JOSE Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 10, 2016

M. Jones
Microsoft

August 9, 2015

JWS Unencoded Payload Option
draft-ietf-jose-jws-signing-input-options-01

Abstract

JSON Web Signature (JWS) represents the payload of a JWS as a base64url encoded value and uses this value in the JWS Signature computation. While this enables arbitrary payloads to be integrity protected, some have described use cases in which the base64url encoding is unnecessary and/or an impediment to adoption, especially when the payload is large and/or detached. This specification defines a means of accommodating these use cases by defining an option to change the JWS Signing Input computation to not base64url-encode the payload. This option is intended to broaden the set of use cases for which the use of JWS is a good fit.

Table of Contents

1. Introduction

The "JSON Web Signature (JWS)" [JWS] specification defines the JWS Signing Input as the input to the digital signature or MAC computation, with the value ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload)). While this works well in practice for many use cases, including those accommodating arbitrary payload values, other use cases have been described in which base64url encoding the payload is unnecessary and/or an impediment to adoption, particularly when the payload is large and/or detached.

This specification introduces a new JWS Header Parameter value that generalizes the JWS Signing Input computation in a manner that makes base64url encoding the payload selectable and optional. The primary set of use cases where this enhancement may be helpful are those in which the payload may be very large and where means are already in place to enable the payload to be communicated between the parties without modifications. Appendix F of [JWS] describes how to represent JWSs with detached content, which would typically be used for these use cases.

The advantages of not having to base64url-encode a large payload are that allocation of the additional storage to hold the base64url-encoded form is avoided and the base64url-encoding computation never has to be performed. In summary, this option can help avoid unnecessary copying and transformations of the potentially large payload, resulting in sometimes significant space and time improvements for deployments.

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.

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

This specification uses the same terminology as the "JSON Web Signature (JWS)" [JWS] and "JSON Web Algorithms (JWA)" [JWA] specifications.

3. The "b64" Header Parameter

This Header Parameter modifies the JWS Payload representation and the JWS Signing Input computation in the following way:

b64
: The "b64" (base64url-encode payload) Header Parameter determines whether the payload is represented in the JWS and the JWS Signing Input as ASCII(BASE64URL(JWS Payload)) or as the JWS Payload value itself with no encoding performed. When the "b64" value is "false", the payload is represented simply as the JWS Payload value; otherwise, it is represented as ASCII(BASE64URL(JWS Payload)). The "b64" value is a JSON boolean, with a default value of "true". Note that unless the payload is detached, many payload values would cause errors parsing the resulting JWSs, as described in Section 5.

The following table shows the JWS Signing Input computation, depending upon the value of this parameter:

"b64"	JWS Signing Input Formula
true	ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload))
false	ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.') || JWS Payload

4. Examples

This section gives examples of JWSs showing the difference that using the "b64" Header Parameter makes. The resulting JWSs both use the JWS Compact Serialization and both use the JWS Payload value [36, 46, 48, 50]. This octet sequence represents the ASCII characters "$.02"; its base64url-encoded representation is "JC4wMg".

The following table shows a set of Header Parameter values without using a false "b64" Header Parameter value and a set using it, with the resulting JWS Signing Input values represented as ASCII

characters:

JWS Protected Header	JWS Signing Input Value
{"alg":"HS256"}	eyJhbGciOiJIUzI1NiJ9.JC4wMg
{"alg":"HS256","b64":false}	eyJhbGciOiJIUzI1NiIsImI2NCI6ZmFsc2V 9.$.02

These examples use the HMAC key from Appendix A.1 of [JWS], which is represented below as a JWK [JWK] (with line breaks within values for display purposes only):

```
{
 "kty":"oct",
 "k":"AyM1SysPpbyDfgZld3umj1qzKObwVMkoqQ-EstJQLr_T-1qS0gZH75
      aKtMN3Yj0iPS4hcgUuTwjAzZr1Z9CAow"
}
```

The rest of this section shows complete representations for the two JWSs above.

4.1. Example with {"alg":"HS256"}

The complete JWS representation for this example using the JWS Compact Serialization (with line breaks for display purposes only) is:

```
eyJhbGciOiJIUzI1NiJ9
.
JC4wMg
.
5mvfOroL-g7HyqJoozehmsaqmvTYGEq5jTI1gVvoEoQ
```

Note that this JWS uses only features defined by [JWS] and does not use the new "b64" Header Parameter. It is the "control", so that differences when it is used can be easily seen.

4.2. Example with {"alg":"HS256","b64":false}

The complete JWS representation for this example using the JWS Compact Serialization (with line breaks for display purposes only) is:

```
eyJhbGciOiJIUzI1NiIsImI2NCI6ZmFsc2V9
.
.
```

```
GsyM6AQJbQHY8aQKCbZSPJHzMRWo3HKIlcDuXof7nqs
```

Note that the payload "$.02" cannot be represented in this JWS in its unencoded form because it contains a period ('.') character, which would cause parsing problems. This JWS is therefore shown with a detached payload.

The complete JWS representation for this example using the flattened JWS JSON Serialization is:

```
{
 "protected":
  "eyJhbGciOiJIUzI1NiIsImI2NCI6ZmFsc2V9",
 "payload":
  "$.02",
 "signature",
  "GsyM6AQJbQHY8aQKCbZSPJHzMRWo3HKIlcDuXof7nqs"
}
```

If using a detached payload with the JWS JSON Serialization, the "payload" element would be omitted.

5. Unencoded Payload Content Restrictions

When the "b64" value is "false", unless the payload is detached, as described in Appendix F of [JWS], many payload values would cause errors parsing the resulting JWSs, depending upon the serialization used.

When using the JWS Compact Serialization, unencoded non-detached payloads including period ('.') characters would cause parsing errors; such payloads MUST NOT be used with the JWS Compact Serialization. Similarly, if a JWS using the JWS Compact Serialization and a non-detached payload is to be transmitted in a context that requires URL safe characters, then the application must ensure that the payload contains only the URL-safe characters 'a'-'z', 'A'-'Z', '0'-'9', dash ('-'), underscore ('_'), and tilde ('~'); non-detached payloads using characters outside this set SHOULD NOT be used.

When using the JWS JSON Serialization, unencoded non-detached payloads must consist of the octets of the UTF-8 encoding of a sequence of Unicode code points that are representable in a JSON string. The payload value is determined after performing any escape processing (as per Section 8.3 of RFC 7159 [RFC7159]) and UTF-8-encoding the resulting Unicode code points. This means, for instance, that these payloads represented as JSON strings are

equivalent ("$.02", "\u0024.02"). Unassigned Unicode code point values MUST NOT be used to represent the payload.

Note that because the character sets that can be used for unencoded non-detached payloads differ between the two serializations, some JWSs using a "b64" value of "false" cannot be syntactically converted between the JWS JSON Serialization and the JWS Compact Serialization. See Section 7 for security considerations on using unencoded payloads.

6. Intended Use by Applications

It is intended that application profiles specify up front whether "b64" with a "false" value is to be used by the application or not, with it then being consistently applied in the application context. For instance, an application that uses detached payloads might specify that "b64" with a "false" value always be used. It is not intended that this parameter value be dynamically varied with different payloads for the same application.

JSON Web Tokens (JWTs) [JWT] MUST NOT use "b64" with a "false" value.

7. Security Considerations

[JWS] base64url-encodes the JWS Payload to restrict the character set used to represent it to characters that are distinct from the delimiters that separate it from other JWS fields. Those delimiters are the period ('.') character for the JWS Compact Serialization and the double-quote ('"') character for the JWS JSON Serialization. This encoding also intentionally excludes characters whose representations may require escaping in some contexts and excludes whitespace and line breaks, as all of these can result in changes to the payload during transmission.

When the "b64" (base64url-encode payload) value is "false", these properties are lost. It then becomes the responsibility of the application to ensure that payloads only contain characters that will not cause parsing problems for the serialization used, as described in Section 5, and that the payload will not be modified during transmission.

8. IANA Considerations

8.1. JWS and JWE Header Parameter Registration

This specification registers the "b64" Header Parameter defined in Section 3 in the IANA "JSON Web Signature and Encryption Header Parameters" registry [IANA.JOSE] established by [JWS].

8.1.1. Registry Contents

- Header Parameter Name: "b64"
- Header Parameter Description: Base64url-Encode Payload
- Header Parameter Usage Location(s): JWS
- Change Controller: IETF
- Specification Document(s): Section 3 of [[this document]]

9. References

9.1. Normative References

[IANA.JOSE] IANA, "JSON Object Signing and Encryption (JOSE)", <http://www.iana.org/assignments/jose>.

[JWA] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, May 2015, <http://www.rfc-editor.org/info/rfc7518>.

[JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, May 2015, <http://www.rfc-editor.org/info/rfc7515>.

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, May 2015, <http://www.rfc-editor.org/info/rfc7519>.

[RFC20] Cerf, V., "ASCII format for Network Interchange", STD 80, RFC 20, October 1969, <http://www.rfc-editor.org/info/rfc20>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data

Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

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

9.2. Informative References

[JWK] Jones, M., "JSON Web Key (JWK)", RFC 7517, May 2015, <http://www.rfc-editor.org/info/rfc7517>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February 2003, <http://www.rfc-editor.org/info/rfc3447>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

Appendix A. Acknowledgements

Anders Rundgren, Richard Barnes, Phillip Hallam-Baker, Jim Schaad, Matt Miller, Martin Thomson, and others have all made the case at different times for being able to use a representation of the payload that is not base64url-encoded in contexts in which it safe to do so.

Thanks to Axel Nennker, Anders Rundgren, Nat Sakimura, Jim Schaad, and Matias Woloski for their reviews of the specification.

Appendix B. Document History

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

-01

- Removed the "sph" (secure protected header) Header Parameter.
- Changed the title to "JWS Unencoded Payload Option".

- Added the section "Unencoded Payload Content Restrictions".

- Added an example using the JWS JSON Serialization.

-00

- Created the -00 JOSE working group draft from draft-jones-jose-jws-signing-input-options-00 with no normative changes.

Author's Address

Michael B. Jones
Microsoft

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