106, 67, 116, 76, 67, 82, 119, 76,
72, 76, 48, 80, 98, 49, 102, 69, 118, 52, 53, 65, 117, 82, 73, 117,
85, 102, 86, 99, 80, 121, 83, 66, 87, 89, 110, 68, 121, 71, 120, 118,
106, 89, 71, 68, 83, 77, 45, 65, 113, 87, 83, 57, 122, 73, 81, 50,
90, 105, 108, 103, 84, 45, 71, 113, 85, 109, 105, 112, 103, 48, 88,
79, 67, 48, 67, 99, 50, 48, 114, 103, 76, 101, 50, 121, 109, 76, 72,
106, 112, 72, 99, 105, 67, 75, 86, 65, 98, 89, 53, 45, 76, 51, 50,
45, 108, 83, 101, 90, 79, 45, 79, 115, 54, 85, 49, 53, 95, 97, 88,
114, 107, 57, 71, 119, 56, 99, 80, 85, 97, 88, 49, 95, 73, 56, 115,
76, 71, 117, 83, 105, 86, 100, 116, 51, 67, 95, 70, 110, 50, 80, 90,
51, 90, 56, 105, 55, 52, 52, 70, 80, 70, 71, 71, 99, 71, 49, 113,
115, 50, 87, 122, 45, 81, 34, 44, 34, 101, 34, 58, 34, 65, 81, 65,
66, 34, 44, 34, 100, 34, 58, 34, 71, 82, 116, 98, 73, 81, 109, 104,
79, 90, 116, 121, 115, 122, 102, 103, 75, 100, 103, 52, 117, 95, 78,
45, 82, 95, 109, 90, 71, 85, 95, 57, 107, 55, 74, 81, 95, 106, 110,
49, 68, 110, 102, 84, 117, 77, 100, 83, 78, 112, 114, 84, 101, 97,
83, 84, 121, 87, 102, 83, 78, 107, 117, 97, 65, 119, 110, 79, 69, 98,
73, 81, 86, 121, 49, 73, 81, 98, 87, 86, 86, 50, 53, 78, 89, 51, 121,
98, 99, 95, 73, 104, 85, 74, 116, 102, 114, 105, 55, 98, 65, 88, 89,
69, 82, 101, 87, 97, 67, 108, 51, 104, 100, 108, 80, 75, 88, 121, 57,
85, 118, 113, 80, 89, 71, 82, 48, 107, 73, 88, 84, 81, 82, 113, 110,
115, 45, 100, 86, 74, 55, 106, 97, 104, 108, 73, 55, 76, 121, 99,
107, 114, 112, 84, 109, 114, 77, 56, 100, 87, 66, 111, 52, 95, 80,
77, 97, 101, 110, 78, 110, 80, 105, 81, 103, 79, 48, 120, 110, 117,
84, 111, 120, 117, 116, 82, 90, 74, 102, 74, 118, 71, 52, 79, 120,
52, 107, 97, 51, 71, 79, 82, 81, 100, 57, 67, 115, 67, 90, 50, 118,
115, 85, 68, 109, 115, 88, 79, 102, 85, 69, 78, 79, 121, 77, 113, 65,
68, 67, 54, 112, 49, 77, 51, 104, 51, 51, 116, 115, 117, 114, 89, 49,
53, 107, 57, 113, 77, 83, 112, 71, 57, 79, 88, 95, 73, 74, 65, 88,
109, 120, 122, 65, 104, 95, 116, 87, 105, 90, 79, 119, 107, 50, 75,
52, 121, 120, 72, 57, 116, 83, 51, 76, 113, 49, 121, 88, 56, 67, 49,
69, 87, 109, 101, 82, 68, 107, 75, 50, 97, 104, 101, 99, 71, 56, 53,
45, 111, 76, 75, 81, 116, 53, 86, 69, 112, 87, 72, 75, 109, 106, 79,
105, 95, 103, 74, 83, 100, 83, 103, 113, 99, 78, 57, 54, 88, 53, 50,
101, 115, 65, 81, 34, 44, 34, 112, 34, 58, 34, 50, 114, 110, 83, 79,
86, 52, 104, 75, 83, 78, 56, 115, 83, 52, 67, 103, 99, 81, 72, 70,
98, 115, 48, 56, 88, 98, 111, 70, 68, 113, 75, 117, 109, 51, 115, 99,
52, 104, 51, 71, 82, 120, 114, 84, 109, 81, 100, 108, 49, 90, 75, 57,
117, 119, 45, 80, 73, 72, 102, 81, 80, 48, 70, 107, 120, 88, 86, 114,
120, 45, 87, 69, 45, 90, 69, 98, 114, 113, 105, 118, 72, 95, 50, 105,

67, 76, 85, 83, 55, 119, 65, 108, 54, 88, 118, 65, 82, 116, 49, 75,
107, 73, 97, 85, 120, 80, 80, 83, 89, 66, 57, 121, 107, 51, 49, 115,
48, 81, 56, 85, 75, 57, 54, 69, 51, 95, 79, 114, 65, 68, 65, 89, 116,
65, 74, 115, 45, 77, 51, 74, 120, 67, 76, 102, 78, 103, 113, 104, 53,
54, 72, 68, 110, 69, 84, 84, 81, 104, 72, 51, 114, 67, 84, 53, 84,
51, 121, 74, 119, 115, 34, 44, 34, 113, 34, 58, 34, 49, 117, 95, 82,
105, 70, 68, 80, 55, 76, 66, 89, 104, 51, 78, 52, 71, 88, 76, 84, 57,
79, 112, 83, 75, 89, 80, 48, 117, 81, 90, 121, 105, 97, 90, 119, 66,
116, 79, 67, 66, 78, 74, 103, 81, 120, 97, 106, 49, 48, 82, 87, 106,
115, 90, 117, 48, 99, 54, 73, 101, 100, 105, 115, 52, 83, 55, 66, 95,
99, 111, 83, 75, 66, 48, 75, 106, 57, 80, 97, 80, 97, 66, 122, 103,
45, 73, 121, 83, 82, 118, 118, 99, 81, 117, 80, 97, 109, 81, 117, 54,
54, 114, 105, 77, 104, 106, 86, 116, 71, 54, 84, 108, 86, 56, 67, 76,
67, 89, 75, 114, 89, 108, 53, 50, 122, 105, 113, 75, 48, 69, 95, 121,
109, 50, 81, 110, 107, 119, 115, 85, 88, 55, 101, 89, 84, 66, 55, 76,
98, 65, 72, 82, 75, 57, 71, 113, 111, 99, 68, 69, 53, 66, 48, 102,
56, 48, 56, 73, 52, 115, 34, 44, 34, 100, 112, 34, 58, 34, 75, 107,
77, 84, 87, 113, 66, 85, 101, 102, 86, 119, 90, 50, 95, 68, 98, 106,
49, 112, 80, 81, 113, 121, 72, 83, 72, 106, 106, 57, 48, 76, 53, 120,
95, 77, 79, 122, 113, 89, 65, 74, 77, 99, 76, 77, 90, 116, 98, 85,
116, 119, 75, 113, 118, 86, 68, 113, 51, 116, 98, 69, 111, 51, 90,
73, 99, 111, 104, 98, 68, 116, 116, 54, 83, 98, 102, 109, 87, 122,
103, 103, 97, 98, 112, 81, 120, 78, 120, 117, 66, 112, 111, 79, 79,
102, 95, 97, 95, 72, 103, 77, 88, 75, 95, 108, 104, 113, 105, 103,
73, 52, 121, 95, 107, 113, 83, 49, 119, 89, 53, 50, 73, 119, 106, 85,
110, 53, 114, 103, 82, 114, 74, 45, 121, 89, 111, 49, 104, 52, 49,
75, 82, 45, 118, 122, 50, 112, 89, 104, 69, 65, 101, 89, 114, 104,
116, 116, 87, 116, 120, 86, 113, 76, 67, 82, 86, 105, 68, 54, 99, 34,
44, 34, 100, 113, 34, 58, 34, 65, 118, 102, 83, 48, 45, 103, 82, 120,
118, 110, 48, 98, 119, 74, 111, 77, 83, 110, 70, 120, 89, 99, 75, 49,
87, 110, 117, 69, 106, 81, 70, 108, 117, 77, 71, 102, 119, 71, 105,
116, 81, 66, 87, 116, 102, 90, 49, 69, 114, 55, 116, 49, 120, 68,
107, 98, 78, 57, 71, 81, 84, 66, 57, 121, 113, 112, 68, 111, 89, 97,
78, 48, 54, 72, 55, 67, 70, 116, 114, 107, 120, 104, 74, 73, 66, 81,
97, 106, 54, 110, 107, 70, 53, 75, 75, 83, 51, 84, 81, 116, 81, 53,
113, 67, 122, 107, 79, 107, 109, 120, 73, 101, 51, 75, 82, 98, 66,
121, 109, 88, 120, 107, 98, 53, 113, 119, 85, 112, 88, 53, 69, 76,
68, 53, 120, 70, 99, 54, 70, 101, 105, 97, 102, 87, 89, 89, 54, 51,
84, 109, 109, 69, 65, 117, 95, 108, 82, 70, 67, 79, 74, 51, 120, 68,
101, 97, 45, 111, 116, 115, 34, 44, 34, 113, 105, 34, 58, 34, 108,
83, 81, 105, 45, 119, 57, 67, 112, 121, 85, 82, 101, 77, 69, 114, 80,
49, 82, 115, 66, 76, 107, 55, 119, 78, 116, 79, 118, 115, 53, 69, 81,
112, 80, 113, 109, 117, 77, 118, 113, 87, 53, 55, 78, 66, 85, 99,
122, 83, 99, 69, 111, 80, 119, 109, 85, 113, 113, 97, 98, 117, 57,
86, 48, 45, 80, 121, 52, 100, 81, 53, 55, 95, 98, 97, 112, 111, 75,
82, 117, 49, 82, 57, 48, 98, 118, 117, 70, 110, 85, 54, 51, 83, 72,
87, 69, 70, 103, 108, 90, 81, 118, 74, 68, 77, 101, 65, 118, 109,
106, 52, 115, 109, 45, 70, 112, 48, 111, 89, 117, 95, 110, 101, 111,

116, 103, 81, 48, 104, 122, 98, 73, 53, 103, 114, 121, 55, 97, 106,
100, 89, 121, 57, 45, 50, 108, 78, 120, 95, 55, 54, 97, 66, 90, 111,
79, 85, 117, 57, 72, 67, 74, 45, 85, 115, 102, 83, 79, 73, 56, 34,
125]

C.2. JOSE Header

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 Input ("p2s") value is [217, 96, 147, 112, 150, 117, 70, 247, 127, 8, 155, 137, 174, 42, 80, 215],
- the Iteration Count ("p2c") value is 4096,
- authenticated encryption is performed on the Plaintext using the AES_128_CBC_HMAC_SHA_256 algorithm to produce the Ciphertext and the Authentication Tag, and
- the content type is application/jwk+json.

```
{
 "alg":"PBES2-HS256+A128KW",
 "p2s":"2WCTcJZ1Rvd_CJuJripQ1w",
 "p2c":4096,
 "enc":"A128CBC-HS256",
 "cty":"jwk+json"
}
```

Encoding this JWE Protected Header as BASE64URL(UTF8(JWE Protected Header)) gives this value (with line breaks for display purposes only):

```
eyJhbGciOiJQQkVTMi1IUzI1NitBMTI4S1ciLCJwMnMiOiIyV0NUY0paMVJ2ZF9DSn
VKcmlwUTF3IiwicDJjIjo0MDk2LCJlbmMiOiJBMTI4Q0JDLUhTMjU2IiwiY3R5Ijoi
andrK2pzb24ifQ
```

C.3. Content Encryption Key (CEK)

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

[111, 27, 25, 52, 66, 29, 20, 78, 92, 176, 56, 240, 65, 208, 82, 112,
161, 131, 36, 55, 202, 236, 185, 172, 129, 23, 153, 194, 195, 48,
253, 182]

C.4. Key Derivation

Derive a key from a shared passphrase using the PBKDF2 algorithm with HMAC SHA-256 and the specified Salt and Iteration Count values and a 128 bit requested output key size to produce the PBKDF2 Derived Key. This example uses the following passphrase:

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

The octets representing the passphrase are:

[84, 104, 117, 115, 32, 102, 114, 111, 109, 32, 109, 121, 32, 108, 105, 112, 115, 44, 32, 98, 121, 32, 121, 111, 117, 114, 115, 44, 32, 109, 121, 32, 115, 105, 110, 32, 105, 115, 32, 112, 117, 114, 103, 101, 100, 46]

The Salt value (UTF8(Alg) || 0x00 || Salt Input) is:

[80, 66, 69, 83, 50, 45, 72, 83, 50, 53, 54, 43, 65, 49, 50, 56, 75, 87, 0, 217, 96, 147, 112, 150, 117, 70, 247, 127, 8, 155, 137, 174, 42, 80, 215].

The resulting PBKDF2 Derived Key value is:

[110, 171, 169, 92, 129, 92, 109, 117, 233, 242, 116, 233, 170, 14, 24, 75]

C.5. Key Encryption

Encrypt the CEK with the "A128KW" algorithm using the PBKDF2 Derived Key. The resulting JWE Encrypted Key value is:

[78, 186, 151, 59, 11, 141, 81, 240, 213, 245, 83, 211, 53, 188, 134, 188, 66, 125, 36, 200, 222, 124, 5, 103, 249, 52, 117, 184, 140, 81, 246, 158, 161, 177, 20, 33, 245, 57, 59, 4]

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

TrqXOwuNUfDV9VPTNbyGvEJ9JMjefAVn-TR1uIxR9p6hsRQh9Tk7BA

C.6. 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:

Ye9j1qs22DmRSAddIh-VnA

C.7. 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, 117, 74, 114, 105, 112, 81, 49, 119, 34, 44, 34, 112, 50, 99, 34, 58, 52, 48, 57, 54, 44, 34, 101, 110, 99, 34, 58, 34, 65, 49, 50, 56, 67, 66, 67, 45, 72, 83, 50, 53, 54, 34, 44, 34, 99, 116, 121, 34, 58, 34, 106, 119, 107, 43, 106, 115, 111, 110, 34, 125]

C.8. Content Encryption

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

[3, 8, 65, 242, 92, 107, 148, 168, 197, 159, 77, 139, 25, 97, 42, 131, 110, 199, 225, 56, 61, 127, 38, 64, 108, 91, 247, 167, 150, 98, 112, 122, 99, 235, 132, 50, 28, 46, 56, 170, 169, 89, 220, 145, 38, 157, 148, 224, 66, 140, 8, 169, 146, 117, 222, 54, 242, 28, 31, 11, 129, 227, 226, 169, 66, 117, 133, 254, 140, 216, 115, 203, 131, 60, 60, 47, 233, 132, 121, 13, 35, 188, 53, 19, 172, 77, 59, 54, 211, 158, 172, 25, 60, 111, 0, 80, 201, 158, 160, 210, 68, 55, 12, 67, 136, 130, 87, 216, 197, 95, 62, 20, 155, 205, 5, 140, 27, 168, 221, 65, 114, 78, 157, 254, 46, 206, 182, 52, 135, 87, 239, 3, 34, 186, 126, 220, 151, 17, 33, 237, 57, 96, 172, 183, 58, 45, 248, 103, 241, 142, 136, 7, 53, 16, 173, 181, 7, 93, 92, 252, 1, 53, 212, 242, 8, 255, 11, 239, 181, 24, 148, 136, 111, 24, 161, 244, 23, 106, 69, 157, 215, 243, 189, 240, 166, 169, 249, 72, 38, 201, 99, 223, 173, 229, 9, 222, 82, 79, 157, 176, 248, 85, 239, 121, 163, 1, 31, 48, 98, 206, 61, 249, 104, 216, 201, 227, 105, 48, 194, 193, 10, 36, 160, 159, 241, 166, 84, 54, 188, 211, 243, 242, 40, 46, 45, 193, 193, 160, 169, 101, 201, 1, 73, 47, 105, 142, 88, 28, 42, 132, 26, 61, 58, 63, 142, 243, 77, 26, 179, 153, 166, 46, 203, 208, 49, 55, 229, 34, 178, 4, 109, 180, 204, 204, 115, 1, 103, 193, 5, 91, 215, 214, 195, 1, 110, 208, 53, 144, 36, 105, 12, 54, 25, 129, 101, 15, 183, 150, 250, 147, 115, 227, 58, 250, 5, 128, 232, 63, 15, 14, 19, 141, 124, 253, 142, 137, 189, 135, 26, 44, 240, 27, 88, 132, 105, 127, 6, 71, 37, 41, 124, 187, 165, 140, 34, 200, 123, 80, 228, 24, 231, 176, 132, 171,

138, 145, 152, 116, 224, 50, 141, 51, 147, 91, 186, 7, 246, 106, 217,
148, 244, 227, 244, 45, 220, 121, 165, 224, 148, 181, 17, 181, 128,
197, 101, 237, 11, 169, 229, 149, 199, 78, 56, 15, 14, 190, 91, 216,
222, 247, 213, 74, 40, 8, 96, 20, 168, 119, 96, 26, 24, 52, 37, 82,
127, 57, 176, 147, 118, 59, 7, 224, 33, 117, 72, 155, 29, 82, 26,
215, 189, 140, 119, 28, 152, 118, 93, 222, 194, 192, 148, 115, 83,
253, 216, 212, 108, 88, 83, 175, 172, 220, 97, 79, 110, 42, 223, 170,
161, 34, 164, 144, 193, 76, 122, 92, 160, 41, 178, 175, 6, 35, 96,
113, 96, 158, 90, 129, 101, 26, 45, 70, 180, 189, 230, 15, 5, 247,
150, 209, 94, 171, 26, 13, 142, 212, 129, 1, 176, 5, 0, 112, 203,
174, 185, 119, 76, 233, 189, 54, 172, 189, 245, 223, 253, 205, 12,
88, 9, 126, 157, 225, 90, 40, 229, 191, 63, 30, 160, 224, 69, 3, 140,
109, 70, 89, 37, 213, 245, 194, 210, 180, 188, 63, 210, 139, 221, 2,
144, 200, 20, 177, 216, 29, 227, 242, 106, 12, 135, 142, 139, 144,
82, 225, 162, 171, 176, 108, 99, 6, 43, 193, 161, 116, 234, 216, 1,
242, 21, 124, 162, 98, 205, 124, 193, 38, 12, 242, 90, 101, 76, 204,
184, 124, 58, 180, 16, 240, 26, 76, 195, 250, 212, 191, 185, 191, 97,
198, 186, 73, 225, 75, 14, 90, 123, 121, 172, 101, 50, 160, 221, 141,
253, 205, 126, 77, 9, 87, 198, 110, 104, 182, 141, 120, 51, 25, 232,
3, 32, 80, 6, 156, 8, 18, 4, 135, 221, 142, 25, 135, 2, 129, 132,
115, 227, 74, 141, 28, 119, 11, 141, 117, 134, 198, 62, 150, 254, 97,
75, 197, 251, 99, 89, 204, 224, 226, 67, 83, 175, 89, 0, 81, 29, 38,
207, 89, 140, 255, 197, 177, 164, 128, 62, 116, 224, 180, 109, 169,
28, 2, 59, 176, 130, 252, 44, 178, 81, 24, 181, 176, 75, 44, 61, 91,
12, 37, 21, 255, 83, 130, 197, 16, 231, 60, 217, 56, 131, 118, 168,
202, 58, 52, 84, 124, 162, 185, 174, 162, 226, 242, 112, 68, 246,
202, 16, 208, 52, 154, 58, 129, 80, 102, 33, 171, 6, 186, 177, 14,
195, 88, 136, 6, 0, 155, 28, 100, 162, 207, 162, 222, 117, 248, 170,
208, 114, 87, 31, 57, 176, 33, 57, 83, 253, 12, 168, 110, 194, 59,
22, 86, 48, 227, 196, 22, 176, 218, 122, 149, 21, 249, 195, 178, 174,
250, 20, 34, 120, 60, 139, 201, 99, 40, 18, 177, 17, 54, 54, 6, 3,
222, 128, 160, 88, 11, 27, 0, 81, 192, 36, 41, 169, 146, 8, 47, 64,
136, 28, 64, 209, 67, 135, 202, 20, 234, 182, 91, 204, 146, 195, 187,
0, 72, 77, 11, 111, 152, 204, 252, 177, 212, 89, 33, 50, 132, 184,
44, 183, 186, 19, 250, 69, 176, 201, 102, 140, 14, 143, 212, 212,
160, 123, 208, 185, 27, 155, 68, 77, 133, 198, 2, 126, 155, 215, 22,
91, 30, 217, 176, 172, 244, 156, 174, 143, 75, 90, 21, 102, 1, 160,
59, 253, 188, 88, 57, 185, 197, 83, 24, 22, 180, 174, 47, 207, 52, 1,
141, 146, 119, 233, 68, 228, 224, 228, 193, 248, 155, 202, 90, 7,
213, 88, 33, 108, 107, 14, 86, 8, 120, 250, 58, 142, 35, 164, 238,
221, 219, 35, 123, 88, 199, 192, 143, 104, 83, 17, 166, 243, 247, 11,
166, 67, 68, 204, 132, 23, 110, 103, 228, 14, 55, 122, 88, 57, 180,
178, 237, 52, 130, 214, 245, 102, 123, 67, 73, 175, 1, 127, 112, 148,
94, 132, 164, 197, 153, 217, 87, 25, 89, 93, 63, 22, 66, 166, 90,
251, 101, 10, 145, 66, 17, 124, 36, 255, 165, 226, 97, 16, 86, 112,
154, 88, 105, 253, 56, 209, 229, 122, 103, 51, 24, 228, 190, 3, 236,
48, 182, 121, 176, 140, 128, 117, 87, 251, 224, 37, 23, 248, 21, 218,
85, 251, 136, 84, 147, 143, 144, 46, 155, 183, 251, 89, 86, 23, 26,

237, 100, 167, 32, 130, 173, 237, 89, 55, 110, 70, 142, 127, 65, 230,
208, 109, 69, 19, 253, 84, 130, 130, 193, 92, 58, 108, 150, 42, 136,
249, 234, 86, 241, 182, 19, 117, 246, 26, 181, 92, 101, 155, 44, 103,
235, 173, 30, 140, 90, 29, 183, 190, 77, 53, 206, 127, 5, 87, 8, 187,
184, 92, 4, 157, 22, 18, 105, 251, 39, 88, 182, 181, 103, 148, 233,
6, 63, 70, 188, 7, 101, 216, 127, 77, 31, 12, 233, 7, 147, 106, 30,
150, 77, 145, 13, 205, 48, 56, 245, 220, 89, 252, 127, 51, 180, 36,
31, 55, 18, 214, 230, 254, 217, 197, 65, 247, 27, 215, 117, 247, 108,
157, 121, 11, 63, 150, 195, 83, 6, 134, 242, 41, 24, 105, 204, 5, 63,
192, 14, 159, 113, 72, 140, 128, 51, 215, 80, 215, 39, 149, 94, 79,
128, 34, 5, 129, 82, 83, 121, 187, 37, 146, 27, 32, 177, 167, 71, 9,
195, 30, 199, 196, 205, 252, 207, 69, 8, 120, 27, 190, 51, 43, 75,
249, 234, 167, 116, 206, 203, 199, 43, 108, 87, 48, 155, 140, 228,
210, 85, 25, 161, 96, 67, 8, 205, 64, 39, 75, 88, 44, 238, 227, 16,
0, 100, 93, 129, 18, 4, 149, 50, 68, 72, 99, 35, 111, 254, 27, 102,
175, 108, 233, 87, 181, 44, 169, 18, 139, 79, 208, 14, 202, 192, 5,
162, 222, 231, 149, 24, 211, 49, 120, 101, 39, 206, 87, 147, 204,
200, 251, 104, 115, 5, 127, 117, 195, 79, 151, 18, 224, 52, 0, 245,
4, 85, 255, 103, 217, 0, 116, 198, 80, 91, 167, 192, 154, 199, 197,
149, 237, 51, 2, 131, 30, 226, 95, 105, 48, 68, 135, 208, 144, 120,
176, 145, 157, 8, 171, 80, 94, 61, 92, 92, 220, 157, 13, 138, 51, 23,
185, 124, 31, 77, 1, 87, 241, 43, 239, 55, 122, 86, 210, 48, 208,
204, 112, 144, 80, 147, 106, 219, 47, 253, 31, 134, 176, 16, 135,
219, 95, 17, 129, 83, 236, 125, 136, 112, 86, 228, 252, 71, 129, 218,
174, 156, 236, 12, 27, 159, 11, 138, 252, 253, 207, 31, 115, 214,
118, 239, 203, 16, 211, 205, 99, 22, 51, 163, 107, 162, 246, 199, 67,
127, 34, 108, 197, 53, 117, 58, 199, 3, 190, 74, 70, 190, 65, 235,
175, 97, 157, 215, 252, 189, 245, 100, 229, 248, 46, 90, 126, 237, 4,
159, 128, 58, 7, 156, 236, 69, 191, 85, 240, 179, 224, 249, 152, 49,
195, 223, 60, 78, 186, 157, 155, 217, 58, 105, 116, 164, 217, 111,
215, 150, 218, 252, 84, 86, 248, 140, 240, 226, 61, 106, 208, 95, 60,
163, 6, 0, 235, 253, 162, 96, 62, 234, 251, 249, 35, 21, 7, 211, 233,
86, 50, 33, 203, 67, 248, 60, 190, 123, 48, 167, 226, 90, 191, 71,
56, 183, 165, 17, 85, 76, 238, 140, 211, 168, 53, 223, 194, 4, 97,
149, 156, 120, 137, 76, 33, 229, 243, 194, 208, 198, 202, 139, 28,
114, 46, 224, 92, 254, 83, 100, 134, 158, 92, 70, 78, 61, 62, 138,
24, 173, 216, 66, 198, 70, 254, 47, 59, 193, 53, 6, 139, 19, 153,
253, 28, 199, 122, 160, 27, 67, 234, 209, 227, 139, 4, 50, 7, 178,
183, 89, 252, 32, 128, 137, 55, 52, 29, 89, 12, 111, 42, 181, 51,
170, 132, 132, 207, 170, 228, 254, 178, 213, 0, 136, 175, 8]

The resulting Authentication Tag value is:

[208, 113, 102, 132, 236, 236, 67, 223, 39, 53, 98, 99, 32, 121, 17,
236]

Encoding this JWE Ciphertext as BASE64URL(JWE Ciphertext) gives this
value (with line breaks for display purposes only):

```
AwhB8lxrlKjFn02LGWEqg27H4Tg9fyZAbFv3p5ZicHpj64QyHC44qqlZ3JEmnZTgQo
wIqZJ13jbyHB8LgePiqUJ1hf6M2HPLgzw8L-mEeQ0jvDUTrE07NtOerBk8bwBQyZ6g
0kQ3DEOIglfYxV8-FJvNBYwbqN1Bck6d_i7OtjSHV-8DIrp-3JcRIe05YKy3Oi34Z_
GOiAc1EK21B11c_AE11PII_wvvtRiUiG8YofQXakWd1_O98Kap-UgmyWPfreUJ3lJP
nbD4Ve95owEfMGLOPflo2MnjaTDCwQokoJ_xplQ2vNPz8iguLcHBoKllyQFJL2mOWB
wqhBo9Oj-O800as5mmLsvQMTflIrIEbbTMzHMBZ8EFW9fWwwFu0DWQJGkMNhmBZQ-3
lvqTc-M6-gWA6D8PDhONfP2Oib2HGizwG1iEaX8GRyUpfLuljCLIe1DkGOewhKuKkZ
h04DKNM5Nbugf2atmU9OP0Ldx5peCUtRG1gMVl7Qup5ZXHTjgPDr5b2N731UooCGAU
qHdgGhg0JVJ_ObCTdjsH4CF1SJsdUhrXvYx3HJh2Xd7CwJRzU_3Y1GxYU6-s3GFPbi
rfqqEipJDBTHpcoCmyrwYjYHFgnlqBZRotRrS95g8F95bRXqsaDY7UgQGwBQBwy665
d0zpvTasvfXf_c0MWAl-neFaKOW_Px6g4EUDjG1GWSXV9cLStLw_0ovdApDIFLHYHe
PyagyHjouQUuGiq7BsYwYrwaF06tgB8hV8omLNfMEmDPJaZUzMuHw6tBDwGkzD-tS_
ub9hxrpJ4UsOWnt5rGUyoN2N_c1-TQlXxm5oto14MxnoAyBQBpwIEgSH3Y4ZhwKBhH
PjSo0cdwuNdYbGPpb-YUvF-2NZzODiQ1OvWQBRHSbPWYz_xbGkgD504LRtqRwCO7CC
_CyyURi1sEssPVsMJRX_U4LFEOc82TiDdqjKOjRUfKK5rqLi8nBE9soQ0DSaOoFQZi
GrBrqxDsNYiAYAmxxkos-i3nX4qtByVx85sCE5U_0MqG7COxZWMOPEFrDaepUV-cOy
rvoUIng8i8ljKBKxETY2BgPegKBYCxsAUcAkKamSCC9AiBxA0UOHyhTqtlvMksO7AE
hNC2-YzPyx1FkhMoS4LLe6E_pFsMlmjA6P1NSge9C5G5tETYXGAn6b1xZbHtmwrPSc
ro9LWhVmAaA7_bxYObnFUxgWtK4vzzQBjZJ36UTk4OTB-JvKWgfVWCFsaw5WCHj6Oo
4jpO7d2yN7WMfAj2hTEabz9wumQ0TMhBduZ-QON3pYObSy7TSC1vVme0NJrwF_cJRe
hKTFmdlXGVldPxZCplr7ZQqRQhF8JP-l4mEQVnCaWGn9ONHlemczGOS-A-wwtnmwjI
B1V_vgJRf4FdpV-4hUk4-QLpu3-1lWFxrtZKcggq3tWTduRo5_QebQbUUT_VSCgsFc
OmyWKoj56lbxthN19hq1XGWbLGfrrR6MWh23vk01zn8FVwi7uFwEnRYSafsnWLa1Z5
TpBj9GvAdl2H9NHwzpB5NqHpZNkQ3NMDj13Fn8fzO0JB83Etbm_tnFQfcb13X3bJ15
Cz-Ww1MGhvIpGGnMBT_ADp9xSIyAM9dQ1yeVXk-AIgWBUlN5uyWSGyCxp0cJwx7HxM
38z0UIeBu-MytL-eqndM7LxytsVzCbjOTSVRmhYEMIzUAnS1gs7uMQAGRdgRIElTJE
SGMjb_4bZq9s6Ve1LKkSi0_QDsrABaLe55UY0zF4ZSfOV5PMyPtocwV_dcNPlxLgNA
D1BFX_Z9kAdMZQW6fAmsfFle0zAoMe4l9pMESH0JB4sJGdCKtQXj1cXNydDYozF7l8
H00BV_Er7zd6VtIw0MxwkFCTatsv_R-GsBCH218RgVPsfYhwVuT8R4HarpzsDBufC4
r8_c8fc9Z278sQ081jFjOja6L2x0N_ImzFNXU6xwO-Ska-QeuvYZ3X_L31ZOX4Llp-
7QSfgDoHnOxFv1Xws-D5mDHD3zxOup2b2TppdKTZb9eW2vxUVviM8OI9atBfPKMGAO
v9omA-6vv5IxUH0-lWMiHLQ_g8vnswp-Jav0c4t6URVUzujNOoNd_CBGGVnHiJTCHl
88LQxsqLHHIu4Fz-U2SGnlxGTj0-ihit2ELGRv4vO8E1BosTmf0cx3qgG0Pq0eOLBD
IHsrdZ_CCAiTc0HVkMbyq1M6qEhM-q5P6y1QCIrwg
```

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

```
0HFmhOzsQ98nNWJjIHkR7A
```

C.9. Complete Representation

Assemble the final representation: The JWE Compact Serialization of this result, as defined in Section 7.1 of [JWE], 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:

eyJhbGciOiJQQkVTMi1IUzI1NitBMTI4S1ciLCJwMnMiOiIyV0NUY0paMVJ2ZF9DSn
VKcmlwUTF3IiwicDJjIjo0MDk2LCJlbmMiOiJBMTI4Q0JDLUhTMjU2IiwiY3R5Ijoi
andrK2pzb24ifQ.
TrqXOwuNUfDV9VPTNbyGvEJ9JMjefAVn-TR1uIxR9p6hsRQh9Tk7BA.
Ye9j1qs22DmRSAddIh-VnA.
AwhB8lxrlKjFn02LGWEqg27H4Tg9fyZAbFv3p5ZicHpj64QyHC44qqlZ3JEmnZTgQo
wIqZJ13jbyHB8LgePiqUJ1hf6M2HPLgzw8L-mEeQ0jvDUTrE07NtOerBk8bwBQyZ6g
0kQ3DEOIglfYxV8-FJvNBYwbqN1Bck6d_i7OtjSHV-8DIrp-3JcRIe05YKy3Oi34Z_
GOiAc1EK21B11c_AE11PII_wvvtRiUiG8YofQXakWd1_O98Kap-UgmyWPfreUJ3lJP
nbD4Ve95owEfMGLOPflo2MnjaTDCwQokoJ_xplQ2vNPz8iguLcHBoKllyQFJL2mOWB
wqhBo9Oj-O800as5mmLsvQMTflIrIEbbTMzHMBZ8EFW9fWwwFu0DWQJGkMNhmBZQ-3
lvqTc-M6-gWA6D8PDhONfP2Oib2HGizwG1iEaX8GRyUpfLuljCLIe1DkGOewhKuKkZ
h04DKNM5Nbugf2atmU9OP0Ldx5peCUtRG1gMVl7Qup5ZXHTjgPDr5b2N731UooCGAU
qHdgGhg0JVJ_ObCTdjsH4CF1SJsdUhrXvYx3HJh2Xd7CwJRzU_3Y1GxYU6-s3GFPbi
rfqqEipJDBTHpcoCmyrwYjYHFgnlqBZRotRrS95g8F95bRXqsaDY7UgQGwBQBwy665
d0zpvTasvfXf_c0MWAl-neFaKOW_Px6g4EUDjG1GWSXV9cLStLw_0ovdApDIFLHYHe
PyagyHjouQUuGiq7BsYwYrwaF06tgB8hV8omLNfMEmDPJaZUzMuHw6tBDwGkzD-tS_
ub9hxrpJ4UsOWnt5rGUyoN2N_c1-TQlXxm5oto14MxnoAyBQBpwIEgSH3Y4ZhwKBhH
PjSo0cdwuNdYbGPpb-YUvF-2NZzODiQ1OvWQBRHSbPWYz_xbGkgD504LRtqRwCO7CC
_CyyURi1sEssPVsMJRX_U4LFEOc82TiDdqjKOjRUfKK5rqLi8nBE9soQ0DSaOoFQZi
GrBrqxDsNYiAYAmxxkos-i3nX4qtByVx85sCE5U_0MqG7COxZWMOPEFrDaepUV-cOy
rvoUIng8i8ljKBKxETY2BgPegKBYCxsAUcAkKamSCC9AiBxA0UOHyhTqtlvMksO7AE
hNC2-YzPyx1FkhMoS4LLe6E_pFsMlmjA6P1NSge9C5G5tETYXGAn6b1xZbHtmwrPSc
ro9LWhVmAaA7_bxYObnFUxgWtK4vzzQBjZJ36UTk4OTB-JvKWgfVWCFsaw5WCHj6Oo
4jpO7d2yN7WMfAj2hTEabz9wumQ0TMhBduZ-QON3pYObSy7TSC1vVme0NJrwF_cJRe
hKTFmdlXGVldPxZCplr7ZQqRQhF8JP-l4mEQVnCaWGn9ONHlemczGOS-A-wwtnmwjI
B1V_vgJRf4FdpV-4hUk4-QLpu3-1lWFxrtZKcggq3tWTduRo5_QebQbUUT_VSCgsFc
OmyWKoj56lbxthN19hq1XGWbLGfrrR6MWh23vk01zn8FVwi7uFwEnRYSafsnWLa1Z5
TpBj9GvAdl2H9NHwzpB5NqHpZNkQ3NMDj13Fn8fzO0JB83Etbm_tnFQfcb13X3bJ15
Cz-Ww1MGhvIpGGnMBT_ADp9xSIyAM9dQ1yeVXk-AIgWBUlN5uyWSGyCxp0cJwx7HxM
38z0UIeBu-MytL-eqndM7LxytsVzCbjOTSVRmhYEMIzUAnS1gs7uMQAGRdgRIElTJE
SGMjb_4bZq9s6Ve1LKkSi0_QDsrABaLe55UY0zF4ZSfOV5PMyPtocwV_dcNPlxLgNA
D1BFX_Z9kAdMZQW6fAmsfFle0zAoMe4l9pMESH0JB4sJGdCKtQXj1cXNydDYozF7l8
H00BV_Er7zd6VtIw0MxwkFCTatsv_R-GsBCH218RgVPsfYhwVuT8R4HarpzsDBufC4
r8_c8fc9Z278sQ081jFjOja6L2x0N_ImzFNXU6xwO-Ska-QeuvYZ3X_L31ZOX4Llp-
7QSfgDoHnOxFv1Xws-D5mDHD3zxOup2b2TppdKTZb9eW2vxUVviM8OI9atBfPKMGAO
v9omA-6vv5IxUH0-lWMiHLQ_g8vnswp-Jav0c4t6URVUzujNOoNd_CBGGVnHiJTCHl
88LQxsqLHHIu4Fz-U2SGnlxGTj0-ihit2ELGRv4vO8E1BosTmf0cx3qgG0Pq0eOLBD
IHsrdZ_CCAiTc0HVkMbyq1M6qEhM-q5P6y1QCIrwg.
0HFmhOzsQ98nNWJjIHkR7A

Appendix D. Acknowledgements

A JSON representation for RSA public keys was previously introduced by John Panzer, Ben Laurie, and Dirk Balfanz in Magic Signatures

[MagicSignatures].

Thanks to Matt Miller for creating the encrypted key example and to Edmund Jay and Brian Campbell for validating the example.

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:

Dirk Balfanz, Richard Barnes, John Bradley, Brian Campbell, Breno de Medeiros, Stephen Farrell, Joe Hildebrand, Edmund Jay, Stephen Kent, Ben Laurie, James Manger, Matt Miller, Kathleen Moriarty, Chuck Mortimore, Tony Nadalin, Axel Nennker, John Panzer, Eric Rescorla, Pete Resnick, Nat Sakimura, Jim Schaad, Ryan Sleevi, Paul Tarjan, Hannes Tschofenig, and Sean Turner.

Jim Schaad and Karen O'Donoghue chaired the JOSE working group and Sean Turner, Stephen Farrell, and Kathleen Moriarty 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]]

-36

- Stated that if both "use" and "key_ops" are used, the information they convey MUST be consistent.
- Clarified where white space and line breaks may occur in JSON objects by referencing Section 2 of RFC 7159.
- Specified that registration reviews occur on the jose-reg-review@ietf.org mailing list.

-35

- Used real values for examples in the IANA Registration Templates.

-34

- Addressed IESG review comments by Pete Resnick, Stephen Farrell, and Richard Barnes.
- Referenced RFC 4945 for PEM certificate delimiter syntax.

-33

- Addressed secdir review comments by Stephen Kent for which resolutions had mistakenly been omitted in the previous draft.

- Acknowledged additional contributors.

-32

- Addressed Gen-ART review comments by Russ Housley.

- Addressed secdir review comments by Stephen Kent.

-31

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

-30

- Added references and cleaned up the reference syntax in a few places.

- Applied minor wording changes to the Security Considerations section.

-29

- Replaced the terms JWS Header, JWE Header, and JWT Header with a single JOSE Header term defined in the JWS specification. This also enabled a single Header Parameter definition to be used and reduced other areas of duplication between specifications.

-28

- Revised the introduction to the Security Considerations section.

- Refined the text about when applications using encrypted JWKs and JWK Sets would not need to use the "cty" header parameter.

-27

- Added an example JWK early in the draft.

- Described additional security considerations.

- Added the "x5t#S256" (X.509 Certificate SHA-256 Thumbprint) JWK member.

- Addressed a few editorial issues.

-26

- Referenced Section 6 of RFC 6125 for TLS server certificate identity validation.
- Deleted misleading non-normative phrase from the "use" description.
- Noted that octet sequences are depicted using JSON array notation.
- Updated references, including to W3C specifications.

-25

- Updated WebCrypto reference to refer to W3C Last Call draft.

-24

- Corrected the authentication tag value in the encrypted key example.
- Updated the JSON reference to RFC 7159.

-23

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

-22

- Corrected RFC 2119 terminology usage.
- Replaced references to draft-ietf-json-rfc4627bis with RFC 7158.

-21

- Replaced the "key_ops" values "wrap" and "unwrap" with "wrapKey" and "unwrapKey" to match the "KeyUsage" values defined in the current Web Cryptography API [WebCrypto] editor's draft.
- Compute the PBES2 salt parameter as (UTF8(Alg) || 0x00 || Salt Input), where the "p2s" Header Parameter encodes the Salt Input value and Alg is the "alg" Header Parameter value.
- Changed some references from being normative to informative, addressing issue #90.

-20

- Renamed "use_details" to "key_ops" (key operations).

- Clarified that "use" is meant for public key use cases, "key_ops" is meant for use cases in which public, private, or symmetric keys may be present, and that "use" and "key_ops" should not be used together.

- Replaced references to RFC 4627 with draft-ietf-json-rfc4627bis, addressing issue #90.

-19

- Added optional "use_details" (key use details) JWK member.

- Reordered the key selection parameters.

-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.

-15

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

-14

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

-13

- Applied spelling and grammar corrections.

-12

- 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.

-11

- 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.

-10

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

-09

- Expanded the scope of the JWK specification to include private and symmetric key representations, as specified by draft-jones-jose-json-private-and-symmetric-key-00.

- Defined that members that are not understood must be ignored.

-08

- 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.

-07

- 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.

-06

- 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.

-05

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

-04

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

- Normatively reference XML DSIG 2.0 for its security considerations.

- 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/