1. Introduction

JSON Web Signature (JWS) uses JSON data structures and the Base64url encoding for representing content (either a payload or a key) in a compact manner.

The goal of JWSs is for the content to be compact, signed, and verified. To do so, the content is encoded, a signature is computed, and the encoded result and the signature are encoded again. The encoded result and the signature are then concatenated together.

2. Terminology

This section defines terminology used throughout this document.

3. JSON Web Signature (JWS) Overview

This section describes how JWSs are used in practice.

Appendix A.

This appendix provides additional information that may be helpful.

Appendix B.

This appendix contains additional technical background.

Appendix C.

This appendix contains implementation details.

Appendix D.

This appendix contains a list of private use names.

4. JWS Header

This section describes the JWS Header.

5. JWS Payload

This section describes the JWS Payload.

6. JWS Algorithm

This section describes the JWS Algorithm.

7. References

This section contains references to related standards and specifications.

Appendix A.1

This appendix contains additional information that may be helpful.

Appendix A.2

This appendix contains additional technical background.

Appendix A.3

This appendix contains implementation details.

Appendix A.4

This appendix contains a list of private use names.
This specification defines the use of ECDSA with the P-256 curve and the SHA-256 cryptographic hash function. The Elliptic Curve Digital Signature Algorithm (ECDSA) is a cryptographic algorithm that provides digital signatures for messages. Comparisons between JSON strings and other Unicode strings must be performed as a case-sensitive comparison of the UTF-8 representations of the strings.

To create a JWS, one MUST follow these steps:

1. Create a JSON object containing a set of desired header parameters and values. These parameters and values are the same as those used in a JWE but are used in a different structure. This JSON object is called the JWS Header, and is defined in Section 4.
2. Create a JSON object containing the signed contents of the JWS. This JSON object is called the JWS Payload, and is defined in Section 5.
3. Sign the JWS Payload using an appropriate algorithm. The signature is called the JWS Signature, and is defined in Section 6.
4. Join the JWS Header, a period ('.') character, and the JWS Signature to form a complete JWS. Three example JWSs using RSA with SHA-1, RSA with SHA-256, and ECDSA with SHA-256 are shown in Table 3.

The JWS Signature contains a base64url encoded HMAC, as defined in Section 3.4.2, or a base64url encoded RSASSA-PKCS1-v1_5 signature, as defined in Section 3.4.3, or a base64url encoded ECDSA signature, as defined in Section 3.4.4. The base64url encoding of an HMAC is generated as follows:

- Apply the HMAC SHA-256 algorithm to the UTF-8 representation of the JWS Signing Input.
- Base64url encode the previously generated HMAC, as defined in this specification.

The base64url encoding of a RSASSA-PKCS1-v1_5 signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

The base64url encoding of an ECDSA signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

To create a JWS, one MUST follow these steps:

1. Create a JSON object containing a set of desired header parameters and values. These parameters and values are the same as those used in a JWE but are used in a different structure. This JSON object is called the JWS Header, and is defined in Section 4.
2. Create a JSON object containing the signed contents of the JWS. This JSON object is called the JWS Payload, and is defined in Section 5.
3. Sign the JWS Payload using an appropriate algorithm. The signature is called the JWS Signature, and is defined in Section 6.
4. Join the JWS Header, a period ('.') character, and the JWS Signature to form a complete JWS. Three example JWSs using RSA with SHA-1, RSA with SHA-256, and ECDSA with SHA-256 are shown in Table 3.

The JWS Signature contains a base64url encoded HMAC, as defined in Section 3.4.2, or a base64url encoded RSASSA-PKCS1-v1_5 signature, as defined in Section 3.4.3, or a base64url encoded ECDSA signature, as defined in Section 3.4.4. The base64url encoding of an HMAC is generated as follows:

- Apply the HMAC SHA-256 algorithm to the UTF-8 representation of the JWS Signing Input.
- Base64url encode the previously generated HMAC, as defined in this specification.

The base64url encoding of a RSASSA-PKCS1-v1_5 signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

The base64url encoding of an ECDSA signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

To create a JWS, one MUST follow these steps:

1. Create a JSON object containing a set of desired header parameters and values. These parameters and values are the same as those used in a JWE but are used in a different structure. This JSON object is called the JWS Header, and is defined in Section 4.
2. Create a JSON object containing the signed contents of the JWS. This JSON object is called the JWS Payload, and is defined in Section 5.
3. Sign the JWS Payload using an appropriate algorithm. The signature is called the JWS Signature, and is defined in Section 6.
4. Join the JWS Header, a period ('.') character, and the JWS Signature to form a complete JWS. Three example JWSs using RSA with SHA-1, RSA with SHA-256, and ECDSA with SHA-256 are shown in Table 3.

The JWS Signature contains a base64url encoded HMAC, as defined in Section 3.4.2, or a base64url encoded RSASSA-PKCS1-v1_5 signature, as defined in Section 3.4.3, or a base64url encoded ECDSA signature, as defined in Section 3.4.4. The base64url encoding of an HMAC is generated as follows:

- Apply the HMAC SHA-256 algorithm to the UTF-8 representation of the JWS Signing Input.
- Base64url encode the previously generated HMAC, as defined in this specification.

The base64url encoding of a RSASSA-PKCS1-v1_5 signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

The base64url encoding of an ECDSA signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

To create a JWS, one MUST follow these steps:

1. Create a JSON object containing a set of desired header parameters and values. These parameters and values are the same as those used in a JWE but are used in a different structure. This JSON object is called the JWS Header, and is defined in Section 4.
2. Create a JSON object containing the signed contents of the JWS. This JSON object is called the JWS Payload, and is defined in Section 5.
3. Sign the JWS Payload using an appropriate algorithm. The signature is called the JWS Signature, and is defined in Section 6.
4. Join the JWS Header, a period ('.') character, and the JWS Signature to form a complete JWS. Three example JWSs using RSA with SHA-1, RSA with SHA-256, and ECDSA with SHA-256 are shown in Table 3.

The JWS Signature contains a base64url encoded HMAC, as defined in Section 3.4.2, or a base64url encoded RSASSA-PKCS1-v1_5 signature, as defined in Section 3.4.3, or a base64url encoded ECDSA signature, as defined in Section 3.4.4. The base64url encoding of an HMAC is generated as follows:

- Apply the HMAC SHA-256 algorithm to the UTF-8 representation of the JWS Signing Input.
- Base64url encode the previously generated HMAC, as defined in this specification.

The base64url encoding of a RSASSA-PKCS1-v1_5 signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

The base64url encoding of an ECDSA signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

To create a JWS, one MUST follow these steps:

1. Create a JSON object containing a set of desired header parameters and values. These parameters and values are the same as those used in a JWE but are used in a different structure. This JSON object is called the JWS Header, and is defined in Section 4.
2. Create a JSON object containing the signed contents of the JWS. This JSON object is called the JWS Payload, and is defined in Section 5.
3. Sign the JWS Payload using an appropriate algorithm. The signature is called the JWS Signature, and is defined in Section 6.
4. Join the JWS Header, a period ('.') character, and the JWS Signature to form a complete JWS. Three example JWSs using RSA with SHA-1, RSA with SHA-256, and ECDSA with SHA-256 are shown in Table 3.

The JWS Signature contains a base64url encoded HMAC, as defined in Section 3.4.2, or a base64url encoded RSASSA-PKCS1-v1_5 signature, as defined in Section 3.4.3, or a base64url encoded ECDSA signature, as defined in Section 3.4.4. The base64url encoding of an HMAC is generated as follows:

- Apply the HMAC SHA-256 algorithm to the UTF-8 representation of the JWS Signing Input.
- Base64url encode the previously generated HMAC, as defined in this specification.

The base64url encoding of a RSASSA-PKCS1-v1_5 signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.

The base64url encoding of an ECDSA signature is generated as follows:

- Generate a digital signature of the UTF-8 representation of the JWS Signing Input using RSASSA-PKCS1-V1_5-SIGN.
- Base64url encode the previously generated signature, as defined in this specification.
The following byte array contains the UTF-8 characters for
"the UTF-8 representation of X", we mean "the
UTF-8 representation of the binary value of X".

We consider using a checksum provided.

The following items remain to be done in this draft:
- Determine if allowing JWS signing with the
  algorithm using SHA-1 is an insecure
  practice.
- Since JWE and JWS algorithms may not be
  the same, it is important to clarify the
  semantic differences in JWE and JWS.
- The following bytes are encrypted using
  the specified key:

The JWS Payload used in this example:

```
34, 72, 83, 50, 53, 54, 34, 125
```

The following byte array contains the UTF-8 characters for
"the UTF-8 representation of X", we mean "the
UTF-8 representation of the binary value of X".

We consider using a checksum provided.

The following items remain to be done in this draft:
- Determine if allowing JWS signing with the
  algorithm using SHA-1 is an insecure
  practice.
- Since JWE and JWS algorithms may not be
  the same, it is important to clarify the
  semantic differences in JWE and JWS.
- The following bytes are encrypted using
  the specified key:

The JWS Payload used in this example:

```
34, 72, 83, 50, 53, 54, 34, 125
```

Additional algorithms MAY be used to protect JWSs with
values will be different. The consequence of this is
additional algorithms is performed
related to the elliptic curve
keys, and the JWS Signing Input is signed using
"The ECDSA P-256 SHA-256 signature for a JWS is validated as follows:
1. The output is the Encoded JWS Signature for the JWS.
2. The ECDSA P-256 SHA-256 - just with correspondingly longer key
3. The JWS Signing Input is signed using
values will be different. The consequence of this is
additional algorithms is performed
related to the elliptic curve
keys, and the JWS Signing Input is signed using
"The ECDSA P-256 SHA-256 signature for a JWS is validated as follows:
1. The output is the Encoded JWS Signature for the JWS.
2. The ECDSA P-256 SHA-256 - just with correspondingly longer key
3. The JWS Signing Input is signed using
null
A.3.2. Decoding
and SHA-512 ECDSA using algorithm P-384 curve
ECDSA using algorithm SHA-256 algorithm SHA-384 algorithm RSA using algorithm SHA-512 algorithm SHA-384 HMAC using algorithm hash SHA-256 algorithm

Decoding the JWS from this example requires processing the

Algorithm d
Name

Algorithm

-00
-02
-03
-51
-50
-49
-48
-47
-46
-45
-44
-43
-42
-41
-40
-39
-38
-37
-36
-35
-34
-33
-32
-31
-30
-29
-28
-27
-26
-25
-24
-23
-22
-21
-20
-19
-18
-17
-16
-15
-14
-13
-12
-11
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1

If the length mod 4 is 3, one '=' padding character is added;

if the length mod 4 is 0, no padding characters are added.

Specifically, if the length mod 4 is 3, one '=' padding character is added; if the length mod 4 is 0, no padding characters are added.

This split introduced no

this example.

Likewise the terms "Decoded JWS Header" and "Decoded JWS Header" are now used, simplified terminology to better match JWE, where the terms "Decoded JWS Header" and "Decoded JWS Header" are now used, simplified terminology to better match JWE, where the terms "Decoded JWS Header" and "Decoded JWS Header" are now used, simplified terminology to better match JWE, where the terms "Decoded JWS Header" and "Decoded JWS Header" are now used, simplified terminology to better match JWE, where the terms "Decoded JWS Header" and "Decoded JWS Header" are now used, simplified terminology to better match JWE, where the terms "Decoded JWS Header" and "Decoded JWS Header" are now used, simplified terminology to better match JWE.