JSON Web Key (JWK) Thumbprint
draft-jones-jose-jwk-thumbprint-00

Abstract

This specification defines a means of computing a thumbprint value
(a.k.a. digest) of JSON Web Key (JWK) objects analogous to the "x5t"
(X.509 Certificate SHA-1 Thumbprint) value defined for X.509
certificate objects. This specification also registers the new JSON
Web Signature (JWS) and JSON Web Encryption (JWE) Header Parameters
and the new JSON Web Key (JWK) member name "jkt" (JWK SHA-256
Thumbprint) for holding these values.

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 October 12, 2014.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a means of computing a thumbprint value
(a.k.a. digest) of JSON Web Key (JWK) [JWK] objects analogous to the
"x5t" (X.509 Certificate SHA-1 Thumbprint) value defined for X.509
certificate objects. This specification also registers the new JSON
Web Signature (JWS) [JWS] and JSON Web Encryption (JWE) [JWE] Header
Parameters and the new JSON Web Key (JWK) [JWK] member name "jkt"
(JWK SHA-256 Thumbprint) for holding these values.

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].

2. Terminology

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

This term is defined by this specification:

JWK Thumbprint
   The digest value for a key that is the subject of this
   specification.

3. JSON Web Key (JWK) Thumbprint

This specification defines the thumbprint of a JSON Web Key (JWK)
value as being a function of the REQUIRED members of the key's JWK
representation. Specifically, this function is the SHA-256 hash of
the octets of the UTF-8 representation of a JSON object [RFC7159]
constructed containing only the REQUIRED members of a JWK
representing the key and with no white space or line breaks before or
after any syntactic elements and with the REQUIRED members ordered
lexicographically by the Unicode code points of the member names.
This JSON object is itself a legal JWK representation of the key
value. The details of this computation are further described in
subsequent sections.

3.1. Example JWK Thumbprint Computation

   This section demonstrates the JWK Thumbprint computation for the JWK
   below (with long lines broken for display purposes only):

```
{
 "kty": "RSA",
  "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbfAAt
        VT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMstn6
        4tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FD
        W2QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n9
        1CbOpbISD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINH
        aQ-G_xBniIqbw0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
 "e": "AQAB",
 "alg": "RS256",
 "kid": "2011-04-29"
}
```

   As defined in JSON Web Key (JWK) [JWK] and JSON Web Algorithms (JWA)
   [JWA], the REQUIRED members of an RSA public key are:

   o "kty"
   o "n"
   o "e"

   Therefore, these are the members used in the thumbprint computation.

   Their lexicographic order (see more about this in Section 3.3) is:

   o "e"
   o "kty"
   o "n"

   Therefore the JSON object constructed as an intermediate step in the
   computation is as follows (with long lines broken for display
   purposes only):

```
{"e":"AQAB","kty":"RSA","n":"0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2
aiAFbWhM78LhWx4cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCi
FV4n3oknjhMstn64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65Y
GjQR0_FDW2QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n
91CbOpbISD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_x
BniIqbw0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw"}
```

   The octets of the UTF-8 representation of this JSON object are:

   [123, 34, 101, 34, 58, 34, 65, 81, 65, 66, 34, 44, 34, 107, 116, 121,
   34, 58, 34, 82, 83, 65, 34, 44, 34, 110, 34, 58, 34, 48, 118, 120,

      55, 97, 103, 111, 101, 98, 71, 99, 81, 83, 117, 117, 80, 105, 76, 74,
      88, 90, 112, 116, 78, 57, 110, 110, 100, 114, 81, 109, 98, 88, 69,
      112, 115, 50, 97, 105, 65, 70, 98, 87, 104, 77, 55, 56, 76, 104, 87,
      120, 52, 99, 98, 98, 102, 65, 65, 116, 86, 84, 56, 54, 122, 119, 117,
      49, 82, 75, 55, 97, 80, 70, 70, 120, 117, 104, 68, 82, 49, 76, 54,
      116, 83, 111, 99, 95, 66, 74, 69, 67, 80, 101, 98, 87, 75, 82, 88,
      106, 66, 90, 67, 105, 70, 86, 52, 110, 51, 111, 107, 110, 106, 104,
      77, 115, 116, 110, 54, 52, 116, 90, 95, 50, 87, 45, 53, 74, 115, 71,
      89, 52, 72, 99, 53, 110, 57, 121, 66, 88, 65, 114, 119, 108, 57, 51,
      108, 113, 116, 55, 95, 82, 78, 53, 119, 54, 67, 102, 48, 104, 52, 81,
      121, 81, 53, 118, 45, 54, 53, 89, 71, 106, 81, 82, 48, 95, 70, 68,
      87, 50, 81, 118, 122, 113, 89, 51, 54, 56, 81, 81, 77, 105, 99, 65,
      116, 97, 83, 113, 122, 115, 56, 75, 74, 90, 103, 110, 89, 98, 57, 99,
      55, 100, 48, 122, 103, 100, 65, 90, 72, 122, 117, 54, 113, 77, 81,
      118, 82, 76, 53, 104, 97, 106, 114, 110, 49, 110, 57, 49, 67, 98, 79,
      112, 98, 73, 83, 68, 48, 56, 113, 78, 76, 121, 114, 100, 107, 116,
      45, 98, 70, 84, 87, 104, 65, 73, 52, 118, 77, 81, 70, 104, 54, 87,
      101, 90, 117, 48, 102, 77, 52, 108, 70, 100, 50, 78, 99, 82, 119,
      114, 51, 88, 80, 107, 115, 73, 78, 72, 97, 81, 45, 71, 95, 120, 66,
      110, 105, 73, 113, 98, 119, 48, 76, 115, 49, 106, 70, 52, 52, 45, 99,
      115, 70, 67, 117, 114, 45, 107, 69, 103, 85, 56, 97, 119, 97, 112,
      74, 122, 75, 110, 113, 68, 75, 103, 119, 34, 125]

   The JWK Thumbprint value is the SHA-256 hash of these octets,
   specifically:

   [55, 54, 203, 177, 120, 124, 184, 48, 156, 119, 238, 140, 55, 5, 197,
   225, 111, 251, 158, 133, 151, 21, 144, 31, 30, 76, 89, 177, 17, 130,
   245, 123]

   The base64url encoding of this JWK Thumbprint value (which would be
   used in the "jkt" members registered below) is:

     NzbLsXh8uDCcd-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs

3.2. JWK Members Used in the Thumbprint Computation

   Only the REQUIRED members of a key's representation are used when
   computing its JWK Thumbprint value. As defined in JSON Web Key (JWK)
   [JWK] and JSON Web Algorithms (JWA) [JWA], the REQUIRED members of an
   elliptic curve public key, in lexicographic order, are:

   o "crv"
   o "kty"
   o "x"
   o "y"

   the REQUIRED members of an RSA public key, in lexicographic order,

are:

o "e"
o "kty"
o "n"

and the REQUIRED members of a symmetric key, in lexicographic order, are:

o "k"
o "kty"

As other key type values are defined, the specifications defining them should be similarly consulted to determine which members, in addition to "kty", are REQUIRED.

3.2.1. JWK Thumbprint of a Private Key

The JWK Thumbprint of a private key is computed as the JWK Thumbprint of the corresponding public key. This has the intentional benefit that the same JWK Thumbprint value can be computed both by parties using either the public or private key. The JWK Thumbprint can then be used to refer to both keys of the key pair. Application context can be used to determine whether the public or the private key is the one being referred to by the JWK Thumbprint.

3.2.2. Why Not Include Optional Members?

OPTIONAL members of JWKs are intentionally not included in the JWK Thumbprint computation so that their absence or presence in the JWK doesn't alter the resulting value. The JWK Thumbprint value is a digest of the key value itself -- not of additional data that may also accompany the key.

3.3. Order and Representation of Members in Hash Input

The REQUIRED members in the input to the SHA-256 hash function are ordered lexicographically by the Unicode code points of the member names.

Characters in member names and member values MUST be represented in the simplest manner possible. In particular, all characters within the Basic Multilingual Plane besides quotation mark, reverse solidus, and the control characters (U+0000 through U+001F) MUST NOT be escaped.

If the JWK key type uses members whose values are themselves JSON objects (as of the time of this writing, none are defined that do),

   the members of those objects must likewise be lexicographically
   ordered.

   See Section 5 for a discussion of further practical considerations
   pertaining to the representation of the hash input.

3.4. JWK Thumbprints of Any Keys

   Note that a key need not be in JWK format to create a JWK Thumbprint
   of it. The only prerequisites are that the JWK representation of the
   key be defined and the party creating the JWK Thumbprint is in
   possession of the necessary key material. These are sufficient to
   create the hash input, as described in Section 3.3.

4. "jkt" Member Definitions

   This section defines "jkt" (JWK SHA-256 Thumbprint) members used for
   holding base64url encoded JWK Thumbprint values in JWK, JWS, and JWE
   objects.

4.1. "jkt" (JWK SHA-256 Thumbprint) JWS Header Parameter

   The "jkt" (JWK SHA-256 Thumbprint) JWS Header Parameter is a
   base64url encoded JWK Thumbprint (a.k.a. digest) of the public key
   that corresponds to the key used to digitally sign the JWS. Use of
   this JWS Header Parameter is OPTIONAL.

4.2. "jkt" (JWK SHA-256 Thumbprint) JWE Header Parameter

   This parameter has the same meaning, syntax, and processing rules as
   the "jkt" JWS Header Parameter defined in Section 4.1, except that
   the JWK Thumbprint references the public key to which the JWE was
   encrypted; this can be used to determine the private key needed to
   decrypt the JWE.

4.3. "jkt" (JWK SHA-256 Thumbprint) JWK Parameter

   The "jkt" (JWK SHA-256 Thumbprint) JWK parameter is a base64url
   encoded JWK Thumbprint (a.k.a. digest) of the JWK. If present, the
   JWK Thumbprint value represented MUST have been computed from the
   other members of the JWK as described in Section 3. Use of this
   member is OPTIONAL.

4.4. Possible Future Alternative Thumbprint Computations

   If, in the future, JWK Thumbprints need to be computed using hash
   functions other than SHA-256, it is suggested that additional related

   JWK, JWS, and JWE parameters be defined for that purpose. For
   example, it is suggested that a new "jkt#S3-256" (X.509 Certificate
   Thumbprint using SHA-3-256) JWK parameter could be defined by
   registering it in the IANA JSON Web Key Parameters registry and the
   IANA JSON Web Signature and Encryption Header Parameters registry.

5. Practical JSON and Unicode Considerations

   Implementations will almost certainly use functionality provided by
   the platform's JSON support, such as the JavaScript JSON.parse()
   JSON.stringify() functions, when parsing the JWK and emitting the
   JSON object used as the SHA-256 hash input. As a practical
   consideration, future JWK member names should be avoided for which
   different platforms or libraries might emit different
   representations. As of the time of this writing, currently all
   defined JWK member names use only printable ASCII characters, which
   should not exhibit this problem.

   In particular, while the operation of lexicographically ordering
   member names by their Unicode code points is well defined, different
   platform sort functions may produce different results for non-ASCII
   characters, in ways that may not be obvious to developers. If
   writers of future specifications defining new JWK Key Type values
   choose to restrict themselves to ASCII member names (which are for
   machine and not human consumption anyway), some future
   interoperability problems might be avoided.

   Unnecessary use of escaping in the input JWK representation should
   likely also be avoided.

   All of these practical considerations are really an instance of Jon
   Postel's principle: "Be liberal in what you accept, and conservative
   in what you send."

6. IANA Considerations

6.1. JWS and JWE Header Parameter Registration

   This specification registers the "jkt" Header Parameters defined in
   Sections 4.1 and 4.2 in the IANA JSON Web Signature and Encryption
   Header Parameters registry defined in [JWS].

6.1.1. Registry Contents

   o Header Parameter Name: "jkt"
   o Header Parameter Description: JWS JWK Thumbprint
   o Header Parameter Usage Location(s): JWS
   o Change Controller: IETF
   o Specification Document(s): Section 4.1 of [[this document]]

   o Header Parameter Name: "jkt"
   o Header Parameter Description: JWE JWK Thumbprint
   o Header Parameter Usage Location(s): JWE
   o Change Controller: IETF
   o Specification Document(s): Section 4.2 of [[this document]]

6.2. JSON Web Key Parameters Registration

   This specification registers the "jkt" JWK member defined in
   Section 4.3 in the IANA JSON Web Key Parameters registry defined in
   [JWK].

6.2.1. Registry Contents

   o Parameter Name: "jkt"
   o Parameter Description: JWK Thumbprint
   o Used with "kty" Value(s): *
   o Parameter Information Class: Public
   o Change Controller: IESG
   o Specification Document(s): Section 4.3 of [[this document]]

7. Security Considerations

   The JSON Security Considerations and Unicode security considerations
   described in Sections 10.2 and 10.3 of JSON Web Signature (JWS) [JWS]
   also apply to this specification.

   Also, as described in Section 5, some implementations may produce
   incorrect results if esoteric or escaped characters are used in the
   member names. The security implications of this appear to be limited
   for JWK Thumbprints of public keys, since while it may result in
   implementations failing to identify the intended key, it should not
   leak information, since the information in a public key is already
   public in nature, by definition.

8. Normative References

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

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

   [JWK]   Jones, M., "JSON Web Key (JWK)",
               draft-ietf-jose-json-web-key (work in progress),
               March 2014.

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

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997.

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

Appendix A. Document History

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

   -00

   o Created draft-jones-jose-jwk-thumbprint.

Author's Address

   Michael B. Jones
   Microsoft

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