JOSE Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 30, 2015
M. Jones
Microsoft
N. Sakimura
NRI
February 26, 2015

JSON Web Key (JWK) Thumbprint
draft-ietf-jose-jwk-thumbprint-03

Abstract

This specification defines a means of computing a thumbprint value (a.k.a. digest) of a key represented as a JSON Web Key (JWK).

Table of Contents

1. Introduction 3
1.1. Notational Conventions 3
2. Terminology 3
3. JSON Web Key (JWK) Thumbprint 3
3.1. Example JWK Thumbprint Computation 4
3.2. JWK Members Used in the Thumbprint Computation 5
3.2.1. JWK Thumbprint of a Private Key 6
3.2.2. Why Not Include Optional Members? 6
3.3. Order and Representation of Members in Hash Input 7
3.4. JWK Thumbprints of Keys Not in JWK Format 7
4. Practical JSON and Unicode Considerations 8
5. IANA Considerations 9
6. Security Considerations 9
7. Relationship to Digests of X.509 Values 10
8. References 10
8.1. Normative References 10
8.2. Informative References 10
Appendix A. Acknowledgements 11
Appendix B. Document History 11
Authors' Addresses 12

1. Introduction

This specification defines a means of computing a thumbprint value (a.k.a. digest) of a key represented as a JSON Web Key (JWK). This value can be used for identifying or selecting the key that is the subject of the thumbprint, for instance, by using the base64url encoded JWK Thumbprint value as a "kid" (key ID) value.

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], 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

The thumbprint of a JSON Web Key (JWK) is computed as follows:

1. Construct a JSON object [RFC7159] 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 [UNICODE] code points of the member names. (This JSON object is itself a legal JWK representation of the key.)

2. Hash the octets of the UTF-8 representation of this JSON object with a cryptographic hash function H. For example, SHA-256 [SHS] might be used as H.

The resulting value is the JWK Thumbprint with H of the JWK. 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]

Using SHA-256 [SHS] as the hash function H, the JWK SHA-256 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 [JWS] of this JWK SHA-256 Thumbprint value (which might, for instance, be used as a "kid" (key ID) value) 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 for the curves specified in Section 6.2.1.1 of [JWK], in lexicographic order, are:

- "crv"
- "kty"
- "x"

- "y"

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

- "e"
- "kty"
- "n"

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

- "k"
- "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.

This specification defines the method of computing JWK Thumbprints of private keys for interoperability reasons -- so that different implementations computing JWK Thumbprints of private keys will produce the same result.

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.

Optional members are not included so that the JWK Thumbprint refers to a key -- not a key with an associated set of key attributes. This has the benefit that while in different application contexts different subsets of attributes about the key might or might not be included in the JWK, the JWK Thumbprint of the key remains the same regardless of which optional attributes are present. Different kinds

of thumbprints could be defined by other specifications that might include some or all additional JWK members, should use cases arise where such different kinds of thumbprints would be useful. See Section 9.1 of [JWK] for notes on some ways to cryptographically bind attributes to a key.

3.3. Order and Representation of Members in Hash Input

The required members in the input to the hash function are ordered lexicographically by the Unicode code points of the member names.

Characters in member names and member values MUST be represented without being escaped. This means that thumbprints of JWKs that require such characters are not defined by this specification. (This is not expected to limit the applicability of this specification, in practice, as the members of JWK representations are not expected to use any of these characters.) The characters specified as requiring escaping by Section 7 of [RFC7159] are quotation mark, reverse solidus (a.k.a. backslash), and the control characters U+0000 through U+001F.

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.

If the JWK key type uses members whose values are JSON numbers (as of the time of this writing, none are defined that do), if the numbers are integers, they MUST be represented as a JSON number as defined in Section 6 of [RFC7159] without including a fraction part or exponent part. For instance, the value "1.024e3" MUST be represented as "1024". This means that thumbprints of JWKs that use numbers that are not integers are not defined by this specification. Also, as noted in The I-JSON Message Format [I-D.ietf-json-i-json], implementations cannot expect an integer whose absolute value is greater than 9007199254740991 (i.e., that is outside the range [-(2**53)+1, (2**53)-1]) to be treated as an exact value.

See Section 4 for a discussion of further practical considerations pertaining to the representation of the hash input.

3.4. JWK Thumbprints of Keys Not in JWK Format

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 from the JWK representation of the key, as

described in Section 3.3.

4. Practical JSON and Unicode Considerations

Implementations will almost certainly use functionality provided by the platform's JSON support when parsing the JWK and emitting the JSON object used as the 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. Note however, that JSON.stringify() cannot be counted on to lexicographically sort the members of JSON objects, so while it may be able to be used to emit some kinds of member values, different code is likely to be needed to perform the sorting.

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.

However, if new JWK members are defined that use non-ASCII member names, their definitions should specify the exact Unicode code point sequences used to represent them, particularly in cases in which Unicode normalization could result in the transformation of one set of code points into another under any circumstances.

Use of escaped characters in the input JWK representation SHOULD be avoided.

While there is a natural representation to use for numeric values that are integers, this specification doesn't attempt to define a standard representation for numbers that are not integers or that contain an exponent component. This is not expected to be a problem in practice, as the required members of JWK representations are not expected to use numbers that are not integers.

Use of number representations containing fraction or exponent parts in the input JWK representation SHOULD 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."

5. IANA Considerations

This specification makes no requests of IANA.

6. Security Considerations

The JSON Security Considerations and Unicode Comparison 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 4, 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.

A hash of a symmetric key has the potential to leak information about the key value. Thus, the JWK Thumbprint of a symmetric key should be typically be concealed from parties not in possession of the symmetric key, unless in the application context, the cryptographic hash used, such as SHA-256, is known to provide sufficient protection against disclosure of the key value.

A JWK Thumbprint will only uniquely identify a particular key if a single unambiguous JWK representation for that key is defined and used when computing the JWK Thumbprint. (Such representations are defined for all the key types defined in JSON Web Algorithms (JWA) [JWA].) For example, if an RSA key were to use "e":"AAEAAQ" (representing [0, 1, 0, 1]) rather than the specified correct representation of "e":"AQAB" (representing [1, 0, 1]), a different thumbprint value would be produced for what could be effectively the same key, at least for implementations that are lax in validating the JWK values that they accept. Thus, JWK Thumbprint values can only be relied upon to be unique for a given key if the implementation also validates that the correct representation of the key is used.

Even more insidious is that an attacker may supply a key that is a transformation of a legal key in order to have it appear to be a different key. For instance, if a legitimate RSA key uses a modulus value N and an attacker supplies a key with modulus 3*N, the modified key would still work about 1/3 of the time, but would appear to be a different key. Thus, while thumbprint values are valuable for identifying legitimate keys, comparing thumbprint values is not a reliable means of excluding (blacklisting) the use of particular keys (or transformations thereof).

7. Relationship to Digests of X.509 Values

JWK Thumbprint values are computed on the members required to represent a key, rather than all members of a JWK that the key is represented in. Thus, they are more analogous to applications that use digests of X.509 Subject Public Key Info (SPKI) values, which are defined in Section 4.1.2.7 of [RFC5280], than to applications that use digests of complete certificate values, as the "x5t" (X.509 Certificate SHA-1 Thumbprint) [JWS] value defined for X.509 certificate objects does. While logically equivalent to a digest of the SPKI representation of the key, a JWK Thumbprint is computed over a JSON representation of that key, rather than over an ASN.1 representation of it.

8. References

8.1. Normative References

[JWA] Jones, M., "JSON Web Algorithms (JWA)", draft-ietf-jose-json-web-algorithms (work in progress), January 2015.

[JWK] Jones, M., "JSON Web Key (JWK)", draft-ietf-jose-json-web-key (work in progress), January 2015.

[JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", draft-ietf-jose-json-web-signature (work in progress), January 2015.

[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.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, March 2012.

[UNICODE] The Unicode Consortium, "The Unicode Standard", 1991-, <http://www.unicode.org/versions/latest/>.

8.2. Informative References

[I-D.ietf-json-i-json] Bray, T., "The I-JSON Message Format", draft-ietf-json-i-json-06 (work in progress),

January 2015.

[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.

Appendix A. Acknowledgements

James Manger and John Bradley participated in discussions that led to the creation of this specification. Jim Schaad also contributed to this specification.

Appendix B. Document History

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

-03

- Addressed review comments by James Manger and Jim Schaad, including adding a section on the relationship to digests of X.509 values.

-02

- No longer register 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.

- Added security considerations about the measures needed to ensure that a unique JWK Thumbprint value is produced for a key.

- Added text saying that the base64url encoded JWK Thumbprint value could be used as a "kid" (key ID) value.

- Broke a sentence up that used to be way too long.

-01

- Addressed issues pointed out by Jim Schaad, including defining the JWK Thumbprint computation in a manner that allows different hash functions to be used over time.

- Added Nat Sakimura as an editor.

-00

o Created draft-ietf-jose-jwk-thumbprint-00 from draft-jones-jose-jwk-thumbprint-01 with no normative changes.

Authors' Addresses

Michael B. Jones
Microsoft

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

Nat Sakimura
Nomura Research Institute

Email: n-sakimura@nri.co.jp
URI: http://nat.sakimura.org/