JSON Web Key (JWK)

Abstract

A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) data structure that represents a cryptographic key. This specification also defines a 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 established by that specification.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7517.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Table of Contents

1. Introduction ........................................ 3
   1.1. Notational Conventions .......................... 3
2. Terminology ........................................... 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. 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 .............. 12
      8.1.1. Registration Template ...................... 12
      8.1.2. Initial Registry Contents ................. 13
   8.2. JSON Web Key Use Registry ..................... 15
      8.2.1. Registration Template ...................... 15
      8.2.2. Initial Registry Contents ................. 15
   8.3. JSON Web Key Operations Registry .............. 16
      8.3.1. Registration Template ...................... 16
      8.3.2. Initial Registry Contents ................. 16
   8.4. JSON Web Key Set Parameters Registry .......... 17
      8.4.1. Registration Template ...................... 17
      8.4.2. Initial Registry Contents ................. 18
   8.5. Media Type Registration ........................ 18
      8.5.1. Registry Contents ......................... 18
9. Security Considerations ............................. 19
   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
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 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 established 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] and JSON Web Encryption [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]. The interpretation should only be applied when the terms appear in all capital letters.

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, where STRING is a sequence of zero or more Unicode [UNICODE] characters.

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

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

2. Terminology

The terms "JSON Web Signature (JWS)", "Base64url Encoding", "Collision-Resistant Name", "Header Parameter", and "JOSE Header" are defined by the JWS specification [JWS].

The terms "JSON Web Encryption (JWE)", "Additional Authenticated Data (AAD)", "JWE Authentication Tag", "JWE Ciphertext", "JWE Compact Serialization", "JWE Encrypted Key", "JWE Initialization Vector", and "JWE Protected Header" are defined by the JWE specification [JWE].

The terms "Ciphertext", "Digital Signature", "Message Authentication Code (MAC)", and "Plaintext" are defined by the "Internet Security Glossary, Version 2" [RFC4949].

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.

**JWK Set**
A JSON object that represents a set of JWKs. The JSON object MUST have a "keys" member, which is an array of JWKs.

---

**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":"f830J3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
 "y":"x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
 "kid":"Public key used in JWS spec Appendix A.3 example"
}
```

Additional example JWK values can be found in Appendix A.

**4. JSON Web Key (JWK) Format**

A 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 whitespace 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 established by Section 8.1 or be a value that contains a Collision-Resistant Name.

4.1. "kty" (Key Type) Parameter

The "kty" (key type) parameter 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 established by [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 established by [JWA]; the initial contents of this registry are the values defined in Section 6.1 of [JWA].

The key type definitions include specification of the members to be used for those key types. Members used with specific "kty" values
can be found in the IANA "JSON Web Key Parameters" registry established by Section 8.1.

4.2. "use" (Public Key Use) Parameter

The "use" (public key use) parameter 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 to be used for public keys used for key agreement operations.

Additional "use" (public key use) values can be registered in the IANA "JSON Web Key Use" registry established by 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) parameter identifies the operation(s) for which the key is intended to be used. 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 [W3C.CR-WebCryptoAPI-20141211] 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_ops" (key operations) values can be registered in the IANA "JSON Web Key Operations" registry established by 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.
The "alg" (algorithm) parameter 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 established by [JWA] or be a value that contains a Collision-Resistant Name. The "alg" value is a case-sensitive ASCII string. Use of this member is OPTIONAL.

4.5. "kid" (Key ID) Parameter

The "kid" (key ID) parameter 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) parameter 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 [RFC5280]. 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) parameter 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 (Section 4 of [RFC4648] -- 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) parameter 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) parameter 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. JWK Set Format

A 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 JWKs. This JSON object MAY contain whitespace 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 established by 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, that are missing required members, or for which values are out of the supported ranges.
5.1. "keys" Parameter

The value of the "keys" parameter 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.

The registration procedure for values is Specification Required 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 Experts may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register JWK parameter: example").

Within the review period, the Designated Experts 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 Experts include determining whether the proposed registration duplicates existing functionality, whether it is likely to be of general applicability or useful only for a single application, and whether the registration description is clear.

IANA must only accept registry updates from the Designated Experts 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 Experts.
8.1. JSON Web Key Parameters Registry

This section 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 section 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 Experts state that there is a compelling reason to allow an exception. 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 Experts state that there is a compelling reason to allow an exception.
Parameter Information Class:
    Registers whether the parameter conveys public or private information. Its value must be either Public or Private.

Change Controller:
    For Standards Track RFCs, list the "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 or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. 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 RFC 7517

- 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 RFC 7517

- 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 RFC 7517

- Parameter Name: "alg"
  - Parameter Description: Algorithm
  - Used with "kty" Value(s): *
  - Parameter Information Class: Public
8.2. JSON Web Key Use Registry

This section 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 section 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 Experts state that there is a compelling reason to allow an exception.

**Use Description:**
Brief description of the use (e.g., "Digital Signature or MAC").

**Change Controller:**
For Standards Track RFCs, list the "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 or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. 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 RFC 7517

- **Use Member Value:** "enc"
  - **Use Description:** Encryption
  - **Change Controller:** IESG
  - **Specification Document(s):** Section 4.2 of RFC 7517
8.3. JSON Web Key Operations Registry

This section 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 section 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 Experts state that there is a compelling reason to allow an exception.

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

Change Controller:
For Standards Track RFCs, list the "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 or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. 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 RFC 7517

- Key Operation Value: "verify"
  - Key Operation Description: Verify digital signature or MAC
  - Change Controller: IESG
  - Specification Document(s): Section 4.3 of RFC 7517
8.4. JSON Web Key Set Parameters Registry

This section 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 section 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 Experts state that there is a compelling reason to allow an exception.

RFC 7517  JSON Web Key (JWK)  May 2015

Parameter Description:
Brief description of the parameter (e.g., "Array of JWK values").

Change Controller:
For Standards Track RFCs, list the "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 or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. 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 RFC 7517

8.5. Media Type Registration

8.5.1. Registry Contents

This section registers the "application/jwk+json" and "application/jwk-set+json" media types [RFC2046] in the "Media Types" registry [IANA.MediaTypes] in the manner described in RFC 6838 [RFC6838], which can be used to indicate that the content is a JWK or a JWK Set, respectively.

- 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 a JSON object; UTF-8 encoding SHOULD be employed for the JSON object.
Security Considerations: See the Security Considerations section of RFC 7517.
Interoperability Considerations: n/a
Published Specification: RFC 7517
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 RFC 7517.
Interoperability Considerations: n/a
Published Specification: RFC 7517
Applications that use this media type: OpenID Connect, Salesforce,
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.

In almost all cases, applications make decisions about whether to trust a key based on attributes bound to the key, such as names, roles, and the key origin, rather than based on the key itself. When an application is deciding whether to trust a key, there are several ways that it can bind attributes to a JWK. Two example mechanisms are PKIX [RFC5280] and JSON Web Token (JWT) [JWT].

For instance, the creator of a JWK can include a PKIX certificate in the JWK's "x5c" member. If the application validates the certificate and verifies that the JWK corresponds to the subject public key in
the certificate, then the JWK can be associated with the attributes in the certificate, such as the subject name, subject alternative names, extended key usages, and its signature chain.

As another example, a JWT can be used to associate attributes with a JWK by referencing the JWK as a claim in the JWT. The JWK can be included directly as a claim value or the JWT can include a TLS-secured URI from which to retrieve the JWK value. Either way, an application that gets a JWK via a JWT claim can associate it with the JWT's cryptographic properties and use these and possibly additional claims in deciding whether to trust the key.

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, except that the "x5u" JWK member is the only feature defined by this specification using TLS.

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]

[IANA.MediaTypes]
Internet Assigned Numbers Authority (IANA), "Media Types", <http://www.iana.org/assignments/media-types>.

[ITU.X690.1994]

Jones                        Standards Track                   [Page 21]

RFC 7517                JSON Web Key (JWK)            May 2015


10.2. Informative References


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. (Line breaks within values are for display purposes only.)

```json
{"keys": [
  {
    "kty":"EC",
    "crv":"P-256",
    "x":"MKBTC1fKUSDii11ySs3526iDZ8AiTo7Tu6KPAqv7D4",
    "y":"4Et66SRW2YiLUrN5vfvVHuhp7x8PxltnMWWlbM4IFyM",
    "use":"enc",
    "kid":"1"},

  {
    "kty":"RSA",
    "n": "0vx7agoebGcQSuuPiLJZXptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbfAAtvT86zu1RK7aPFFxuhDR1L6tSoc_BJECpebWKRJjBZcI7V4n3oknjhMs tn64tZ_2W-5JsGY4Hc5n9yBAXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FDW2QvzqY368QQMicAta5qsz8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CqOpbISD08qNLyrdkt-bFTWhAI4vMqFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqb w0LS1jF44-csFCur-kEgU8awapJzKnqDKgw",
    "e":"AQAB",
    "alg":"RS256",
    "kid":"2011-04-29"}
]}
```
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 a RSA algorithm. This example extends the example in the previous section, adding private key values. (Line breaks within values 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 within values are for display purposes only.)

```json
{"keys": [
  {
    "kty": "oct",
    "alg": "A128KW",
    "k": "GawgguFyGrWKav7AX4VKUg"
  },
  {
    "kty": "oct",
    "k": "AyM1SysPpbyDfgZld3umj1qzKObwVMkoqQ-EstJQLr_T-1qS0gZH75aKtMN3Yj0iPS4hcguTuwjAzr1Z9CAow",
    "kid": "HMAC key used in JWS spec Appendix 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 (with line breaks within values for display purposes only):

```json
{"kty":"RSA",
"use":"sig",
"kid":"1b94c",
"n":"vrj0fz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hjgAsz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEEnHU1ucuCLNwBuUiC0i1--7dYbsr4iJmG0Qu2j8DsVyl1azpJC_NG84Ty5KkthuCaPod7i7w0LK9orSMhBEwwZDCxTWq4aYWAwchc8t-emd9qOvWtVMDC2BXksRnhg6X5bUYLY6AyHKvj-nUy1wzgjYQDwH
```
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 (with line breaks within values for display purposes only):

```json
{
}
```
"kty":"RSA",
"kid":"juliet@capulet.lit",
"use":"enc",
"n":"t6Q8PWS1dkJj9hTP8hNYFlvadM7DfLW9mWepOJhJ66w7nyoK1gPNqFMSQRY0
t125Go-TEkodhW3r0uijvhhV7bcV0ll54w5ACGgPradJ6c5sR0-Iqom-QFCNP8
Sjg086MwoqQU_LYywLAgZ21WSdS_PERyGFInnj3Q0l8hns5jCtLcrwLHL0
Pb1fEv45AuRIuUfVcPySBWynDyGxvjYGDMS-AqWS9iQ2Zi1gT-QgUmipg0X
OC0Cc29rgLe2ymLHjpHc1tKvAvY5-L32-lSeZ0-0s6U15_aXrk9gw8cPuaX1
_I8sLGuISdVt3C_Fn2PZ3Zi8744FPPFGC1gqs2Wz-Q",
"e":"AQAB",
"d"
"GRtBiQmhOZtyszkgKdg4u_N-R_mZGU_9k7JQ_jn1DnfTuMdSNpTeaSTyWfS
NKuaAwnOEbIQVyi1QbWVVZ5NY3Ybc_IhUJtrf17bAY recovery WaCL3hdPKxy9U
vqYPyGR0KIXTQRQns-dV7jahl7LycrptMrM8dWBo4_pMaenNpPiqOg0xnu
ToxutRZjFyG40x4k4a3GORQd9CscZ2vsUDmsXOfUENoYMqA6cP1Mh33tsu
ry15k9qM5pG0X_IXAXmxzAh-tWiZowk2K4xyH9tS3LqlyY8C16WmeRDK2a
hecG85-oLQt5VepWHKmjo1_gj3Dsgqch96X525esAQ",
"p":"2r2n504hKSN8sS4CgcQHfbst08XboFqDkum3sc4h3GRxrTmDl1ZK9uw-PIHF
QP0FkxxXrvr-WE-ZErqhivH_2iCLU5SWa16XhAr1K1KaUxPPSYBY9yk31s0Q8
UK96E3_0rAAdAyTAJs-M33xClfNgqh56HDnEtTQhH3rCT5T3yjWrs",
"q"
"1u_RiFDP7LBylh3N4GLT90pSKYp0UZyiaZWB0CBNJ6qXaj10RWjsU0c6I
edis4S7B_coSKB0Kj9PaBzg-IyilorvQQuPamQu66rihMhjVt6GTLv8CLCYK
ryL52zjQk0E_ym2QnkwSUX7eYT7LbAHRK9qocDE5008F804Is",
"dp"
"KkMTWqBueFvWz2_DbjipPQqyHSHjj990L5XMOzqYAJMcLMZtbUtwKqvVDq3
thE03ZIcohbDtt6SbfMvZggabpQxNxBpooO5f_a_HgMXL_hqigI4y_kq51w
Y521wUn5rRr-jYo1h41KR-vz2PyhEAyYrhtwtxtvQlCRVid6c",
"dq"
"AvFs0-gRxvn0bwJomsNfXyCkl1Wnuej-QFluMGfwGiltQBWtfZ1er7t1xDkbN9
GQTB9yqDoYaN06H7CftrkxHJBqaj6nkF5KKS3Tq5qczK0mx3e3KRbBy
mXbbx5qWpU5ELD5xFeicafWY63TmAmUUlLRFCC33sDea-octs",
"qi"
"LSqi-w9CPyUREmEP1rsBL7KwNt0V5eOpFmAqu0Qw57NBucxSeoPwmuUqQ
abu9w0-Py4dQ57_bapoKRu1980bvu Fn63sHEFglZVqJDMeAvmjm4sm-Fp00
Yu_neotgQ0hzbI5gry7ajdYy9-2Lnx_76aBZo0U9HCJ-UsfS018"
}

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, 101, 64, 99, 97, 112,
117, 105, 100, 34, 58, 34, 116, 34, 58, 34, 107, 110, 34, 58, 34, 107, 100, 34,
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):
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:


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:


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:

Ye9j1qs22DmRSAAddIh-VnA

C.7. Additional Authenticated Data

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


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:

The resulting Authentication Tag value is:

```
[208, 113, 102, 132, 236, 236, 67, 223, 39, 53, 98, 99, 32, 121, 17, 236]
```
value (with line breaks for display purposes only):

AwhB8lxrlKjF02LGWeq27HHt9yZAbFv3p5ZicHpj64QyHC44qzlZ3EmmZTqO
wIqZJ13jbyHB8LgPiqU1h6F2HPGzw8L-meeQ0jvDUTre07NtorB8bwBQyZ6g
0kQ3-DEIOg1fYxV8-FJvN6Wbq6ISbc6d6-7i0tjSHV-8D3r-3jCRIe0S5K0y30i34Z
G0iAc1EK21B11c_AE11rIrrLblsQaKd-098Kap-UgmyWPFreu3J3nP
bD4V9e950wEfMGLOpM2JnaTDWqOkoJ_xplq2VnPz8iguLchBoKllyqFJLl2oMW
wqhB9o9j-0800as5mmLSvQMTf1IrEBBmZMHBZ8EWF9fWvwF0DWQjGKnHmBZQ-3
lvqTc-M6-gWAGD9PhOn4Fp2oib2HlizGtIeA8GryUpfluljC1le1DQKg0ewhKuKz
h04DKMN5Mbugf2atmu90P0ldx5eCUtRG1gMv7q5ZXHTjgPDr5b2N731uOocGAU
qHdg0hgh9OVJv_ObCtdjsJH4C1JSsdUhrXvyX3hJh2Xd7cWjRzu3_Y1GxYU6-s3GFBi
rfqQeIpJDBTHpcoCmyrwyHYFglnqBZRotRrsS95rg8F59brXqsaD7UgQGwb7Wby665
d0zpvTasvxFx_c0wMAl-nefArK0w_Px6g4EDujaGJGnSxLsLw_0ovdApDIFLHYE
PyagyHjouQUiQg7mBwyrwaF0t6gB8hv8omLNFEMdPjAzUuMuh6tBDwGKzd-Ts_
ub9hxrpJ4us0Wnt5rGyuoN2N_c1-TqlxXm5oto14xnoAyBQPwIEgH3Y4zhwBKhH
Pj50sGcdwNydPp-PyvF-2NZzOi10vWQBRHSbPyWyz-xbGkgD504LrtQwCO7CC_
_CyyURilSessPvSsM1RACKXj4LE0e82TidqqkJ0juRfK5rqLiq8nBE9osqQDAs0oFQZ1
GrBrqxDsNYIAWmxkoss-16n3x4QftByvVx5sCEU0M_0MqG7C0xZWMOPEFraepCoy
rv0UIng8i8l9JKBxxETY2BpgKbXlCsaAuAkAMSc9AiB0AO0HymHtqtvMks07AE
hNC2-YzPyx1KfHnoS4LLeE_pFSmLmj46P1NSge9C5G5tETYXGA66b1XzhHtmwSmPsc
cr9LWhVmAaA7_bxyObnFUXgWtK4vzzQbJz363UTk40TB-JvKnFVcFwsaw5WChj0o
4oJp07d2yNZ7MFWaj2HTeab9uwumQ0THmBduZ-QQNZ0ysB0ye7STC1vVme0JNjrF_c3Re
hKTfmdlXGVldPzCplrz7QqRqH8JP-l4mEQvCaWg9ONHlemczGOS-A-wunmwwJi
B1vvgJRFfdpdv-4uHk-QLpqu-1WFrztZKcgq3tWtDuRoQ_0ebQuUT_VSgjF
OmyWkoj51LbxthN19hqXGWbLgErR6MWhv23k01zn8Wk6i7JuFwErNysfswlA1Z5
Tpbj9GvAd1L29HNzwpB5QnHq2PKZ0QMDm13jF8nfs0J8B3E7bmbn_TnFqeb13Xb2lS
Cz-Kw1MGhvIgPnGtnpR_dAp9xSiyAM9dQ1yeVXk-AnGWBUn5uyWSgyCp0cJWx7Hmx
38z0UeBu-MytLeqndM7XlytsVzbjOTSVRNhYEM1zAvnS1gs7uMQAGRdgRIELTJE
SGMjy_b4Bz9g6sVeILKksIo_QDsrABlaLe55UY0z4Fs5OV5MPtowcDncPjLxGNa
D1BFx_Z9kAdM2Zw6fAmsfLe02A0eMe41p0pMESH0BjAs3jGckCQxj1CxnYDzy0Z78
H00BV_Er7z6Gtv0xMxwwFCtatsv_R-GsBCH2eRgVpsYhWvU7tR4HarpsDjBfC4
r8_c8f9Z278sQS08ljfj0ja6L2x0nKzImNFUX6w0-SqAuyQyzX3L3120X4l-7Q5fGDoHn0xV1xw5-D5mHDHZ3x0up2b2ppqTkbz9eW2vUVW1M80I9atBFPrkMAO
v9omA-6wV5UH0-1WmHlqg8vnswp-Javonc4t6RVUzuqJN0oDncBCG0vH1J7CTC
88LQxsqlLHHiu4Fz-U2SGnxLtgj0-ihiT2ELGRv4v08E1BosTm0cx3qgG0Pq0eOLBD
I Hzsdz.srcAGrC0kVhMbyqlM6eqEHm-q5P6ylQCfkg

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

0HFmh0zsQ98nNWJjIHkR7a
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 (with line breaks for display purposes only) is:

eyJhbGciOiJQQkVTMi1INUzI1NitBMTI4S1ciLCJwMnMiOiIiY0NUY0paMVJ2ZSF9DSn
VKcmwUTF3IiwiDjJjio0MDk2LCjibmM10iJBMIT4Q0JDLUhTMjU2IiwY3R5Ijoi
andr2pz24ifQ.
TrqXowuNUfDV9VPTNbyGvEJ9MjefAvn-TR1iuIXr9p6hsRq69T7kBA.
Ye9j1qis2Dm RSAdHa-sNe-Vh-Aa.
AwhB8Ixrlkfn0Z2LGwEqg27HT4g9fyzAbFv3p5zicHjp64QyHC44qqlJ3EmnZTqo
wIQZ3JjbyHB8LgePiqUJhf62MHPHgzB8L-1mEEQjfiIvURTe097n0torBk8bzwBQyz6g
0kQ3DE0IgfLyxV8-FJ0NYBwqN1bc6d_i70tjSHV-8DRi9p-13CRieO5kY30i34Z4
G0iAc1EK2B11c_AE1IIPII_wvvtRiUigY8oqYXakWd1_098Kap-UgmyWPFreUJ3lp
nbD4v950weFmGLOPfo2MjnaTDcwQoko-xplqZvN2p8ziugLchBoklyQFJL2mOWB
wqhu909j-080oas5mmLSvQMfTIrIEbBZMhHZMBZ8EFw9fWwwFuODWJGKlnMnHzBZ-3
LvTc-M6-gWAD6DPhONfP20ib2HgizG1eA8GRyUpfUfLuljClIE1dGowekhKZ
h04DKMNbngf2atmU90P0Ldx5peCUTRG1gMVL7Qup5 ZhTjgPDnPb5bn731U0oCAU
qHdg5ghgOJrj_ObCtdjsH04C1FSJsdUhrXtXy3HJh2X7dw3RzuU_3Y16gYU6-e3gFpi
rfqqEipJDBTHpcocMynrwYjYHfnglaQRzotRrsR95gs59f5BRQxaSAyDUgQGWBoWQ6y65
0p2ypvTasvxFx_c0OMWA-neeFAkOWxP6gE4UdJGI5XMxv9CStL7w_00ovDApDILHY
HePyagyHjouQUuGiq7BsYwyrwaF06tg8hV8omLNfMEMDPJazUzMwHw6tBDwGkzdt-s
ub9hxpjr34uswnt5g7uyonN2n-ci-TQlxm5otol4MxnoAyBYBPwIegSH3Y4zhwKBhB
Pjso0cdwnyDyGBPp-UUyf-2NzzD0iQ10vWQ9RHSBPWyz_xbgkgD504LRqtwCoCC
_CyuURYilsEssPvSMRX_4U4LE0c82tiDdqjKOjRFK5rLiq8Be9s0Q5DlaOoFzqi
GrBrqxsDNSyIAmxxkos-13nX4qtByxV85sCE5U_0MrqG7C0xZWMOPEFrDepAv-cOy
rvoUing8i8ljjKBKkETY2BpgEgkBYCxsAUAkKamSC9CiBiAxAO0UHyTqtLaMsk0AE
hNC2-YzPyx1FkhMs4LLe6E_PFsmLmjA6P1NSged9C5G5eTEYXGAN6bixZBhTmwrPsC
oro9LWhVmAaA7_bxyObfnFuxgWtK4vzzQjB3J6UtK40TB-JvkWgfWVcfsaw5WChj6o
4jp072dyN7WMFaJz2hTEabz9wumQ@TMhBduZ-QQN3pYoBS7TSClvMe0NjRFc_jRe
hKTFmdlXGVldPzCzCprr7ZQQgRQhF8JP-14MEvQnaCwGnONHlemczGOS-A-wtmmwvji
B1v_vgJRf4FdpV-4hUk4-QLpu3-1LWFxrtZKcggq3tW7dUoRo5_QebqBUT_UVSCgsFc
OmyWkoj56ibxthN19h1QXGwLGFrrR6MWhv23kv01zn8FvWi7fFEnRysafswsWALAZ
Tbpj9GnvAdl2H9NHwzpB5NqHpnZNKq3MNDj13Fn8zf08J883Etbmn_Fqfbc13Xb3J15
Cz-Ww1MhGwIpGnMBT_Adp9x5iyAM9qD1yexVxk-AIgwBLN5uuyWSeqCxp6CjxW7HxA
30z0UEBu-Myl-tCfNdlxtysVzcbjO5TVRhenYEMizAUAnS1gs7uQMAGRdflETJLE
SGMjbj_4bZq9s6eV1KksIiQ_QdseABaeL555UyOzF4ZS0V5PMtoCwv_dncPNlXg
D1BFX_Z9kAdMZW6FAsmsfFleOzaAmoMe4l9p0MESH0B4sJGdcKtQxj1CNYndDYoZ7Fl
H009Bv_Er7zd6Vtw0MwwFCtatsv_R-GsBCH218RVPsfYhvuV8T84HarpzsBduF4cr
r8_c8fc9Z278sQ081jffOjaL2x0nINzMFXU6Xw0-Ska-QeuvYX3Z_L3120X4Lp-
7QsFGDoHn0Xf1Xws-D5mDHD3zxOup2b2TppkDTZ9b9eW2vXXvUviM80I9atBFPKMGAO
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:


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.

Author's Address

Michael B. Jones
Microsoft

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