JOSE Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 28, 2015

M. Jones
Microsoft

May 27, 2015

JWS Signing Input Options
draft-jones-jose-jws-signing-input-options-00

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.

Also, JWS includes a representation of the JWS Protected Header and a period ('.') character in the JWS Signature computation. While this cryptographically binds the protected Header Parameters to the integrity protected payload, some of have described use cases in which this binding 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 include a representation of the JWS Protected Header and a period ('.') character in the JWS Signing Input.

These options are intended to broaden the set of use cases for which the use of JWS is a good fit.

material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 28, 2015.

Copyright Notice

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 either the base64url encoding of the payload or the addition of information beyond the payload itself to the JWS Signing Input present a burden to implementation and adoption, particularly when the payload is large and/or detached.

This specification introduces two new JWS Header Parameter values that generalize the JWS Signing Input computation in a manner that makes these two behaviors selectable and optional.

The primary set of use cases where these enhancements 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 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.

The advantages of not having to add a representation of the JWS Protected Header and a period ('.') character as a prefix to the payload representation in JWS Signing Input are similar. If the prefix is present in the JWS Signing Input, then space has to be allocated to hold the JWS Signing Input that includes a copy of the (large) payload representation. This is necessary because most cryptographic libraries underlying the JWS implementation will require that the JWS Signing Input be represented as a contiguous octet sequence.

In summary, these options 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. Header Parameters Defined

[JWS] defines the JWS Signing Input value to be ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload)). The following Header Parameters are defined that modify the JWS Signing Input computation in the following ways:

sph
: The "sph" (secure protected header) Header Parameter determines whether the octet sequence ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.') is included in the JWS Signing Input or not. When the "sph" value is "false", it is not included. The "sph" value is a JSON [RFC7159] boolean, with a default value of "true".

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 payload 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, as described in Appendix F of [JWS], many payload values (such as those including period ('.') characters) would cause errors parsing the resulting JWSs, depending upon the serialization used. See Section 7.2 for

limitations on using unencoded payloads.

The following table shows the JWS Signing Input computation, depending upon the values of these parameters:

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

4. Examples

This section gives some examples of possible uses of these Header Parameters and resulting JWSs using the JWS Compact Serialization. All these examples 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 different sets of Header Parameter values and the resulting JWS Signing Input values represented as ASCII characters:

JWS Protected Header	JWS Signing Input Value
{"alg":"HS256"}	eyJhbGciOiJIUzI1NiJ9.JC4wMg
{"alg":"HS256","sph":false}	JC4wMg
{"alg":"HS256","b64":false}	eyJhbGciOiJIUzI1NiIsImI2NCI6ZmFs c2V9.$.02
{"alg":"HS256","sph":false,"b64":false}	$.02

All these examples use this HMAC key from Appendix A.1 of [JWS], which is represented 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 using the JWS Compact Serialization for the four JWSs with the sets of Header Parameters listed 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 uses neither of the new Header Parameters defined in Section 3. It is the "control", so that differences from it when the new Header Parameters are used can be easily seen.

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

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

```
eyJhbGciOiJIUzI1NiIsInNwaCI6ZmFsc2V9
.
JC4wMg
.
ojui4Wd9BM62Ag1zcfUAPHZGj_nWl2oHEJN1QIVH4IM
```

4.3. 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.

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

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

```
eyJhbGciOiJIUzI1NiIsInNwaCI6ZmFsc2UsImI2NCI6ZmFsc2V9
.
.
diN05PDK8714HY7InlOPYcJ8dIBpr-JVNA_20hkfnSc
```

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.

5. Intended Use by Applications

It is intended that application profiles specify up front whether the "sph" and/or "b64" are to be used by the application, with them then being consistently applied in the application context. For instance, an application using potentially large detached payloads might specify that both "sph" and "b64" always be used. Another application that always uses alphanumeric payloads might specify that "b64" always be used. It is not intended that these parameter values be dynamically varied with different payloads for the same application.

6. IANA Considerations

6.1. JWS and JWE Header Parameter Registration

This specification registers the "mac" (MAC algorithm) Header Parameter defined in Section 3 in the IANA JSON Web Signature and Encryption Header Parameters registry defined in [JWS].

6.1.1. Registry Contents

- Header Parameter Name: "sph"
- Header Parameter Description: Secure Protected Header
- Header Parameter Usage Location(s): JWS
- Change Controller: IETF
- Specification Document(s): Section 3 of [[this document]]

- 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]]

7. Security Considerations

7.1. Unsecured Header Parameters

The same security considerations apply to the case when the "sph" (secure protected header) value is "false" as apply when Header Parameters are carried in the JWS Unprotected Header. These are described in Section 10.7 of [JWS].

7.2. Unencoded Payloads

[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" (secure protected header) 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 and that the payload will not be modified during transmission. For instance, this means that payloads cannot contain period ('.') characters when using the JWS Compact Serialization, unless the payload is detached. Similarly, if the JWS 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 ('~'), unless it is detached.

8. References

8.1. Normative References

[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>.

[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, March 1997.

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

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

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

8.2. 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, February 1997.

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

[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 using a representation of the payload that is not base64url-encoded and/or not cryptographically binding the Header Parameter values to the integrity protected payload in contexts in which it safe to do so.

Appendix B. Document History

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

-00

o Created draft-jones-jose-jws-signing-input-options.

Author's Address

Michael B. Jones
Microsoft

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